Ginis Web 3D Modeler – A Framework for 3D Terrain Visualization on Web

Dejan Rančić, Dušan Dačić
CG&GIS Lab, Faculty of Electronic Engineering, University of Niš
Aleksandra Medvedeva 14, 18000 Niš, Serbia and Montenegro
{ranca, dusandacic}@elfak.ni.ac.yu

SUMMARY

This paper presents a framework that integrates technologies and standards applied to the design of three-dimensional GIS systems on the Internet. Constructing such a framework has enabled us to design numerous prototypes and test different concepts of 3D GIS visualization, reusing existing standard systems. Described X3D descriptive language is chosen as a suitable platform for 3D GIS system development. GINIS Web 3D Modeler, three-dimensional terrain visualization framework on the Web, is described as a way of integrating already existing and commercial GIS systems developed at CG&GIS Lab, into a 3D Web-enabled GIS Service. All components of the framework (on both the server and the client side) are described. Possibilities of behavioral programming using Java and JavaScript are also investigated. The implemented level of detail algorithm is, also, presented as well as the measured performances.

KEYWORDS: *Terrain visualization, Virtual reality, X3D, Web*

INTRODUCTION

The complexity of modern Geographical Informational Systems (GIS), and tasks they are meant to perform, demands integration of three-dimensional spatial information, semantics, and all the necessary tools to work with and analyze relationships between them. Regional to national 3D landscape models bear a tremendous potential for illustrating spatial phenomenon to professionals and to the general public alike (Nebiker, 2002). Currently, existing systems lack either three-dimensional analysis capabilities, as do the majority of commercial GIS systems, or are missing tools for analysis of information related to spatial objects (Zlatanova, 2002). There are still only a few systems that have the capabilities to process 3D spatial queries that seem simple enough: "mark all landscape areas that are not covered by radar network" (Rančić, 2003a), or "mark all available apartments in Central Street having windows NOT looking at the newest discotheque". For a while it's justifiable to develop systems with above described capabilities. Remote access to these systems over the Internet is a challenge yet to meet. There are three basic problems to be solved when designing this functionality: data structuring on the server side, ways to organize communication and queries between client and server side, and query result visualization on the client computer (Stoter, 2003). World Wide Web (WWW) is based on standard data formats, such as HTML and XML, JPEG and GIF image formats, and GML for 2D vector graphics images. These standards enable the use of "thin" client-server model in heterogeneous environment that the Web is.

Client is considered "thick" or "fat", if the main GIS functionality and the data rendering are client-side hosted. Consequently the server in this specific system would be called "thin". The server is called "thick" if GIS functionality and pre-rendering is hosted server-side. Within this system, the client would be called "thin". Most appropriate way in designing our system was the use of a "medium" client-server model, where 3D rendering process is on the client-side and done by a third-party standardized plug-in. Altmaier and Kolbe (Altmaier, 2003) even exclude rendering for

interactive 3D worlds on the server since real-time navigation in static images would not be possible anymore.

By copying and concentrating all the data on one monolithic system, we are making this data harder to maintain and update, diverting our concentration from issues that are more important for successful development and deployment of 3D visualization. Therefore, we decided to utilize our existing and proven web services as providers of feature data needed for 3D visualization. By complying with WWW standards, it's easy to develop Web-based applications without concern about particularities of client environment.

Decision of utilizing standard web services led us to another standard, this time for 3D visualization. The use of X3D (Extensible 3D), formerly known as Virtual Reality Modeling Language (VRML), in development of 3D client-server GIS systems is not an unknown idea. VRML exists since year 1994 (Carey, 1997), and it is no longer considered a new and immature standard, but until recently the widespread use of this standard was limited by narrow bandwidths, and also by the cost of powerful 3D graphics subsystems that could enable real-time rendering of X3D (VRML) three-dimensional objects (Walsh, 2001). With lower prices of graphics hardware, more powerful hardware used for data compression, and constant increase of average bandwidth, we now have the possibility to construct 3D GIS systems for wide use, lowering the price for end-users, and thus increasing user population and justifying development effort.

