

PLATFORM FOR VISUALIZATION AND WEB-BASED COLLABORATION OF SPATIAL INFORMATION

Satoshi Ueyama

Center for Spatial Information Science, The University of Tokyo
5-1-5, Kashiwanoha, Kashiwa-shi, Chiba 277-8568, Japan
uym@csis.u-tokyo.ac.jp

Yuki Akiyama

Graduate School of Frontier Science, The University of Tokyo
4-6-1, Komaba, Meguro-ku, Tokyo 153-8505, Japan
aki@iis.u-tokyo.ac.jp

Ryosuke Shibasaki

Center for Spatial Information Science, The University of Tokyo
5-1-5, Kashiwanoha, Kashiwa-shi, Chiba 277-8568, Japan
shiba@csis.u-tokyo.ac.jp

KEYWORDS: WoC, Visualization, Internet

ABSTRACT: Today, An enormous amount of spatial information is published and increasing minute by minute. Large-scale spatial information created by Governments or Companies is difficult to process because of its enormous quantity of data. It causes problems like preventing users from reaching to information they requires, difficulty of extracting valuable data and so on. On the other hand, Small-scale spatial information created by individuals has issues like limited or narrow geographic coverage and inconsistency of quality. However, aggregating them, they can be valuable information.

This study will develop a platform for advanced visualization and processing of spatial information, and try to make use of WoC(Wisdom of Crowds) with doing web-based collaboration in large scale. That platform is useful for processing both large-scale spatial information and aggregating of small-scale one.

1. INTRODUCTION

Today, GIS is popularized and widely used in companies and institutes[1]. The advent of GIS enables people to process mass of spatial information in a short time and generates mass of secondary information. On the other hand, since governments and companies get to publish satellite images and surveying data, primary information is also increasing.

This study set two goals for use of increasing spatial information. The first goal is aggregation of small-scale spatial information created by individuals or laboratories. As progress of communication technology, amount of small-scale information is increasing sharply. Small-scale information is expected to contain information mass survey cannot acquire while it has issues like limited or narrow geographic coverage and inconsistency of quality. Aggregation of a lot of small-scale information is required to acquire high quality and wide-area information[2]. The second goal is management of large-scale information created by governments or companies. Nowadays, a lot of valuable information like terrain data or satellite image is published. However, they are often spoiled because they are published through unusable and complex systems. To avoid the situation, a method that enables users to reach information easily is required.

To achieve the goals, an environment where an end-user can develop applications for spatial information is desirable. Currently, one of systems for that purpose is “Goole Maps API” by

Google Inc. Google Maps API is an Application Programming Interface(API) for Google Maps, Web GIS service hosted by the company. Google Maps is characterized by its high portability comes from using only standard features of web browser. However, high portability makes limitation of expressiveness. In spite of Google Inc. publishes “Google Earth”, GIS with higher expressiveness, Google Earth does not have API like Google Maps API. Therefore, in this study, we define the API that is flexible and supports expressive GIS. Then, we implement the API in a sample application.

2. API FOR SPATIAL INFORMATION

This chapter defines API described in Chapter 1. The API is consisted of three components -- Object Model, Style Sheet, Tile URI. We discuss concepts of the API in section 1, and then explain each a component in following sections.

2.1 Concepts

The API is based on URI, CSS, and DOM, which are technologies used on The World Wide Web(WWW). URI is a string ID assigned to every resource. Using URI is essential for developing systems fitted in WWW. DOM is an interface for accessing XML or HTML documents and its elements. CSS is a language for style sheet, which specifies appearance of displayed documents. DOM and CSS are often manipulated by scripts attached to documents, modify documents dynamically acceding to user's demand.

The API is an application based on aforementioned technologies, to realize "Programmable Earth", an environment that enables users to manipulate spatial information easily.

