

# Design and Implementation of WebGD Framework

Surya Halim

*School of Electrical Engineering and Computer Science  
Oregon State University  
Corvallis, Oregon, 97331-4602  
[halim@eecs.oregonstate.edu](mailto:halim@eecs.oregonstate.edu)*

## Abstract

A typical Internet map-server application allows only *retrieval* of maps and map-related data. We have been developing *web-based GIS/database* (WebGD) applications that allow users to *modify* geographical features and their associated data from standard web browsers. The code shared by these applications is organized as the WebGD framework. This framework allows the map interface to be customized with *configuration files*. We have also built a WebGD application generator (WebGD-Gen) that *automatically* produces a WebGD application from a database schema. This application generator greatly simplifies the process of creating a complex Web-based GIS/database application and significantly reduces the development time and maintenance cost. The WebGD framework and WebGD-Gen currently support advanced features such as *tight integration* of a web-based map interface with a database, *automatic selection* of the *spatial reference*, and *automatic generation* of web forms. Generated web forms can be used to *insert*, *query*, and *update* geographical features along with their associated data. In this paper, we describe the design and implementation of the WebGD framework.

## 1 Introduction

The Internet is emerging as the major venue for sharing information because it supports globalization and broad-scale interoperability (Bisby, 2000). Dynamic web-based *mapping applications* provide customized on-line *geographical information system* (GIS) that allows users without desktop-based GIS software to perform *spatial queries* and produce maps with standard Web browsers. Such web-based applications can produce maps, statistics, and even subsets of spatial data by using the *location-based data*.

Management of location-based data takes place within social, legal, political, and economic contexts. Hence, a spatial decision support system is required to facilitate *collaboration* and *communication* (Aspinall, 2000). However, this goal conflicts with the limitations of a typical web-based GIS application that allows only the retrieval of maps and map-related data. A web server provides information to the client, but the client cannot feed the information back to the server (Kingston, 1998). This *unidirectional* flow of information is a major problem with the current typical mapping application. Furthermore, the map information and the data stored in a database are not integrated well. Map coverage is limited to *one* region. Creating an interactive web application

with a map interface is time-consuming and error prone. Commercial map servers and geographical database management systems are expensive. Analysis tools implemented are limited. This situation hinders use of Web-based GIS applications in data gathering, analysis, and decision-making.

We have been developing a set of tools that significantly reduces the cost of application development (Wangmutitakul et al., 2003, Wuttiwat et al., 2003; Sano et al., 2003, Wangmutitakul et al., 2004). The code shared by interactive Internet GIS applications is organized as the Web-based GIS/database (WebGD) framework. Most of the complex workings for delivering GIS functions over the Web are included in this framework. Using the WebGD framework, a web-based map interface through which users can *insert*, *query*, and *delete* geographical features and the data associated with them can be created quickly. An important feature of a WebGD application is that it tightly integrates spatial data and associated tabular data, therefore, enables analysis involving location-based data (Sharma 2003). These functions are available to users located at different geographical locations for economical and timely data management.

We have also developed a *web-form generator* that automatically generates code for traditional web-forms from the *schema* of a relational database (Eum et al., 2003). This functionality was recently extended as the WebGD application generator (WebGD-Gen). The new application generator can produce the code of an entire WebGD application from the schema of a relational database and the information on *geometry columns*. The code generated for the map interface can *insert*, *query*, and *delete* geographical features (i.e., points, line-strings, and polygons). When a relational schema and the GIS data for the map layers are available, a *non-customized* application can be quickly assembled with the WebGD framework and WebGD-Gen.

WebGD applications use the following *open-source* software components. PostgreSQL, an *object-relational* database, and PostGIS together to manage geospatial data. PostGIS is an extension of PostgreSQL for GIS applications (Ramsey). MapServer (Univ. of Minnesota) generates maps to be displayed on a web browser by using geospatial data provided by PostGIS. Web pages, including the one that displays the maps, are generated by server-side scripts written in PHP. The PHP Mapscript module interacts with MapServe (McKenna, DM Solutions). When a request to insert or delete a map feature is received by a PHP script, the script directly accesses the PostgreSQL database, using the PostGIS extension. By using open source software components, we are able to develop applications that can run on a PC without the use of any licensed software.

