Multi-layer Control and Graphical Feature Editing Using Server-Side Rendering on Ajax-GIS

Takeo Sakairi*, Takashi Tamada, Katsuyuki Kamei, and Yukio Goto

Advanced Technology R&D Center, Mitsubishi Electric Corporation
Sakairi.Takeo@db.MitsubishiElectric.co.jp,
{Tamada.Takashi,Kamei.Katsuyuki}@bx.MitsubishiElectric.co.jp,
Goto.Yukio@aj.MitsubishiElectric.co.jp

Abstract. This paper presents the methods of the multi-layer control and the graphical feature editing by the server side rendering on Ajax-GIS. Ajax-GIS uses divided raster image file called "tile" in order to keep light handling. We propose that the multi-layer control is realized by means of merging transparent tiled images in the server application as the requests of the client application. Furthermore we propose the graphical feature editing protocol that sent from a client and send back to an image in order to edit a feature such as moving vertices, changing color. In an evaluation experiment of an actual map data, we confirmed the effectiveness of these methods as compared with conventional methods.

Keywords: Ajax-GIS, Server-Side Rendering, Multi-layer Control, Graphical Feature Editing.

1 Introduction

GIS (Geographic Information System) [1] is a computer system to display digital map. GIS can display a variety of information on a map, and can share the information via Internet. GIS that has these characteristics is available for area marketing, disaster prevention [2], facility management, and so on.

A few years ago, a technology that is called Ajax[1] (Asynchronous JavaScript and XML) [3] appeared. Ajax technology offers a fast-loading and asynchronous updates. A typical example is GoogleMaps [4]. GoogleMaps had applied Ajax technology to web-based GIS (hereinafter referred to as "Ajax-GIS"). Ajax-GIS uses divided raster image data called "tile" in order to keep light handling. Ajax-GIS can display map by assembling tiled raster images which are transferred asynchronous from server using JavaScript program. Ajax-GIS have some issues that are attributable to using raster images. For example, it is difficult to realize a multi-layer control and graphical feature editing by using only web-browser rendering engine.

* Corresponding author.

[1] Asynchronous communication can be achieved by an XMLHttpRequest object. An XMLHttpRequest is an API that can be used by JavaScript.

M. Kurosu (Ed.): Human-Computer Interaction, Part IV, HCII 2013, LNCS 8007, pp. 722–729, 2013.

To realize these functions, we propose to apply to server-side rendering for Ajax-GIS.

2 Ajax-GIS Architecture

In this chapter, we describe the basic Ajax-GIS architecture(Fig.1). Ajax-GIS comprise server side and client side. Ajax-GIS server manages several maps which are residential map, digital road map (DRM) data, disaster information, and so on. These map data are raster image data made from vector maps, and they were divided into tiled images by the rasterizing process on the server side. These raster maps are created in every scale and every layer, and managed by the server, and are used as a background map. Ajax-GIS can display the maps by means of assembling tiled raster images with the JavaScript program on the client side.

