Computer Aided Drafting, Design and Manufacturing
Volume 24, Number 2, June 2014, Page 23

CADDM

# Editing Vector Graphics in the Virtual Simulation Environment

*SHI Xiao-long, LAI Shun-nan*

Shenzhen Graduate School, Beijing Engineering Technology Research Center of Virtual Simulation and Visualization, School of Electronics Engineering and Computer Science, Peking University, China.

**Abstract:** We present a method to represent multi-resolution vector graphics such as road networks or railway networks in virtual environment. These vector data can be interactively edited and the landscape and be explored in real time at any altitude from flight view to car view. We design a context-focused vector description of linear can areal features, with associated customized definition painter to specify their appearance (color and material) and their display mode (detailed mode or simplified mode). There are some special problems in drawing vector graphics in virtual environment. Floating-point round-off error appear when we use a low view point to observe the scene, and it leads to scene jittering. Drawing 3D wide lines turns into a problem on 3D terrain. We design a view-based self-adaptive interpolation algorithm and an offset line generating algorithm to solve it. Our results show high performance with good visual quality.

**Key words:** vector graphics; virtual reality; virtual globe

## 1 Introduction

Vector graphics plays a significant role in global virtual environment. Vector data such as roads, railway, lakes, fields could be considered as part of the ground. Vector graphics has important applications in the analysis and management of virtual landscapes.

A 3D vector editing tool is useful in urban planning and terrain fields editing. There are many 2D vector graphics editing tools, such as JOSM, QGIS and Merkaartor, but these are not intuitive. Vector data usually has semantic information and it can be interpreted into landscapes in virtual environment. Using a 3D vector editing tool to interact with virtual simulation environment is an intuitive idea. However, 2D editing tool cannot be transplanted into 3D virtual world directly. Global virtual environment is different from a traditional virtual environment. It contains large scale data and the value of the data is very big. So designing a good strategy to explore the same scene and removing the scene jittering phenomenon are necessary to have a fine visual quality. The surface of the globe is not flat and the elevation data is changing when exploring from global view to near view point, results that drawing vector primitives on it is a challenge. Traditional vector data storage formats are not suitable for data transfer between server and client. We make some improvements on the traditions formats to suit our needs.

## 2 Related Work

Virtual globes are known for their ability to render massive real-world terrain, imagery, and vector datasets. The servers providing data to virtual globes such as Google Earth and NASA World Wind host datasets are measured in the terabytes. In fact, in 2006, approximately 70 terabytes of compressed imagery were stored in Bigtable to serve Google Earth and Google Maps[1]. No doubt, that number is significantly higher today.

### 2.1 Height Maps

Height maps are the most widely used terrain representation. A height map can be thought of as a grey-scale image where the intensity of each pixel represents the height at that position. Typically, black indicates the minimum height and white indicates the maximum height. DEM data is usually represented in

**Project Item:** Supported by National Basic Research Program of China (973 Program, No. 2010CB328002), National Natural Science Foundation of China (Nos. 61232014, 61121002, 61173080) and National Key Technology Research and Development Program of China (No. 2013BAK03B07).

**Corresponding author:** SHI Xiao-long, Male, **E-mail:** shixiaolong@pku.edu.cn

this format.

### 2.2 Rendering Vector Data on Terrain

There are 3 general approaches how to display vector data together with surface models.

1) The most common approach is to render vector graphics into an image and to project it on the terrain as texture with multi-texturing. The texture can be created using rasterizing algorithms such as flood fill or scan-line fill. In the simplest case, a single on-channel texture can be used, where a value of zero indicates the Texel outside of a polygon and a value greater than one representing the interior of a polygon and its opacity. When the globe is rendered, the color of its base texture is blended with the polygon's color based on the polygon map. If multiple polygon colors are required, the polygon map can be expanded to four channels or multiple textures can be used. Multiple textures are also useful for supporting overlapping polygons.

A major benefit of this approach is that vector graphics automatically conform to the globe, even if the globe is rendered with terrain. Also, performance is independent of the number of polygons or number of points defining each primitive. Instead, performance is affected by the size and resolution of the polygon map − which leads to its weakness. Using a low resolution image will lead to aliasing when at a low view point. Using a higher-resolution map will, of course, fix the problem at the cost of using a significant amount of memory.