The goal of this paper is to present these technologies and standards that we used in setting direction for the development of interoperable three-dimensional GIS systems on the Internet. We will show that our framework design represents a suitable platform for the design of 3D GIS systems on the Web that have very low development cost, multi-platform support, that are very easy to maintain and expand, and intrinsically enable easy integration into other systems.

X3D SUPPORT FOR TERRAIN VISUALIZATION

VRML respectively its successor X3D were introduced by the Web3D Consortium to distribute interactive virtual worlds on the web. Both are mark-up languages and standardized. Whereby X3D is fulfilling the concepts of XML. Besides X3D is specified more modular. The rendering concept is mainly based on a scene graph definition and a node structure (Web3D Consortium, 2004). VRML and X3D are accomplishing the basic concepts for a 3D GUI. Concepts of constructing a core virtual world and especially the *External Authoring Interface* (EAI) grading the possibilities around X3D/VRML up. By using the EAI, one can add individual functionality to virtual worlds. Developed either by scripting or higher programming languages, 3D scenes can get highly interactive (Held, 2004).

X3D is very useful for developing service for interactive three-dimensional visualization on the Internet. X3D describes 3D object by using scene graph structure. Scene graph is hierarchical tree-like structure that describes the entire 3D scene, including geometrical object representation, object attributes and relations between objects (Carey, 1997). Scene graph makes it easy to build complex objects composed of simple ones. Separate models and scenes could be easily combined in order to build complex virtual environment. A node or a sub-tree presents each object, and it can be only one entity or composition of spatially or logically related entities. Each object in the scene has its own unique name. Primitive objects can be enhanced by additional attributes, as material, texture or some other attribute, by using attribute-defining nodes. Attributes defined in one node are applied to all children of that node.

Another interesting aspect of X3D's data structure is that the *IndexedFaceSet* and *IndexedLineSet* constructs of X3D do not only contain the xyz-coordinates for an object, but also an index list of pointers to these coordinates. In other words, 3D objects are stored with 'internal' topology. This means that nodes (vertices) that are part of two or more faces (sides) of the object only have to be stored once (per object). The way 3D data is stored in X3D is therefore very efficient. The second X3D construct that is interesting for 3D geo-data retrieval is the <*Anchor/*> element. This can hold a URL to be accessed when an event occurs (e.g. a mouse click). This can be used for 'identify' operations, or - in OpenGIS terminology- a *GetFeature* or *GetFeatureInfo* request (De Vries, 2004).

A VRML/X3D node can be compared with object or class in Object Oriented (OO) languages like C++ or Java. Objects are generated dynamically after the browser parses the VRML/X3D file. VRML/X3D has 54 nodes that cover almost every aspect of 3D modeling. In order to model terrain, *ElevationGrid* node is used together with *QuadLOD* node to model Level Of Detail (LOD) functionality. *QuadLOD* node is derived node type developed in order to enable implementation of LOD (Reddy, 1999) algorithm. LOD algorithm enables the viewer to transfer and render only the detail that can be seen on the raster display, depending on the virtual viewpoint. One node can only contain data for two levels of detail, but using the quad-tree-like structure, multiple level of detail can be achieved. LOD algorithm should be able (Zlatanova, 2002):

- To form a terrain view with varying detail level, based on viewpoint location and terrain configuration, meaning that areas with small or none height variation can be represented with fewer polygon count than areas with high level of variations, such as canyons, cliffs, and mountains.
- To solve appearance of cracks in terrain rendering that appear due to resolution mismatch of neighboring quadratic areas at different levels of detail.
- To enable smooth and seamless transition between terrain tiles that present the same area but at different level of detail.

To achieve such functionality, it's not enough to use only VRML/X3D descriptive language; therefore, we looked for possibility of introducing procedurally described behavior. VRML standard enables procedures to be defined through the use of *External Authoring Interface* system, basically a framework of Java base classes that can be sub-classed or used directly, and when linked dynamically with VRML/X3D browser, used as a behavioral description of VRML objects and scenes. To use/subclass these classes, one could use Java or JavaScript programming languages.