We provide an overview of WebGD applications in Section 2. Section 3 presents the details of the WebGD framework, and Section 4 describes the capabilities of the WebGD-GEN application generator. Section 5 concludes this paper.

## 2 Overview of a WebGD Application

The Web interface of one of the WebGD applications, Oregon Natural Heritage Information System (NHIS), is shown in Figure 1. This application provides a map interface for a copy of the Biotics 4.0 database maintained by the Oregon Natural Heritage Information Center. Biotics 4.0 is a desktop GIS application built on the database developed by NatureServe. The key elements in this database are *element occurrences* (EOs), which are *areas* of land and/or water in which species are or were present (NatureServe, 2002). EO records have both *spatial* and *tabular* data, and the database contains approximately 700 relational tables (Fogelson, 2002). The Biotics Mapper implemented with ArcView by NatureServe provides a map interface that allows EO representations and their associated data to be created, updated, and deleted (NatureServe, 2003). In our implementation, we can perform

these operations with standard Web browsers. Additionally, approximately 3500 web forms are provided for all the tables in the database.



**Figure 1** Interface of WebGD Application NHIS

The NHIS application enables *bi-directional* movement of *geospatial data* as well as ordinary data. Scientists and others with proper authentication can *insert*, *query*, and *delete* geographical features such as, EO polygons, lines, and points, as well as the data associated with them. Queries can be executed by spatially *selecting an area* on the map or by using a traditional web form. In addition, one-meter resolution *digital orthographic quadrangles* (DOQ), or aerial images, are included as a layer. When DOQ images are combined with other map layers such as highways, county boundaries, streams, and streets, locations can be easily pinpointed by taking advantage of features between map layers (Wuttiwat et al., 2003).

The major operations supported by the map interface of a WebGD application are as follows:

1. To retrieve information on the geographical features in the area of interest, user can zoom in or zoom out to that area by using the map navigation tools. In the case of the NHIS application, if the user zooms in enough to the coastal areas or the areas along Interstate highways in Oregon, one-meter resolution aerial photos are displayed. The user can also go to a new area by selecting an entry in the **Quick View** menu.
2. To get information about a geographical feature, the user can select a layer in the legend and **Information** in the function menu, and then click the boundary of the feature.
3. Function **Insert** allows a geographical feature to be added with mouse clicks on the map. **Done** need be pressed after all points are entered in the case of entering multi-point geographical feature.

4. Function **Search by Area** allows the user to retrieve the list of features that are within a *bounding box* specified on the map and only the ones that satisfy user specified search condition. The features that satisfy the search condition are then *highlighted* on the map. Furthermore, the user can select features in the list by marking the checkboxes associated with them. Then, if the map is refreshed, the selected features are highlighted. When the view of geographical features do not fit into the current map area, the user can click on **Get Full Coverage** that will automatically select the largest extent needed to fit all of the geographical features entirely.
5. Data administration interface can be activated by clicking on the **Database** entry in the menu bar below the banner. A tree icon can be clicked to display a *treeview* for browsing. The treeview for **Higher Taxonomy** is the major one.