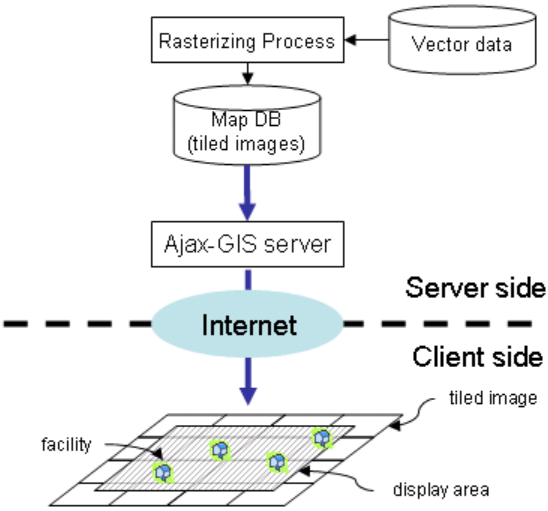

Fig. 1. Ajax-GIS operation flow

A primary characteristic of Ajax-GIS is the increased responsiveness and interactivity of web pages so that entire web pages do not have to be reloaded each time there is a need to fetch data from the server. When a user operates a map, for example scrolling the map, classic Web-GIS can't accept the user's next operation. But if Web-GIS applies Ajax technology, a user can repeat an operation without getting a reply from the server. Fig.2 shows the difference between Classic Web-GIS and Ajax-GIS. In this way, this system offers fast-loading, asynchronous display updates, and smooth map scrolling. Moreover the execution environment on the client side requires only a web-browser without particular plug-in software (e.g. Java Runtime Environment [5], Flash [6]), making this system highly convenient.

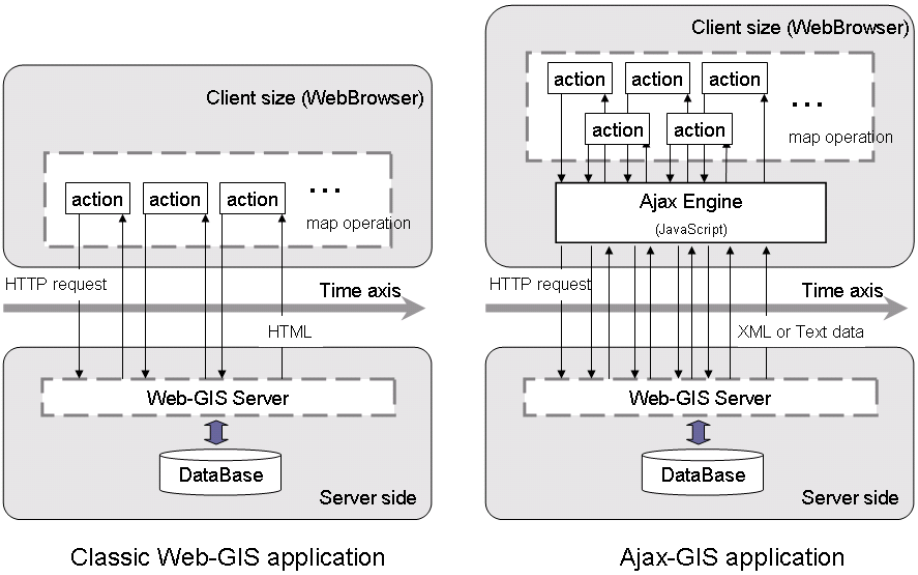

Fig. 2. Classic/Ajax Web-GIS model

3 Multi-Layer Control

In this chapter, we describe multi-layer control for Ajax-GIS. We propose that the multi-layer control is realized by means of merging tiled images in the server application as the requests of the client application on the fly.

3.1 Conventional Method

Typical implementation for Ajax-GIS realizes the multi-layer control by means of overlaying some transparent tiles at the client side (hereinafter referred to as "conventional method1"). With low number of overlaid images, this is simple and efficient. However, if larger quantities of layers are displayed by this way, the html tree that stores the image data will grow large, then the performance of map operation as a map scroll will be slow. As a result, Ajax-GIS client application should draw many tiled images. The sums of images are determined by multiplying the number of tiles by the number of layers. We developed the sample application to examine this conventional method1. As a result, we found that the display time is proportional to the number of layers. In this test, the number of layers is more than five, the map display feel very slow.

On the other hand, there is another conventional method to create image from vector data in real time by server side (hereinafter referred to as "conventional method2"). However, this method has issues that processing load becomes larger and created images are difficult to reuse by reason of client particular operations.

3.2 Proposed Method

In order to solve these issues, we apply the server-side rendering to Ajax-GIS. Fig.3 shows the architecture of multi-layer control by proposed method. At server side, tiled images with transparent background are created from map database every layer and every scale, and these images are stored in tiled image database. At client side, client map are displayed by assembled tiled images.

