Using GIS Tool for Presenting Spatial Data: Case Study Nakorn Pathom Province

S. Choimeun¹, N. Phumejaya¹, S. Pomnakchim¹, C. Chantrapornchai²

¹ Computer Program, Faculty of Science, Nakorn Pathom Rajabhat University, Thailand
² Department of Computing, Faculty of Science, Silpakorn University, Nakorn Pathom, Thailand

E-mail: ctana@su.ac.th

Abstract

We propose a tool and library for collecting spatial data based on Google Maps API. The tool is implemented using AJAX and XML technology. Based on Google Earth®, the user can define the interested regions by using our drawing tool. After subregions are defined, the information about the regions can be put. These information can put in the local database. Then the user can view the data by regions or types on their map. We provide library supports for drawing, overlays, graph presentation and comparison. The tool supports the KML and NMA files where the user specification can be export and the offline data in such a form can be imported to the system as well. We demonstrate a case of using the tool to collect the spatial data in agriculture area.

Keywords: Google Maps API, Spatial data, AJAX, XML, NMEA.

1. Introduction

The popularity in GIS applications leads to development in various GIS tools. The software helps manage geographical information in many ways such as collecting associated data for landmark in traveling and directions. Google maps become popular tools which provide maps as well as tools for user data. With its open technology, the users can share information, ideas, or develop their own applications using Google maps API [3]. Also, there are others GIS technology. Though they are powerful, some of them are commercials which are still expensive. To develop applications based on them will be costly and complex.

Various applications are developed based on Google Maps such as the work by NECTEC [2]. This work proposed a way to estimate the travel time in Bangkok using Google Maps. The work by WLHP [9] presented a GIS application to display pollution. Hrvoje Podnar et.al. [1] visualize the students population using Google Maps. Shinji Kobayashi et.al. [6] use Google Maps to refer the patients to the nearest hospitals. Bruce A. Ralston [8] presents a tool to generate areas using KML for population survey in the United States.

In this work, we take advantage of the Google Maps API, including the online maps, and geographical coordinates. We develop the tools to record GIS data from various sources such as user inputs, and from KML as well as NMA formats. Then, the data is recorded in the database for further development in GIS applications.
2. Backgrounds

In this section, we briefly describe the necessary backgrounds in GIS technology as it is related to our work.

Geographical information systems are the systems which take advantages of spatial data and define the relationships to the interest issues. The information may be the home address, for example, which is mapped to the spatial data such as the latitude and longitude. Then the database further stores more information about the address.

The important part in developing the GIS applications is the data capture and storage. The data which includes the spatial coordinate and associative database needs to be provided in any valid means. The spatial data may be imported from interactive equipments or files. Then the coordinate is converted appropriately. Many applications are available to help import data such as ArcInfo (http://www.esri.com/software/arcgis/arcinfo/index.html), ArcView (http://www.esri.com/software/arcview/index.html), Mapinfo (http://www.mapinfo.com), ERDAS (http://www.erdas.com) etc. After that, the associated database is imported by any normal program. To store spatial information, it is common that the vector format is used. The storage may keep only points, lines, or areas [3].

Google Maps [5] is an open technology that provides maps. The user can use the web browser to look at the maps. Several convenient tools are built-in such as zooming in and out, marking, view satellite data, etc. While Google maps provide map information, Google Earth [4] is a software to view satellite data in high resolutions. It can save the data in KML file. The KML file is extended from XML which is used to describe the data[11-12]. The KML file can be used together with the Google as shown in Figure 1. Besides the KML file, NMEA 0183 is a standard for GPS data. It is the protocol used by GPS producers to communicate with other devices [13]. In this work, we also consider to import data from GPS device as well.

![Fig. 1. KML Used in Google Maps [8].](image-url)
3. Tool Architecture

In the development tool, it contains 5 modules as in Figure 2.

1. User module: it is to communicate with the users by importing or exporting spatial data in NMA files or KML files, or by user input specification.
2. GPS module: it is used to store data from GPS devices and convert NMA files to XML files for display.
3. Converter module: it is to convert KML files to XML files for the display purpose.
4. Presentation module: it takes the spatial data in the database table and convert to XML for display. When converted to XML, we also tag whether it is a point, line or area.
5. Drawing model: it will take the XML files from other modules to overlay on Google Maps in various forms according to the coordinate specified in the database.

Figure 3 presents how the tool interacts to the other parts. It will invoke the online map from Google Maps and use XML to creates markup. It communicates with Google Earth using KML files. It also reads the data from to the GPS receivers via NMA files.
4. Tool Functionality and Library Codes

Our developed tool is called NPRU tool. We test the usability of the tool for each function as shown in Table 1. It is seen that the user can specify using points, lines and areas. These inputs support the functions specified in each column.

Table 1. Functionality of the Tool.