Several WebGD applications produced with the WebGD framework and WebGD-Gen can be accessed at the following URLs:

```
http://yukon.een.orst.edu/ms_apps/nhis_cs540/gmap75_main.phtml
http://yukon.een.orst.edu/ms_apps/digir/gmap75_main.phtml
http://yukon.een.orst.edu/ms_apps/calflora/gmap75_main.phtml
http://yukon.een.orst.edu/ms_apps/w6grin_cs549/gmap75_main.phtml
http://yukon.een.orst.edu/ms_apps/soilviewer3/gmap75_main.phtml
```

The first application is Natural Heritage Information System. Although this application can cover the whole USA or the world, the data are currently available only for Oregon. The second application provides a map interface for a local database containing Distributed Generic Information Retrieval (DiGIR) records harvested from DiGIR providers, DiGIR (Distributed Generic Information Retrieval) is an XML-based communication protocol for a *federation* of databases managed by natural history museums. The third application provides a prototype map interface for flora occurrences in California. The fourth one is a Web-based mapping application for a plant germplasm collection maintained at Western Regional Plant Introduction Station (USDA-ARS). The fifth one allows the soil information at the location where a mouse click occurs on the map interface to be retrieved. The soil map displayed is for Yolo County in California.

One salient feature of the current WebGD framework is *dynamic switching of spatial references*. Typically, different geographic regions and localities have preferred *map projections* in order to avoid distortions in the maps created (Dana). The framework allows the whole world to be covered with multiple-levels of maps, e.g., the world map, continent maps, and regional maps. The map interface then automatically selects the most suitable projection for the region whose portion is displayed. For example, the world can use the *geographical coordinate system*, the United States the *Albers equal-area projection*, and Oregon the *Lambert conformal conic projection*. Thus, spatial analysis can be performed with the most appropriate projection for a particular area. The *dynamic switching* of the *spatial reference*, the *map file*, the *legend*, the *map commands*, and the *quick view menu* supported by the current WebGD framework allow any part of the world to be covered with its own scale and spatial reference, including regions with one-meter resolution aerial images. This is a very important feature, especially now that the cost of storing aerial images for the entire US has dropped to affordable levels (10 tera-bytes needed to store aerial images for the entire US now cost around \$10,000). Furthermore, many states are putting aerial images in the public domain.

### 3 WebGD Framework

We use *configuration files* extensively as part of the WebGD framework for application customization. A *region configuration* file, a *map layer configuration* file, and a *quick view configuration* file are used to customize a map interface. When the user performs a map operation that results in a change of *the current region*, the spatial reference is automatically switched to that of the new region in order to minimize map distortion.

When the region used by the map interface is changed, the layers in the map legend, the list of the map-navigation and data-manipulation commands, and the quick view list are reconfigured for the new region. These reconfigurations are necessary because different regions may require different sets of map layers, commands, and quick view lists.

#### 3.1 Region Configuration

Whenever the map region changes due to a user action on the map interface, the region configuration file for the new region is loaded dynamically. Then the map interface is customized according to the new region configuration file.

A *region configuration file* for each map region includes the following definitions:

1. the *map projection* for the region,
2. the *name of the region* displayed on the map interface,
3. the *unit of distance measurement*,
4. the *name of the map layer configuration file* for the region, and
5. the *name of the quick view configuration file*.

For example, the region configuration file for the world is as follows:

```
$region = array(
    "gid" => 1,
    "name" => "world",
    "display_name" => "World",
    "srid" => 4326,
    "rank" => 1,
    "units" => "MS_DD",
    "mapfile" => "gmap75_world.map",
    "quickview" => "world_qview.php",
    "legend" => "world_maplayers.php",
    "proj4text" => "+proj=longlat +ellps=WGS84 +datum=WGS84"
);
```

The following options can be specified in a configuration file:

**gid** — the unique number assigned to each region, which is the primary key value of the row representing the region in table `region` in the database.

**name** — the canonical name of the region.

`displayed_name` — the name of the region used on the map interface. This name may include capital letters, white spaces, and other punctuation marks.

`srid` — the *spatial reference identifier* for the map projection assigned to this region.

`rank` — the priority number used in determining the region for the map area to be viewed. The region with the highest rank number is selected among the regions that completely encompass the map area to be viewed.

`units` — the unit of measurement associated with the region.

`mapfile` — the map file to be loaded when the region is selected.

`quickview` — the name of the quick view configuration file for the region.

`legend` — the name of the map layer configuration file for the region.