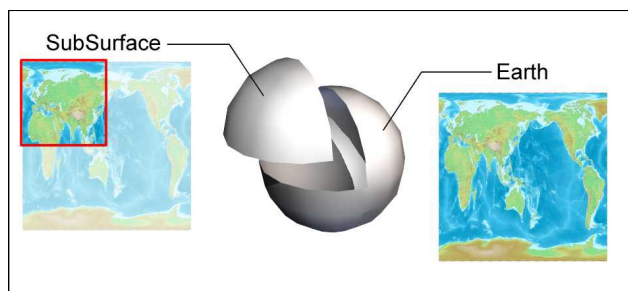


Figure 2-1: Earth and SubSurface Object

2.2 Object Model

In the API, we think of "Earth" as "Document" in XML or HTML documents. Therefore Earth is the top-level object. Earth, represents whole globe, is divided into "SubSurface"s, represents a part of the globe. SubSurface divides parent object into four parts. SubSurface is divided into SubSurfaces recursively. Such an algorithm is known as "quad tree", typical method for structuring 2D space. Figure 2-1 shows relation between

Earth and Subsurface. We extracted functions that can be used from externals from Earth and SubSurface as interfaces. IEarth is an interface extracted from Earth and ISubSurface is from SubSurface. Earth and SubSurface have a common function that holds SubSurfeces devide themselves as child elements. We extracted that function as ISubSurfaceContainer interface. Table 2-1, Table 2-2 and Table 2-3 shows methods and properties of IEarth, ISubSurface and ISubSurfaceContainer.

Table 2-1: Method and Properties of IEarth

Name	Description
addStyleRule	adds specifeid style-sheet rule with specified selector
createSurface	creates a new surface of specified type
getSurfaceByTilepath	retrieves a surface object that has specified tile-path
tileLoader	retrieves an object that provides texture images
transformPoint	transforms specified point(x, y, z)in world coordinate into screen coord
XYtoLatLng	translates (x,y) coord on a projected map into (latitude, longitude)

Note: "world coordinate" is a coordinate system where the origin is at center of the earth, the radius of the earth is 1.


```

}
overlay
{
  altitude: 400m;
}

```

Figure 2-3: Example of The Style Sheet Language

As shown in Figure 2-3, Our style sheet language is designed to resemble CSS. However, containing its own extended notation, our style sheet language is NOT a subset or an equivalent of CSS.

2.4 Tile URI

We defined Tile URI, URI assigned to each a SubSurface. Tile URI has three components -- scheme, tilelib and tilepath. Scheme is fixed to the string "tile:". Tilelib is a namespace for tilepath following tilelib. For experimental use, simple tilelib like "mlit1984" shown in Figure 2-4 can be used. For practical use, domain name like "mlit1984.csis.u-tokyo.ac.jp" is desirable to avoid collision. Table 2-4 shows components of Tile URI, and Figure 2-4 shows an example of Tile URI.

Table 2-4: Components of Tile URI

Name	Description	Example
scheme	Fixed to "tile:"	tile:
tilelib	Namespace for tilepath	mlit1984
tilepath	Unique path of an area on the earth	trssqqrtrsqttt

```

tile:mlit1984/trssqqrtrsqttt

```

Figure 2-4: An Example of Tile URI

Then we discuss tilepath in detail. Figure 2-5 shows SubSurface division. In Figure 2-5, top-level SubSurface is divided into four children. Child on northwest named a1, on northeast named b1, on southeast c1, southwest named d1. Second-level SubSurfaces are named in the same way, a2, b2, c2, d2. Following lower levels are also in the same way. Now a character is assigned to each of ai, bi, ci, di respectively for all i shown in definition (1). Concatenating characters from top-level to lower level to determine certain element uniquely. a3 under c2 under c1 is notated as "CCA". Now length of the string represents i, the nest level. Characters assigned to ai, bi, ci, di can be whatever they are distinct each other. We use q, r, s, t shown in definition (2). This is in the same way as Google Maps, popular Web GIS system.

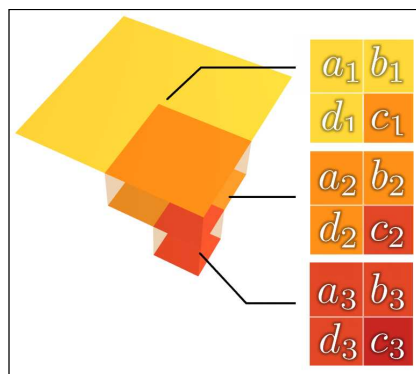


Figure 2-5: Image of SubSurface division

$$a_i := A, a_i := B, c_i := C, d_i := D | \forall i \dots (1)$$

$$a_i := q, a_i := r, c_i := s, d_i := t | \forall i \dots (2)$$

3. TEST IMPLEMENTATION



Figure 3-1: Screenshot of The Initial Screen

We developed a sample application as an example of the API implementation. The sample application is implemented with Java. Figure 3-1 shows initial screen of the sample application. This chapter evaluates functions of the API using the sample application.