<table>
<thead>
<tr>
<th>Function</th>
<th>Add</th>
<th>Delete</th>
<th>Update</th>
<th>Save</th>
<th>Import</th>
<th>Export</th>
<th>Display</th>
<th>Symbol</th>
<th>Attributes</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>points</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>lines</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>area</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

The comparison of our tool and others are shown in Table 2. Column “Network connection” means the tools required network connection or not. Column “Satellite View” implies the tool can show satellite view. Column “Data Format” implies the data format that is supported by the tools. Column “GPS Support” implies the tool can read data from GPS or not. It is seen that our tool is capable about the same as Google Maps except that we are interested in the support of KML and data can store locally where it can be manipulated easily.

Table 2. Comparison of the Tools.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Network Connection</th>
<th>Satellite View</th>
<th>Data Format</th>
<th>GPS Support (NMA)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google Maps</td>
<td>100%</td>
<td>Yes</td>
<td>RSS, data stored on google server</td>
<td>Mobile support, Yes</td>
</tr>
<tr>
<td>ArcGIS</td>
<td>Some</td>
<td>Plugin</td>
<td>SHP, db, *KML</td>
<td>Yes</td>
</tr>
<tr>
<td>MapInfo</td>
<td>Some</td>
<td>Plugin</td>
<td>SHP, db, *KML</td>
<td>Yes</td>
</tr>
<tr>
<td>Google Earth</td>
<td>Some</td>
<td>Yes</td>
<td>KML, KMZ</td>
<td>No</td>
</tr>
<tr>
<td>Point Asia</td>
<td>100%</td>
<td>Selective</td>
<td>Cannot record</td>
<td>Yes</td>
</tr>
<tr>
<td>NPRU</td>
<td>100%</td>
<td>Yes</td>
<td>KML, data stored in database</td>
<td>Yes</td>
</tr>
</tbody>
</table>

We use our tool to develop GIS for agriculture in Nakorn Pathom province. In the area, it is the province that contains 25 sub-district and 217 villages. Figures 4-5 show the user interface where a user can overlay on the Google Maps. The user uses the provided drawing tools to create the overlays. Figure 6 shows the corresponding function implemented for polygons. For example, Figure 6(a) is the function for creating polygon area and Figure 6(b) is the function to calculate polygon area.

In the polygon area, we need to define an icon to store information about items in the area such as rice, vegetables and is used as a marker on the map. The icon stores information such as a type of products, latitude, longitude, amount of product. The code for creating icon is in Figure 7. For example, Figure 8 extracts the coordinates from the icon area and displayed from the database.
Fig 4. Defining Regions.

Fig 5. Define Sub-regions.

(a) Indicating Polygon Area.
(b) Calculating Polygon Area.

Fig 6. Example Code for Polygon.

```javascript
function calcPolygonArea(polygon) {
  // Calculate the area of a polygon.
  var result = polygon.getArea() / 100.0
  return result;
}
```

Fig 7. Code for Implement Icon.
We have an input screen for the users to input data into each icon. The user data are stored in the local database. In the following application, we are interested in the agriculture product in the area. The production data are given in the database. From the tool, the satellite view can be shown Figure 9. The polygon is shown in the overlay format. The code for overlays are in Figures 10-11.

```javascript
function hideOverlays(poly)
  this.poly = poly;
  for(var i=0; i < this.poly.length; i++)
    this.poly[i].hide();
}

(a) Hiding Overlays.

function showOverlays(poly)
  this.poly = poly;
  for(var i=0; i < this.poly.length; i++)
    this.poly[i].show();
}

(b) Showing Overlays.

Fig 10. Code for Overlays.
Fig 11. Code for Displaying Overlay Regions.

The information in the database can be linked and displayed in many ways. The user can search the area as shown in Figure 12. The code for this is in Figure 13.
Fig 12. Input Data for Searching.

```php
function make_autocom(autoObj, showObj) {
    var mkAutoObj = autoObj;
    var mkServiceObj = showObj;

    new Autocomplete(mkAutoObj, function() {
        this.setValue = function(id) {
            document.getElementById(mkServiceObj).value = id;
        }
        if (this.ishotclick) {
            return;
        }
        else {
            var selectId = document.getElementById("selectId").value
            return "," + this.value + "=" + selectId;
        }
    });
}
</php>

Fig 13. Code for Searching.

```
The production can be shown by regions and types in graphs. Figure 11 compares the production of corns by months at Thayang district. The code to select production data by month is shown in Figure 12. Also the code to compare the production is shown in Figure 13.