`proj4text` — the projection string used for the region.

The region configuration file for Oregon, for example, contains the following definitions:

```
$region = array(
    "gid" => 150137,
    "name" => "oregon",
    "display_name" => "Oregon",
    "srid" => 6010,
    "rank" => 150137,
    "units" => "MS_FEET",
    "mapfile" => "gmap75_oregon.map",
    "quickview" => "oregon_qview.php",
    "legend" => "oregon_maplayers.php",
    "proj4text" => "+proj=lcc +lat_1=43.0 +lat_2=45.5 +lat_0=41.75
                  +lon_0=-120.5 +x_0=400000.00000 +y_0=0.0"
);
```

### 3.2 Map Layer Configuration

The *map layer configuration file* provided for a region specifies the layers to be included in the legend and their characteristics. It is possible for multiple map regions to share one map layer configuration file.

The map layer configuration file for the Oregon region, for example, contains the following definitions:

```
$layer_groups = array (
    'grp_eo_py' => array(
```

```

'geom_type' => 'polygon',
'table' => 'eo_py',
'layer_selectable' => true,
'gid_column' => 'gid',
'geom_col' => 'the_geom',
'legend_label' => 'EO Polygons',
'search_script' => 'forms/eo/eo_py_eo_search.phtml',
'select_script' => 'forms/eo/eo_py_eo_select.phtml',
'edit_script' => 'forms/eo/eo_py_edit.phtml',
'normal_layer' => 'eo_py',
'searched_layer' => 'eo_py_searched',
'checked_layer' => 'eo_py_checked',
'selected_layer' => 'eo_py_selected',
'img_src' => 'images/eo_poly.png',
'img_width' => 26,
'img_height' => 26,
'onclick' => 'activate_layer("grp_eo_py")',
'data_srid' => 6010
),
'grp_eosrc_pt' => array(
'geom_type' => 'point',
'table' => 'eosrc_pt',
'layer_selectable' => true,
'gid_column' => 'gid',
'geom_col' => 'the_geom',
'legend_label' => 'EOSRC Points',
'search_script' => 'forms/eo/eosrc_pt_search.phtml',
'select_script' => 'forms/eo/eosrc_pt_select.phtml',
'edit_script' => 'forms/eo/eosrc_pt_edit.phtml',
'normal_layer' => 'eosrc_pt',
'searched_layer' => 'eosrc_pt_searched',
'checked_layer' => 'eosrc_pt_checked',
'selected_layer' => 'eosrc_pt_selected',
'img_width' => 5,
'img_height' => 5,
'onclick' => 'activate_layer("grp_eosrc_pt")',
'data_srid' => 6010
),
. . .

```

The map legend displayed in the map interface according to the configuration file above is shown in Figure 2.



**Figure 2** Map Legend of WebGD Application NHIS

We now explain each option used in a map-layer configuration file:

`geom_type` – the type of geometry features contained in the layer. The value can be `polygon`, `multipolygon`, `linestring` or `multilinestring`. Exact spatial operations performed depend on this type. For example, if the type is `point`, a point is inserted on the map with an insert operation. On the other hand, if the type is `polygon`, a polygon feature can be inserted.

`table` – the name of the table in the database that contains the geometry column for the layer.

`layer_selectable` – the boolean option that determines whether spatial operations can be performed on the layer or not. Some layers, such as those for counties and highways in the above example, are static, and they are not selectable for spatial operations.

`gid_column` – the name of the column that contains the identifiers for the geometry features in the map layer.

`geom_col` – the name of the geometry column containing the geometry features in the layer. The default name is `the_geom`.

`legend_label` – the name of the layer in the legend.

`search_script` – the name and the location of the search form associated with the layer.

`select_script` – the name and the location of the select form associated with the layer.

`edit_script` – the name and the location of the edit form associated with the layer.

`normal_layer` – the name of the map layer displayed when no highlighting occurs.