2) The second approach is based on shadow volumes[2, 3]. A triangulated polygon is raised above terrain, duplicated, and then lowered below terrain, forming the caps of a closed volume encompassing the terrain. Terrain intersecting the volume is shaded using shadow volume rendering. This approach has several advantages: good visual effect and independent from LOD of terrain. But the algorithm requires the depth buffer of terrain and multiple render passes. It is not suitable in too large scene.

3) The third approach is a geometry based mapping, to adapt the vector data to terrain surface and to render them as separate geometric primitives.

Agrawal[4] develops a mapping method based on the technologies of a view-dependent dynamic block-based LOD mesh simplification scheme and out-of-core management of large terrain data by using real-world terrain raster and vector data sets. Sun[5] provides a geometry-based approach in interactive manipulation mode for mapping vector and DEM.

### 2.3 Semantics-based Visualization

One approach to parameterize visualization of model contents is a semantics-based image abstraction[6]. To apply a visualization approach to model contents, CityGML[7] introduced a semantics driven classification and exchange format that has been standardized by the OGC and is accepted by a growing number of GIS software vendors. In the system presented here, semantic information is derived from material and texture information, or defined explicitly at run-time to enable a customized parameterization of visual attributes. Brewer[8] proposed conventions for using colors in cartography-oriented design. The system presented here uses qualitative color schemes to represent entity types of physical model.

The OpenStreetMap[9] project is a knowledge collective that provides user-generated street maps. OSM follows the peer production model that created Wikipedia; its aim is to create a set of map data that's free to use, editable, and licensed under new copyright schemes. There's also a growing community of software developers who develop software tools to make OSM data available for further use across different application domains, software platforms, and hardware devices. The OSM project's hub is the main OSM Web site. Our work is inspired by them and we expand it to virtual environment.

## 3 Data Model and Storage Format

In traditional 2D GIS such as the Esri shapefile[10], the map data is represented mainly in three different ways: a point is a single location in space defined by coordinates, a linestring is an array of points representing a road, a river or a border, and a polygon is an enclosed area representing a field or a lake. There are also some package data structures such as multi-point and multi-patch packaging previous three kinds of shapes as a whole. The descriptive information of the vector data is usually attached as an afterthought in a secondary database. The data structure is complex. It cannot contain all the information(topology and attributes) in a single file. In our data model, these concepts are distilled down into **Point3D**, **Polyline3D**, and **FeatureCollection** with tags being a way to describe each feature. This chapter discusses these core concepts of data model in more detail. The Elements (also data primitives) below are the three basic components of our data model of the physical world.

1) Point3D

Points on Earth are called **Point3Ds** and are represented by a three-dimensional coordinates(X, Y, Z). For example, **Point3D**s are used to represent shops, bus stops, benches, and post boxes. A **Point3D** without any tags will always be a sub-element of another element.

2) Polyline3D

An ordered list of **Point3D**s is called a **Polyline3D**. A **Polyline3D** has a maximum of n **Point3D**s to ensure that tools and users are not overwhelmed with very large structures that are difficult to manipulate. The n is generally given as 2000. They are used for representing linear features like roads, railways, rivers and pipes.

Areas do not have a specific data type, and are simply a kind of closed **Polyline3D** where the first point is the same as the last point. They are used to represent building outlines, lakes, and forests. A closed **Polyline3D** can be interpreted as a closed polyline or an area. We use a tag area = yes/no to distinguish them.

3) FeatureCollection

**FeatureCollection**s are ordered lists of **Point3D**s, **Polyline3D**s or **FeatureCollecitons**. Each member of a **FeatureCollection** has an optional role that gives an additional piece of information about the sub-element. **FeatureCollection**s can represent road or bicycle routes, turn restrictions, and administrative boundaries. It can also include a multi-polygon that describes an area (whose boundary is the 'outer polylines') with holes (the 'inner polylines').

The **FeatureCollection**s are not categories and should not be used solely to group things together. Some additional information should to be grouped together with them.

Then the concepts described below are additional member in the basic data primitives.