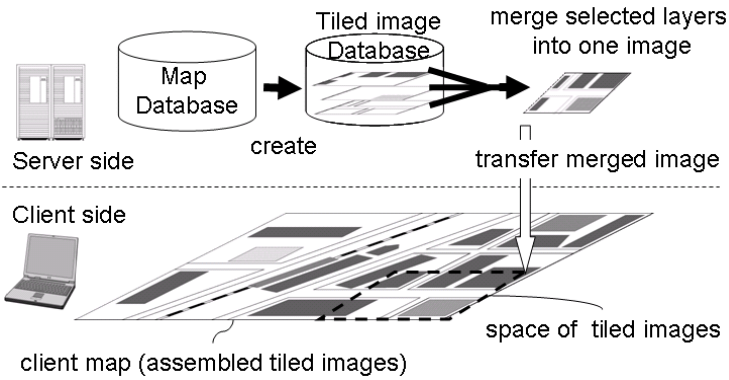

Fig. 3. Multi-layer control by proposed method

Fig.4 shows the multi-layer control process. At client side, client (1) selects display layers. This display layer information is transferred server via Internet using URL parameter. URL parameter contains display layer names. At server side, after received this information, if there is no tiled image that is selected layer in the tiled image database, the server (2) creates tiled images of selected layers. If all layers images are created in advance, this process can skip. Then server (3) merges selected tiled images into a image. Then server (4) returns the image that is overlaid selected layer images to a client. At client side, client application (5) displays merged images.

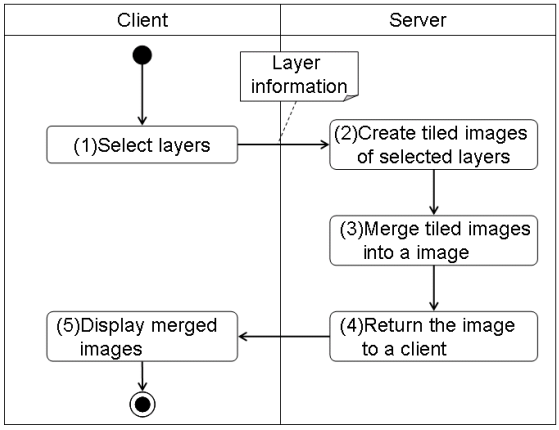

Fig. 4. Multi-layer control process

3.3 Evaluation

We developed a test application, and verified the effectiveness of our proposed method. The evaluation process is done with a JavaScript automatic map navigator. The path which is randomly generated is broken into 100 drag-and-drop steps where after each step (mouse release), queries for layer operations and for images are dispatched. As a result, we found that the proposed method is 1.5 times faster than the conventional method2 (Table.1).

Table 1. Graphical feature editing protocol

	Number of tiles	Total processing time(s)	Average processing time per one tile (ms)	Total data size of tiles (kb)
conventional method2	1160	78.9	68.0	3105.5
proposed method	1190	50.9	42.8	3101.1

4 Feature Drawing Function

In this chapter, we describe the feature drawing function for Ajax-GIS. We propose that the graphical feature editing protocol that sent from a client and send back to an image in order to edit a feature.

4.1 Conventional Method

In conventional method, graphical editing feature was realized by drawing function which is implemented in web-browser. Some rendering engines such as VML (Vector Markup Language)[7], SVG (Scalable Vector Graphics)[8] and Canvas are implemented in a web-browser. However conventional method has a main issue that implementation of drawing functions by JavaScript is dependent on these rendering engines. Therefore, we should implement drawing functions for some web-browsers. As a result, software development costs and maintenance costs will be increased.

4.2 Proposed Method

In order to solve this issue, we apply the server-side rendering. Server-side rendering can centralize drawing operations at a server, so a dependence issue associated with web-browser will be solved. This proposed method has merits which are not to need cross browser compatibility. When a client operates the feature editing such as moving coordinates or changing color, these operations are transferred as a URL parameter such as GET or POST. In order to draw feature, we defined the specific protocol to communicate between client and server. Table.2 shows the some protocol elements.