`searched_layer`, `checked_layer`, `selected_layer` – the names of the layers used to highlight the geometry features *searched for*, *checked*, or *selected*, respectively. The geometry features returned by a search operation are *searched for*, those whose checkboxes are turned on in the search result form are *selected*, and the geometry feature chosen for editing is *selected*.

`img_src`, `img_width`, `img_height` – the file name, the width, and the height of the icon in the legend.

`onclick` – the name of the javascript event handler activated when the user selects the layer in the legend.

`data_srid` – the *spatial reference identifier (srid)* that designates the map projection used by the geometry features in the map layer. If this srid is different from that of the map region, then geometry features in the layer are reprojected before they are displayed on the map.

### 3.3 Quick View Configuration

The *quick view* mechanism allows the user to select the map area of her interest. That is, the user can switch to a new map area quickly by selecting the map area from the quick-view dropdown list.

Each entry in a *quick view configuration file* describes the name of the map area represented by the entry, the map projection used by the *extent*. The quick view configuration file for the Oregon region, for example, can be as follows:

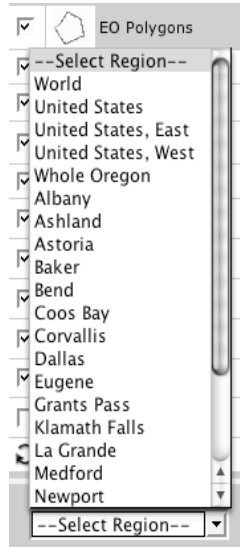
```
$qview = array(
  array(
    'name' => 'World',
    'srid' => 4326,
    'extent' => '-180,-90,180,90'
  ),
  array(
    'name' => 'United States',
    'srid' => 4326,
    'extent' => '-125,13,-65,53'
  ),
  array(
    'name' => 'United States, East',
    'srid' => 4326,
    'extent' => '-102,22,-60,50'
  ),
  array(
    'name' => 'United States, West',
    'srid' => 4326,
    'extent' => '-135,30,-105,50'
  ),
  array(
```

```

    'name' => 'Whole Oregon',
    'srid' => 6010,
    'extent' => '46461.662375,-
43912.968464,2487069.754184,1785176.331408',
    ),
...

```

The quick view list of regions displayed in the map interface according to the configuration file above is shown in Figure 3.



**Figure 3** Quick View Selection of WebGD Application NHIS

Each entry in a quick view configuration file can specify the following options:

`name` — the name of the map area.

`srid` — the spatial reference identifier for the extent explained below.

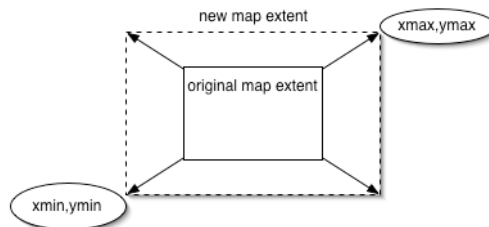
`extent` — the `xmin`, `ymin`, `xmax`, and `ymax` values describing the positions of the lower left and upper right corners of the map area.

### 3.4 Implementation of Operations for Map Navigation and Data Manipulation

We now describe how the map navigation and data manipulation features are implemented in the WebGD Framework.

- **Map Zoom-In**

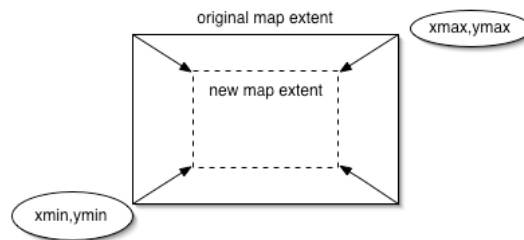
The user can activate a zoom-in operation by clicking on the map or by specifying a rectangular zoom-in area. The mouse can be dragged with its button pressed to define the rectangular area. The location of the mouse click or the extent of the zoom-in area is captured in the screen coordinates by JavaScript code executed on the Web browser and sent to the Web server. Script `gmap75.php` executed by the Web server converts these coordinate values to the map-coordinate values used by the current map region. This extent is then converted to the global coordinate (longitude/latitude) system and is compared with the extents of the regions defined. The smallest region encompassing the new map area is selected as the new region, which may be identical to the old region. The extent of the new map area is then converted with the spatial reference for the new region. Finally, this extent is used to perform the zoom-in operation.