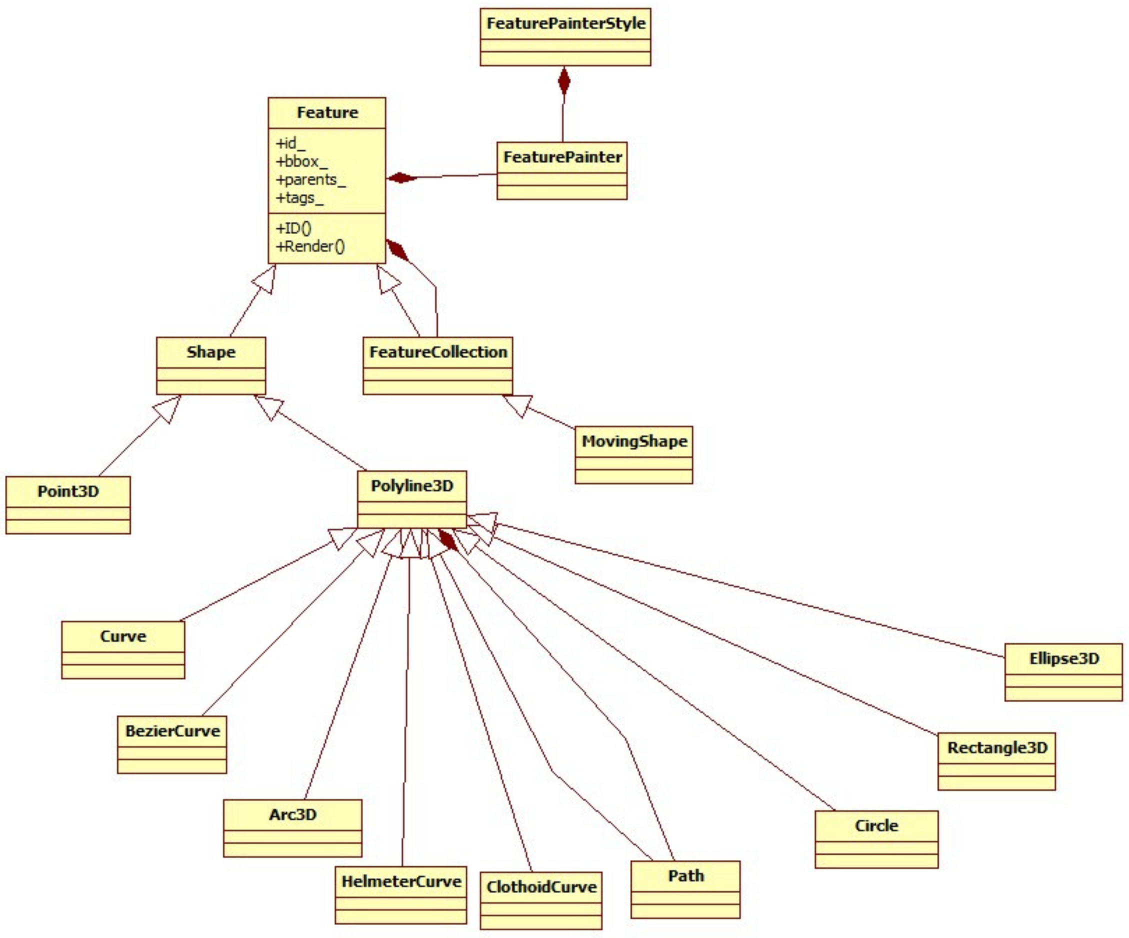

**Fig.1. Implementation of data model.**

4) Tag

Tag is "key-value" pair of strings optionally attached to each element in our data model. These tags describe the feature they are attached to, and can be any pair of strings with the only restriction that keys be unique inside one element. If there are no tags associated with a feature, most renderings of the data won't display that feature. Attached Tag is also a semantic part of the primitive. There is no fixed dictionary of tags, and everyone can define your own dictionary.

5) Identifier

Any element in our vector dataset, of any of the three basic primitives above, is identified by a unique numerical id. These numbers have no other purpose than to allow referencing of individual features, and have no special meaning. A **FeatureColleciton** or a **Polyline3D** uses these identifiers to reference its sub-elements. Two **Polyline3D**s are said to meet only if they reference the same **Point3D** identifier, rather than two **Point3D**s with identical coordinates. Closed **Polyline3D**s representing areas need to reference the same **Point3D** ID twice.

The implementation of our data model is shown in Fig.1, and it is shown that the FeaturePainter is an important class to interpret the tags defined in feature. The view based multi-resolution scene exploration method is relied on defining new tag and interpreting the tags based on view point. The classes derived from polyline3D are diffierent curve generating methods.