Table 2. Graphical feature editing protocol

Key	Contents	Overview
cid	Client identification ID	identify uniquely the client
oid	Editing feature ID	identify uniquely the feature
t	Editing feature's coordinate ID	identify uniquely the editing feature's coordinate ID
lc	Line color	line color(e.g. blue, black)
fc	Face color	face color(e.g. gray, white)
w	Line width	line width(e.g. 2point)
ls	Line style	line style(e.g. dotted line)
fs	Face style	face style(e.g. fill)

For example, this protocol is used such as below.

```
http://localhost:80?cid=1555372038&oid=2&op=L&t=n20003&v=
n20003[1067,1388]&i=1193.5,1351.5,136.5,117.5&lc=255,143,
0&fc=255,255,255&w=3&ls=0&fs=0&s=0.5
```

4.3 Geometry Editor Infrastructure

We developed the prototype called "Geometry Editor" that is implemented our proposed method. Geometry Editor is a web-based editor currently supporting polyline and polygon editing through node manipulation. Nodes can be inserted, removed, and moved while an object can be shifted, scaled, and rotated in its entirety. The editor aims to support easy integration with other applications. There are two separate blocks of html code that must be included in the target application, one being

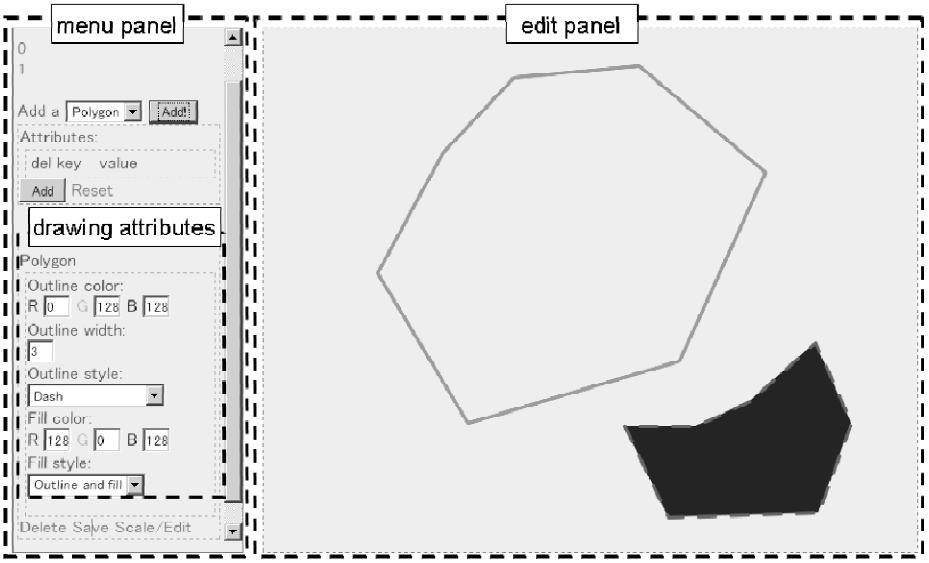

Fig. 5. Geometry Editor Screen

the component defining the editable region, the other being the component defining the property editor.

Fig.5 shows the Geometry Editor screen. The screen consists of two parts: a left side, displaying the menu panel that has some components to edit editing parameters (i.e. outline color, outline width, fill color, and so on), and the edit panel to draw shapes on the right side. When a client draws shapes on the edit panel, the client sends the information written by the editing protocol to the server. After the server receives the information, and then creates an image by means of server side rendering, and returns it to the client. We describe about this process in detail in the following section.

Fig.6 shows the HTML div structure of the editor. The layers related to the editor are contained in the editor layer which is in turn contained in the layer of the application to be integrated with.