```
function call_times(area){
    var f1 = document.getElementById("form_0").value;
    plant = mn;
    var xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
    if(f1 == "01"){
        url = "../dbFile/timeplant_01.php";
    }
    else if(f1 == "02"){
        url = "../dbFile/timeplant_02.php";
    }
    url += "?area="+area+"&plant="+plant+"&dummy="+ (new Date()).getTime();
    xmlhttp.open("GET", url, false);
    xmlhttp.send(mn);
    createTimes(xmlhttp.responseText);
}
```

**Fig 11. Display Corn Production by Months.**

**Fig 12. Code to Select Corn Production by Months.**
function createTimes(responseText) {
  var resxml = responseText;
  var xml = XOM.parse(resxml);
  var time_detail = "";
  var tcst_ch = 2;
  var bgCh = "";
  var nst = [];
  var pstr = "";
  var hstr = [];
  var pstr = "";

  var states = xml.documentElement.getElementsByTagName("state");
  var year = states[0].getElementsByTagName("year");
  var month = states[0].getElementsByTagName("month");
  var mon = states[0].getElementsByTagName("mon");
  var sum_plant = states[0].getElementsByTagName("sum_plant");
  var sum_area = states[0].getElementsByTagName("sum_area");
  var homes = states[0].getElementsByTagName("home");
  var units = states[0].getElementsByTagName("unit");
  var price = states[0].getElementsByTagName("price");

  for (var j = 0; j < year.length; j++) {
    var sumPlant = "";
    var sumArea = "";
    var sumHome = "";
    var sumPrice = "";
    var pstr = "";
    if (parseInt(sum_plant[j].text) == 0) {
      sumPlant = "ไม่มีข้อมูล";
      pstr = sum_plant[j].text;
    } else {
      if (units[j].text == "ล้านชิ้น") {
        sumPlant = new NumberFormat(parseInt(sum_plant[j].text) / 1000).toFormatted();
        sumPlant += "ล้าน";
        pstr = parseFloat(sum_plant[j].text) / 1000;
      } else if (units[j].text == "แสนชิ้น") {
        sumPlant = new NumberFormat(parseInt(sum_plant[j].text) / 1000).toFormatted();
        sumPlant += "แสน";
        pstr = parseFloat(sum_plant[j].text) / 1000;
      } else if (units[j].text == "หมื่นชิ้น") {
        sumPlant = new NumberFormat(parseInt(sum_plant[j].text) / 1000).toFormatted();
        sumPlant += "หมื่น";
        pstr = parseFloat(sum_plant[j].text) / 1000;
      } else {
        sumPlant = new NumberFormat(sum_plant[j].text).toFormatted() + " " + units[j].text;
        pstr = sum_plant[j].text;
      }
    }
  }
```javascript
if (parseInt(sum_area[j].text) == 0) { sumArea = "ไม่สามารถคำนวณได้"; 
} else { sumArea = new NumberFormat(sum_area[j].text).toFormatted(); }

if (parseInt(homes[j].text) == 0) { sumHome = "ไม่สามารถคำนวณได้"; 
} else { sumHome = new NumberFormat(homes[j].text).toFormatted(); }

if (parseInt(price[j].text) == 0) { sumPrice = "ไม่สามารถคำนวณได้"; 
} else { sumPrice = new NumberFormat(price[j].text).toFormatted(); }

test_ch = test_ch + 1;

if (test_ch%2 == 0) { bgCh = "#E0E0F2"; } else { bgCh = "#FFFFFF"; }

for (var detail = 0; detail < 1000; detail++) {
    detail += "<table>") + "<year[]).text"></td><td align="left">" + (year[i].text + "(<year[]).text"></td><td align="left">" + (mon[j].text + "(<mon[]).text"></td><td align="left">" + (sumPlant[]).text + "(<sumPlant[]).text"></td><td align="left">" + (sumHome[]).text + "(<sumHome[]).text"></td><td align="left">" + (sumPrice[]).text + "(<sumPrice[]).text"></td></tr>");

    n.push(mon[i].text + year[j].text.substr(2, 2));
    p.push(0);
    a.push(sum_area[j].text);
    h.push(homes[j].text);
    pr.push(price[j].text);

    if (x.length <= 0) {
        subM = "";
    } else {
        subM = "x";
    } else {
        subM = "y";
    } else {
        subM = "z";
    }

    for (var li = 0; li < n.length; li++) {
        if (li == n.length - 1) {
            subM = n[li];
            subX = p[li];
            subY = a[li];
            subZ = h[li];
            subP = pr[li];
        } else {
            subM = n[li] + "x";
            subX = p[li] + "y";
            subY = a[li] + "z";
            subZ = h[li] + "w";
            subP = pr[li] + "z";
        } 
    }
}
subM2 = subM;
```
Fig 13. Code to Compare the Data by Months.

The productions are compared by regions using a pie chart and compared by month using line graph shown in Figures 14-15.
5. Conclusion

In this paper, we present a tool to help gather the GIS data. We take advantages of Google Maps so that we can define coordinates in various forms. Also, the spatial data can be gathered from user input, GPS receivers, or Google Earth. Thus, the tool can import the data from these sources using the KML or NMA format. We compare the functionality of the tools and demonstrate the usage of the tool to develop the agriculture application. In the future, the tool will be revised accordingly: display geographical coordinates in many forms, communicate with the GPS device to get the coordinate online and exports the data for into local database for future uses.
Acknowledgement

The research is supported in part by NPRU research funding, and SURDI from Silpakorn University, Thailand. We also would like to thank National Agriculture Department, Thailand for the data.

References

9. West London Housing Partnership, EST, WLHP Measure Map [Online], from http://wlhpmeasuremap.aesolutions.co.uk/map/index