**Figure 4** Map Zoom-In

- **Map Zoom-Out**

The mechanism for this operation is similar to the zoom-in mechanism. The difference is that the new extent of the map area is computed by `gmap75.php` so that the old map area will fit in the rectangular area specified for zoom-out.



**Figure 5** Map Zoom-Out

The algorithm that computes the new map extent when a map zoom-out operation occurs is shown in Figure 6.

```

/* width and height of zoom box in screen coordinates */
rect_image.width = rect_image.maxx - rect_image.minx
rect_image.height = rect_image.maxy - rect_image.miny

/* width and height of map prior to zoom-out, in map
 * coordinates.
 */
map_prev.width = map_prev.maxx - map_prev.minx
map_prev.height = map_prev.maxy - map_prev.miny

/* scale factor for zoom-out */
image_scale_x = map_image.width [in pixel] /
                rect_image.width [in pixel]
image_scale_y = map_image.height [in pixel] /
                rect_image.height [in pixel]

zoom_factor = MIN(image_scale_x, image_scale_y)

/* width and height of map after zoom-out request */
map_after.width = map_prev.width x zoom_factor
map_after.height = map_prev.height x zoom_factor

/* map scale after zoom-out request */
scale = map_after.width / map_image.width

/* new map extent for zoom-out */
map_after.minx = map_before.minx - (rect_image.minx x scale)
map_after.maxx = map_after.minx + map_after.width
map_after.miny = map_before.miny -
                ((map_image.height - rect_image.maxx) x scale)
map_after.maxy = map_after.miny + map_after.height

```

**Figure 6** Algorithm for Map Zoom-Out

- **Map Panning**

The user can perform a map panning operation by pressing the mouse button, dragging the cursor to the new position within the boundary of the map, and releasing the mouse button. The difference between the initial and the new mouse positions are then used as the *vector for translation*.

- **Get Information**

After selecting the `Get Information` operation, the user can click on the map at the location of the target geometry feature. Then the location of the mouse click are captured as screen coordinate values and then converted into the map coordinates for the current map region by JavaScript code executed by the Web browser. These coordinate values are sent to the *select script* provided for the maplayer containing the target geometry feature. If the spatial reference of the current map region is different from that of the maplayer containing the target geometry feature, the map coordinate values of the click location are converted to

the spatial reference of the maplayer. Finally, the point-based spatial query is performed as follows.

1. For *point* or *line* features, a small bounding box is constructed around the click location. Then the features that intersect with this bounding box are retrieved.
2. For *polygon* features, the features that contain the location of the mouse click are retrieved..

The selected features are then highlighted on the map, and the data associated with them are displayed as a table. The user can then select any one of the features in the table for detailed information.

- **Search by Area**

To perform a search for geographical features within an rectangular area, the user can use the **Search by Area** command. The search area can be specified as a rectangular box on the. Conversions of the coordinate values are performed as done for the **Get Information** command. The area-based spatial query is performed as follows.

1. For *point* features, the ones that are completely within the rectangular area of the spatial query are selected.
2. For *line* features, the ones that are within or intersect the rectangular area of the spatial query are selected.
3. For *polygon* features, the ones that are within or that intersect with the rectangular area of the spatial query are selected. *Polygon* features that completely contain the the rectangular area are excluded with option `$search_mode => border`. This option is useful when there are may features with large areas. Without this option, too many features may be selected.

Selected geometry features are highlighted on the map, and the data associated with them are displayed as a table.