Our data files are distributed in a JSON format representing the **Point3D**, **Polyline3D**, and **FeatureCollection** concepts. The data model can be represented in key-value pairs directly. We use the MongoDB to store the spatial features and provide CRUD(Create, Read, Update, Delete) Restful web services.

## 4 Template-based Primitives Representation and Generation

Some vector graphics like roads and railways consist of repeat pattern. Storing them into 3D model is a waste of space and it will be a network burden. The method cannot be applied to 3D primitive dynamic generation. In traditional 2D GIS application, there are a couple of styles to display the lines or areas. Our data model can make user-defined tags. We define some primitives semantics tags included the repeated pattern.

Generating wide line is challenging. The earth is a sphere and elevation is different everywhere. To draw a line down to terrain is difficult. There are three ways to achieve the end: geometry method, shadow method and texture synthesis. The second method is not suitable to large scale scene and it requires depth information of terrain. The pre-computation using in the third method cost too much time. The first method is the method we use in our method.

The elevation of same coordinate is changing when exploring among different view point. The DEM (Digital Elevation Model) data is organized in quad-tree data structure in our virtual system ViWo. Considering these factors, the subdivision parameter must dynamic change based on view point.. We create a primitive generation method. The process is shown in Fig.2.

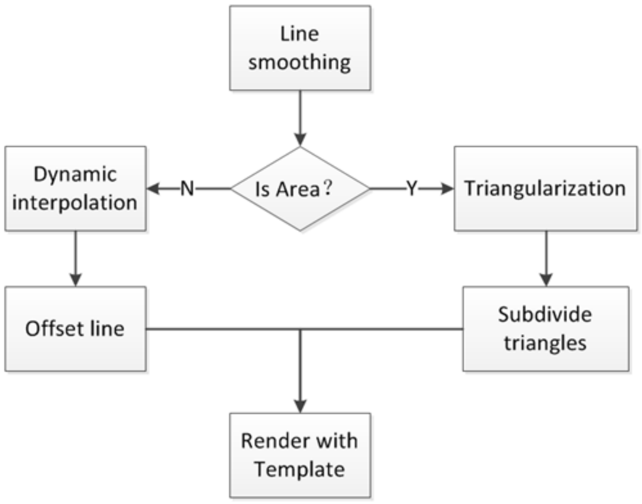

**Fig.2. Primitive generation process.**

**Dynamic interpolation algorithm**: While drawing vector data in 3D environment, elevation value is required for each point and polyline vector needs to match with the terrain mesh, or problems such as "going through the earth" or "suspension in the air" may appear and affect the display effect. To weaken these effects, vector data needs to be interpolated according to the terrain. There are two ways available.

One is to calculate the intersections of spatial line and the 3D terrain mesh directly. Then move the line upper or lower and do it again. At last, join the intersections and it will fit the terrain mesh. However, this method has high complexity and low efficiency in computing. The terrain mesh information is needed in this method. In our system, the terrain mesh information is not provided. We can just get the height of selected point. The other way is to project the spatial line and terrain mesh to the ground and it will be equidistant grids without considering elevation. This method cannot generate a perfect result but it can generate an acceptable result to our system. The sample rate is a key parameter to decide the number of points in the final polyline. For this model, we propose a dynamic and practical interpolation algorithm to reduce the number of points which is described as follows.

**First step:** according to the view point, distance of two control points($D_{2c}$) the current DEM subdivision level, we choose a proper sampling parameter $n$.

$$n = \frac{D_{2c}}{2\_PIXEL\_DISTANCE}$$

where, 2_*PIXEL_DISTANCE* is a spatial distance of two points away from each other 2 pixels in screen space.

**Second step**: use the line function to calculate the points among two control points and scale the points up to terrain surface.

If using this method alone, the sampled points between two control points are too many. We mainly used two strategies to reduce points. First if 2 control points are in viewport, the interpolation algorithm will process. Second, if only points reflecting significant feature are added to final rendering line. The importance of a point $P_i$ is measured by the degree between $P_iP_{i-1}$and $P_{i-1}P_{i-2}$ as Fig.3 shows. $P_{i-1}$ is the previous added point. If the degree $\alpha$ is greater than 5°, the point will add to rendering line.