3.1 Object Model

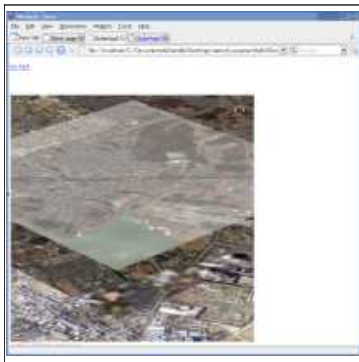


Figure 3-2: Object Manipulation

Manipulating objects, we need some program calling the API. Then we used Javascript bridge and called the API from Javascript engine built in web browser. Figure 3-2 shows a result of object manipulation running the script shown in Figure 3-3. The script shown in Figure 3-3 creates a new Overlay and appends it as a child of a SubSurface. As shown in Figure 3-2, the sample application displays created Overlay.

```
var earth = new Earth();
var overlay = earth.createSurface("overlay");

var t = earth.getSurfaceByTilepath("trssqqrtrsqqttt");
if (t != null)
    t.appendChild( overlay );
```

Figure 3-3 Test Code in Javascript

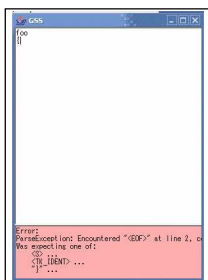


Figure 3-4: Style Sheet Editor

3.2 Style Sheet

The sample application has an input window for style sheet. Style sheet language input into that window is evaluated immediately and results are displayed in the main window. Figure 3-4 shows style sheet editor built in that input window. As shown in Figure 3-4, errors are displayed in bottom part of the window due to unacceptable input. The sample application implements several properties and is able to apply to rendering. Table 3-1 shows properties implemented by the sample application.

Table 3-1 Style Sheet Properties

Property	Description
altitude	Indicates altitude of the tile.
tilelib	Indicates source of images drawn on the tile, equivalent of the tilelib component of Tile URI.
tilepath	Indicates path of an image drawn on the tile, equivalent of the tilepath component of Tile URI.



Figure 3-5 Use of altitude Property

Figure 3-5 shows an example of use of altitude property, Figure 3-6 shows one of tilelib property, and Figure 3-7 shows one of tilepath property. In Figure 3-5, style sheet specifies altitude of the overlay surface to 900m. As shown in Figure 3-5, the main window displays the overlay surface at specified altitude. In Figure 3-6, texture source of tiles that have tilepath matching to regular expression pattern `^trssqqttrrsqqttd.?s` is changed. As shown in Figure 3-6, some textures are replaced with old pictures taken in 1989. In Figure 3-7, style sheet changes textures of all of tiles.



Figure 3-6: Use of tilelib Property

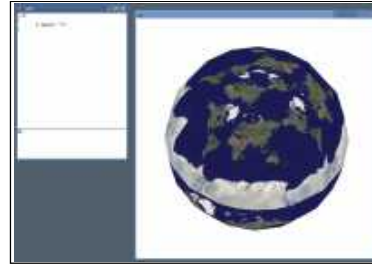


Figure 3-7: Use of tilepath Property

4. CONCLUSION

This chapter describes results and future works of this study as conclusion.

4.1 Results

In this study, we defined API for Spatial Information. Then we showed the API enables users to develop applications for spatial information easily. Since the API provides functions with brief expression, even user who has written tiny scripts should learn readily. Making use of the API, people handle spatial information would make them more valuable.

4.2 Future Works

Event handling system that detects inputs of a user or changes of data is required for developing interactive application. Example of events is "certain point is in viewport", "certain point is out from viewport", "certain area is zoomed in", "certain area is zoomed out", and so on. Our future work is to enumerate enough types of events to develop interactive application and improve the API to handle them seamlessly.

References

- [1] Wanglin YAN. (2003). GIS の原理と応用: Union of Japanese Scientists and Engineers.
- [2] Surowiecki, J. (2004). The Wisdom of the Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations: Doubleday.