- **Geometry Feature Editing on the Map**

The current version of the WebGD framework allows the locations of point features to be updated. The user can *drag* a point feature from an old location to a new location. In the future, we plan to support editing of more complex geometry features such as linestrings and polygons.

## 4 WebGD Application Generator

Several tools have been developed to augment the WebGD framework and simplify application development. The WebGD Web-site generator (WebGD-Gen) can create an entire WebGD application, including a web-based mapping interface. WebGD-Gen *automatically* generates a consistent set of Web scripts from *configuration files*, which are again automatically generated from a relational database schema. Since form generation is automatic, the cost of application development is greatly reduced. For a database such as Biotics that contains approximately 700

tables, programming all the required 3,500 (700 x 5) forms manually can be costly, even infeasible.

WebGD-Gen is implemented as a collection of *templates*. Each template, combined with a corresponding *configuration file*, generates one of the following six types of Web scripts: *search*, *select*, *edit*, *information*, *action*, and *treeview* scripts. Templates and configuration files are written in PHP. The Web scripts generated by them are also in PHP. The generated scripts are executed on a Web server by a PHP interpreter. Each script, except for an action script, creates a Web form that is displayed on a client computer by a Web browser. Figure 2 illustrates the interactions among the Web scripts and forms.

Furthermore, WebGD-Gen can automatically generate the statements for inserting, searching, and deleting *geographical features* if the following parameters are present in the configuration file:

```
$web_gd = 'MULTIPOLYGON';           // type of geographical features
$layer_name = 'grp_eo_py';          // layer group in legend
$geometry_column = 'the_geom';      // geometry column containing shapes
$gid_column = 'gid';                // geographical feature IDs
$db_table_srid = 6010;              // spatial reference
```

Necessary parameters for correct operation of GIS application above are also automatically generated by WebGD-Gen from *geometry column* information in the database.

The forms generated for geographical features can perform the following additional functions compared to those for ordinary database tables:

1. A search form can be activated from a map interface. In this case, the extent of a search box specified on the map is passed as additional search parameters.
2. A select form includes additional JavaScript code for highlighting geographical features retrieved or selected by the user.
3. An edit form can insert a record for a geographical feature, after transforming the coordinate values from the spatial reference used by the current map interface to the one used by the geometry column for the record.

The WebGD framework and WebGD-GEN were developed *incrementally* and *iteratively* during the last four years. We first implemented in 2000 an application that allowed point features to be inserted on a map by using ASP with ArcIMS and ArcSDE. In 2001, we re-implemented this application with ASP.NET, as ASP.NET provides Web controls, which are better building blocks for Web pages. Based on this application, the first version of WebGD framework was created in 2002 in order to support multiple applications (Wangmutitakul et al., 2004).

In early 2003, we re-implemented an application called Motels Oregon with MapServer, PostGIS, and PostgreSQL (Sano et al., 2003). This version on Linux was more reliable and faster than the old one, as well as being built with free software. While implementing the next MapServer application, which was a germplasm resource management system (GEM-GIS), we created the first version of WebGD framework for MapServer. This framework was then enhanced so that it can handle polygon features as well as point features.

The two major enhancements made to the WebGD framework in 2004 were *dynamic switching of spatial references* for different regions and *automatic generation* of Web forms that can be used to insert, query, and delete geographical features. Compared to an application that simply displays geographical features as points on a map, the current WebGD framework is roughly 20 times more complex in terms of the time we spent implementing the required features.

## 5 Conclusion

We have been developing the WebGD framework and the WebGD-GEN application generator for Web-based GIS/database applications. A Web-based GIS application generated by them are unique in the following respects:

1. The geographical features can be inserted, queried, and deleted from the map interface displayed on a standard Web browser.
2. A region sensitive map interface that can be customized easily by a set of configuration files.
3. Web-forms that manage data on geographical features and data in ordinary database tables can be automatically generated.
4. Dynamic switching of spatial references allows an application to cover different regions with different map files, map legends, and quick-view lists. This is an important feature needed for an application that covers the entire USA or the world.