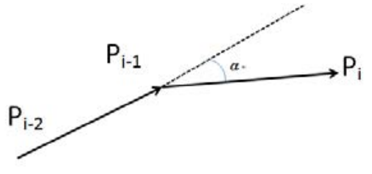

**Fig.3. Dynamic interpolation algorithm.**

## 5 View-related Multi-resolution Rendering

Rendering vector graphics in width line with pattern is suitable for observing virtual scene at near view point. However, when observing the scene at a very high view point, the rendering method is not proper. The computation time is too much. Additional, people are concerned about different aspects of the scene at different height. When at very high view, they usually focus on overall of whole world and not care about the detailed information. When they in a low view point, they generally want to see more detail of the real scene. The simplified format is not enough. We define tag restrict the feature painter. There is more than one feature painter for a feature. The painter choose appropriate feature to draw the feature.

We define the display minimum level tag and display maximum tag to let the painter choose display mode based on the quad-tree subdivision level.

**Local coordinate system**: When looking at the virtual scene in a near view point, the scene jitters irregularly. The phenomenon is caused by the computation accuracy. The earth's radius is too large for 32 bit double to represent. Floating-Point round-off error can be eliminated by using local coordinate system when in low view point. We setup several local coordinate systems in large cities in China to solve the problem.

## 6 Results

### 6.1 View-related Multi-resolution Rendering

Fig.4 shows three scenes. The height of view point is decreasing from left to right. At a high view, the railways are drawn in polylines simply with color to show an overview of the whole scene. In middle distance, lines are drawn in special GIS representation formats. In low view point, the railways are drawn with detailed information.

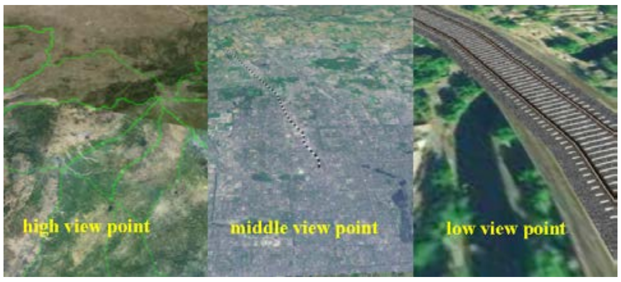

**Fig.4. View-related multi-resolution rendering.**

### 6.2 Drawing Simple Lines and a Wide Line on Terrain

Fig.5 shows the effect of our algorithm. It attaches with terrain and has a fine visual effect, in some sharp corner of edge, the results generated by the algorithm is not good enough. With the dynamic interpolation algorithm, the number of points interpolating in the final polylines are reduced largely. The compare results of normal method and our method are shown in Table 1.

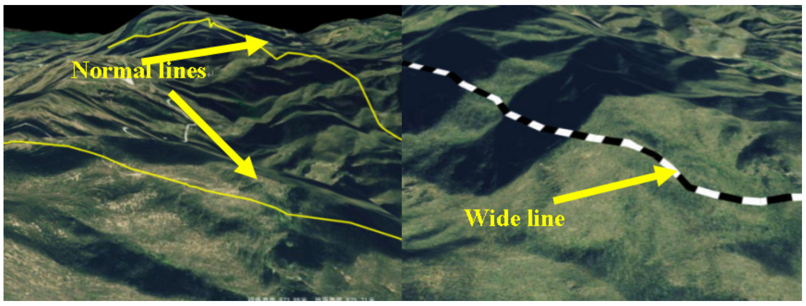

**Fig.5. Drawing lines on terrain.**

**Table 1. The Compare Results of Dynamic Interpolation Algorithm and Normal Method**

| Number of control points | Using our method | Fixed sampling rate method |
|---|---|---|
| 12 | 620 | 4080 |
| 18 | 69 | 307 |
| 20 | 2116 | 9737 |
| 192 | 471 | 707 |
| 225 | 625 | 1758 |

### 6.3 Using Tool to Design City

We use the tool as a city planning tool. In the future, we will use these generated information to generate 3D city model which is shown in Fig.6 and Fig.7.