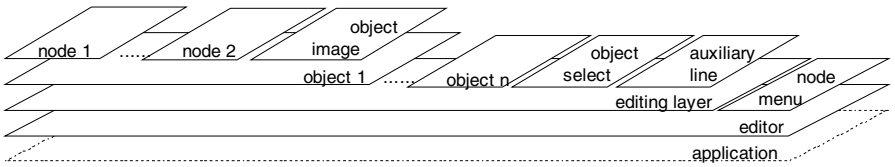

Fig. 6. Geometry Editor HTML Architecture

4.4 Ajax-GIS application

We built the Geometry Editor into out Ajax-GIS. Fig.7 shows the feature editing process between client and server. First, client (1) selects the layer of the editing target, then (2) selects region in order to narrow down the target. Next, client (3)

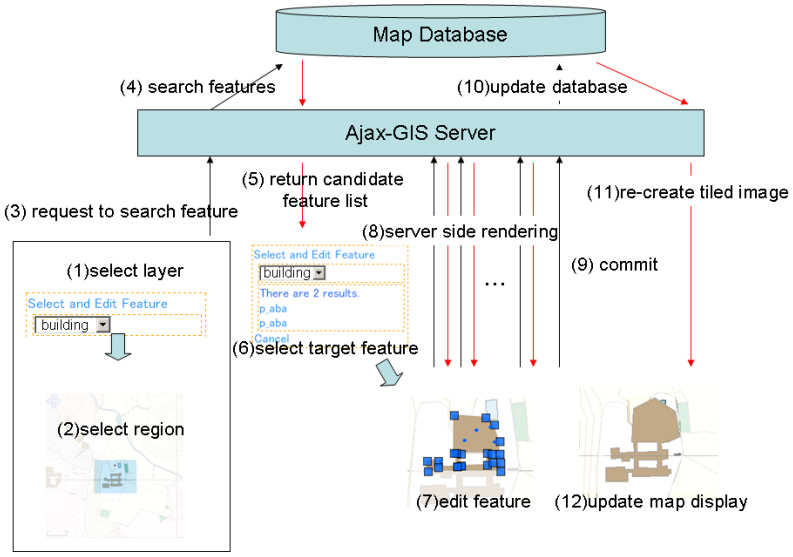

Fig. 7. Feature Editing Process

sends this selection condition to the Ajax-GIS server. After Ajax-GIS receives the selection condition, Ajax-GIS server (4) searches for features from Map database, and then (5) return candidate editing feature list to the client. The client (6) selects a target feature from the list. Then the client can (7) edit a target feature by (8) server side rendering using defined protocol. After finishing editing, the client (9) commits update data to Ajax-GIS server. When Ajax-GIS server receives the update request, Ajax-GIS server (10) updates the Map database. Then Ajax-GIS server (11) re-creates tiled image including this feature, and sends this tiled image to the client. Finally the client (12) updates map display.

For example, a time to generate a polygon fill hundreds of coordinates on the server side was less than 100ms. As a result, the proposed method has been real-time editing feature without user stress.

5 Conclusion and Feature Works

In this paper presents the methods of the multi-layer control and the graphical feature editing by the server side rendering in Ajax-GIS. In the future, we will seek further efficiency and rationalization of our method, and enrich the graphical feature editing protocol. And we will work toward practical application of a real system.

References

1. Aronoff, S.: Geographic information systems: a management perspective, p. 58 (1989)
2. Sakairi, T., Tamada, T., Nakata, H.: GIS Crisis-management System using Ajax Technology. In: SICE Annual Conference 2008, August 20-22, pp. 3043–3046. The University Electro-Communications, Japan (2008)
3. Crane, D., Pascarello, E., James, D.: Ajax In Action. Manning Publications (2005)
4. GoogleMaps, `http://maps.google.co.jp`
5. Java, `http://www.java.com`
6. Flash, `http://www.adobe.com/software/flash/about/`
7. VML, `http://msdn.microsoft.com/en-us/library/bb263898.aspx`
8. SVG, `http://www.w3.org/Graphics/SVG/`