The cost of running our applications is extremely low. We could put copies of such large databases as Biotics, Fishbase, and a part of National Germplasm Resource Information System on a \$800 PC. The software tools we use, such as the University of Minnesota MapServer, PostgreSQL DBMS, PostGIS, Apache, and PHP are all available for free. The GIS data used, such as those from USGS, TIGER/LINE, and Digital Chart of the World (DCW), are also in the public domain. Automatic code generation of a WebGD application will save a great deal of effort in the development of a spatial decision-support system. Although some manual customization is required, the time needed to customize can be lowered to weeks or months compared to the years required to build a *spatial decision-support system* from scratch.

## References

- Aspinall, R. (2000) A framework for use of spatial information in analysis and prediction of land use change. In M. J. Hill, R. J. Aspinall, (Eds.), *Spatial Information for Land Use Management*. Chapter 15 (pp. 205-215). Gordon and Breach.
- Bisby, F.A. 2000. The quiet revolution: biodiversity informatics and the Internet. *Science* 289, 2309-2312
- Dana, Peter H. *Map Projection Overview*, [http://www.colorado.edu/geography/gcraft/notes/mapproj/mapproj\\_f.html](http://www.colorado.edu/geography/gcraft/notes/mapproj/mapproj_f.html).

- DM Solutions Goup Inc. *PHP MapScript*, <http://www.maptools.org>.
- Kingston, R. (October 1998). Web-based GIS for public participation decision making, In *Procs of NCGIA PPGIS Meeting*, Santa Barbara, California. Retrieved May 2003 from <http://www.ncgia.ucsb.edu/varenius/ppgis/papers/kingston/kingston.html>.
- Eum, D. and Minoura, T. (June 2003). Web-based database application generator. *IEICE Transactions on Information and Systems*, Vol. E86-D, No. 6.
- Fogelson, C. (December 2002). Biotics 4.0 data model version 1.0. Retrieved January 5, 2004, from <http://whiteoak.natureserve.org/hdms/HDMS-DataModel.shtml>.
- McKenna, Jeff. *Mapserver PHP/MapScript Class Reference - Versions 3.6, 4.0 & 4.2*, DM Solutions Group Inc.
- NatureServe (February 2002). Element Occurrence Data Standard. Retrieved January 4, 2004, from <http://whiteoak.natureserve.org/eodraft/all.pdf>.
- NatureServe (December 2003). Biotics 4.0 Getting Started Guide. Retrieved January 5, 2004, from <http://whiteoak.natureserve.org/hdms/biotics-learn-more.shtml> (now obsolete).
- Sano, J., Wanalertlak, N., Maki, A., & Minoura, T. (July 2003). Benefits of web-based GIS/database applications. In *Prosc. of 2nd Annual Public Participation GIS Conference*, Portland, Oregon.
- USDA-ARS. Western Regional Plant Introduction Station, USDA - Agricultural Research Service, Pullman, Washington, <http://www.ars-grin.gov/ars/PacWest/Pullman/>.
- Ramsey, Paul. *PostGIS Manual*, Refrations Research Inc.
- University of Minnesota. *MapServer*, <http://mapserver.gis.umn.edu>, 2003.
- Sharma, A. (December 2003). Web-based analysis module for a germplasm collection. Master of Science report, School of Electrical Engineering and Computer Science, Oregon State University.
- Wangmutitakul, P., Li, L., and Minoura, T. User Participatory Web-Based GIS/Database Application. In *Proc. of Geotec Event Conference*, March 2003.
- Wangmutitakul, Paphun, et al. WebGD: Framework for Web-based GIS/database Applications, *Journal of Object Technology* 3, 4, 209-225, 2004.
- Wuttiwat, T., Minoura, T., and Steiner, J. (May 2003). Using Digital Orthographic Aerial Images as User Interfaces. In *Proc. of ASPRS Annual Conference*, Anchorage, Alaska.