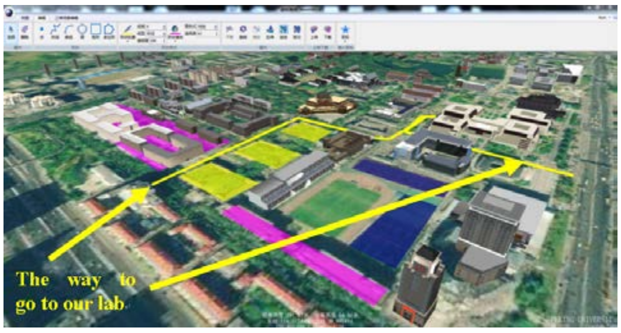

**Fig.6. Using the tool to design school and direct the way.**

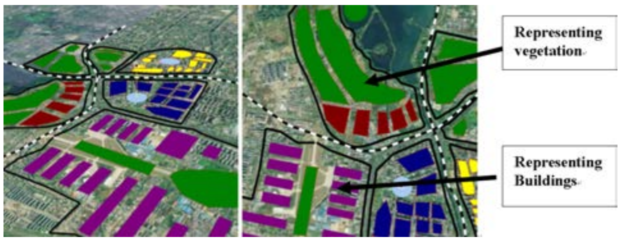

**Fig.7. Using the tool to design city.**

## 7 Conclusions

We design a data model to present vector graphics. The data model can include customized attributes. Presentation of template based vector data and visualization of view-related multi-resolution ones can be implemented using this data model. We use this data model to implement a tool designing the urban. We use adaptive subdivision algorithm to let the line down to terrain surface with less points. We use semantics-based visualization strategies to ensure the performance of the scene exploration. In low view point, scene jittering phenomenon happens, we use local coordinate system to make coordinate value smaller to eliminate it.

## References

[1] Chang F, Dean J, Ghemawat S, et al. Bigtable: A distributed storage system for structured data [J]. *ACM Transactions on Computer Systems (TOCS)*, 2008, 26(2): 4.

[2] Dai C. Rendering 3D vector data using the theory of stencil shadow volumes. The International Archives of the Photogrammetry [J]. *Remote Sensing and Spatial Information Sciences*, 2008, 37: 643-647.

[3] Schneider M. Efficient and accurate rendering of vector data on virtual landscapes, 2007.

[4] Haklay M，Weber P. Openstreetmap: User-generated street maps [J]. *Pervasive Computing, IEEE*, 2008, 7(4): 12-18.

[5] Sun M. Large-scale vector data displaying for interactive manipulation in 3D landscape map [J]. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2008, 37(7): 507-511.

[6] Yang M. Semantics-driven portrait cartoon stylization. Image Processing (ICIP) [C]// *2010 17th IEEE International Conference on, IEEE*, 2010.

[7] Kolbe T H. *Representing and Exchanging 3D City Models with CityGML* [M]. 3D geo-information sciences, Springer, 2009 : 15-31.

[8] Brewer C A. Color use guidelines for mapping and visualization [J]. *Visualization in modern cartography*, 1994, 2: 123-148.

[9] Agrawal A. Geometry-based mapping and rendering of vector data over LOD phototextured 3D terrain models.

2006.
[10] ESRI. Shapefile technical description. An ESRI White Paper, 1998.

**SHI Xiao-long** is currently a senior student majoring in CS. His research interests include virtual reality, computer graphics, visualization.

**LAI Shun-nan** is an engineer of School of Electronic Engineering and Computer Science, Peking University, She received Master's degree from Dept. of Control Enginnering, Huazhong University of Science and Technology. Her interests include CAD and sensor technology.

## 参考文献(10条)

1. Chang F;Dean J;Ghemawat S Bigtable:A distributed storage system for structured data 2008(02)

2. Dai C Rendering 3D vector data using the theory of stencil shadow volumes. The International Archives of the Photogrammetry 2008

3. Schneider M Efficient and accurate rendering of vector data on virtual landscapes 2007

4. Haklay M Weber P Openstreetmap:User-generated street maps 2008(04)

5. Sun M Large-scale vector data displaying for interactive manipulation in 3D landscape map 2008(07)

6. Yang M Semantics-driven portrait cartoon stylization. Image Processing(ICIP) 2010

7. Kolbe T H Representing and Exchanging 3D City Models with CityGML 2009

8. Brewer C A Color use guidelines for mapping and visualization 1994

9. Agrawal A Geometry-based mapping and rendering of vector data over LOD phototextured 3D terrain models 2006

10. ESRI Shapefile technical description 1998