VRML/X3D node used to link Java or JavaScript with VRML/X3D world is *Script* node. This node has input and output ports like almost every node in VRML/X3D. In VRML/X3D terminology, ports are equivalent to *Get/Set* functions of some object in OOP terminology. Output ports are also called *events*. VRML/X3D ports or events are used to enable communication between VRML/X3D objects: it's possible, for example, for proximity sensor to switch the light on in a virtual environment when the virtual viewpoint reaches the certain threshold. The same logic is used to incorporate a Script node into virtual environment. Script node has no direct virtual (or visual) representation in VRML/X3D scene, but it can be used as a gateway to receive events, pass them to Java code for processing, and pass programmatically generated events back to the virtual world. Java code can also be used to access *Factory* engine of the VRML browser that has the ability to introduce new objects into the VRML scene. JavaScript can be used for simple world scripting, and for processing user input from an environment external to a VRML world, such as a web page that has VRML world embedded in it. The best thing about JavaScript is that it can be coded directly inline into the VRML scene or HTML environment that surrounds the VRML.

A snippet of JavaScript code that is used to model a simple behavior together with a VRML code that links it with a scene is shown in figure 1. Java stands out as a complicated but powerful medium for programming VRML objects behavior.

```
DEF MoveObject Script {
        eventIn SFFloat set_timeFraction
        eventOut SFVec3f location
        url "javascript:
        function set_timeFraction (time_fraction, tm){
        location[0]   =   0.0;
        location[1]   =   4.9*(1-time fraction);
        location[2]   =   0.0;
}" }
DEF Timer TimeSensor {
         loop TRUE;
}

ROUTE Timer.fraction changed TO  MoveObject.set timeFraction

ROUTE MoveObject.location_changed TO MovingObject.set_translation
```

Figure 1: JavaScript sample.

It enables a powerful programming environment, and executes a lot faster then a JavaScript code. Java has advanced network communications abilities, and can be used efficiently to model user interface into a VRML world, that has no other user interface but the one that enables us to move in virtual space.

In order to efficiently build VRML/X3D scenes, we used the possibility of language extension building of VRML/X3D language through PROTO declarations tag <*ProtoDeclare*>. By utilizing PROTO declarations, we have been able to develop usable templates for scene objects, allowing us to easily define transformations from Web Service Application memory space to VRML/X3D standard coding. Developing a template for each scene object we gained in readability of our framework design. Creating new derived or composed classes by PROTO Declarations also allowed us to reduce the file sizes compared to unstructured files with a lot of redundant data repetition.

GINIS WEB 3D MODELER FRAMEWORK

GINIS Web 3D Modeler is a Web Services framework developed at Computer Graphics and GIS Laboratory at Faculty of Electronic Engineering in Niš, Serbia and Montenegro. It is an experimental framework for testing the possibilities of presenting virtual world on the Internet containing GIS informative capabilities.

Conceptual model of the framework is shown in figure 2. By using X3D methodology, all of the actual 3D graphics rendering process is done solely on the client side. Interaction with the user is done inside the virtual environment, or through the scripts embedded in the web page that has access inside the virtual world. To access virtual environment, user selects geographic area of a special interest, and gets directed to a page with 3D rendering of a desired area, together with the appropriate functionalities of that view. The main part of the framework is GINIS Server, which is responsible for texture extraction, DEM (Digital Elevation Model) data extraction, as well as attribute data extraction. The special continuous georeferenced raster map, formed at CG&GIS Lab at Faculty of Electronic Engineering in Niš (Rančić, 2003b) is used for the texturing of the terrain. The server uses data from the GINIS Web Map Service, GINIS DEM data server and GINIS GIS object database.

GINIS Web Map Service is in charge of creating maps of geo-information. It fully complies with the Web Map Service Implementation Specification (WMS) (OGC, 2001) and has been in use for a period of time, constantly improving its performance and quality of service.

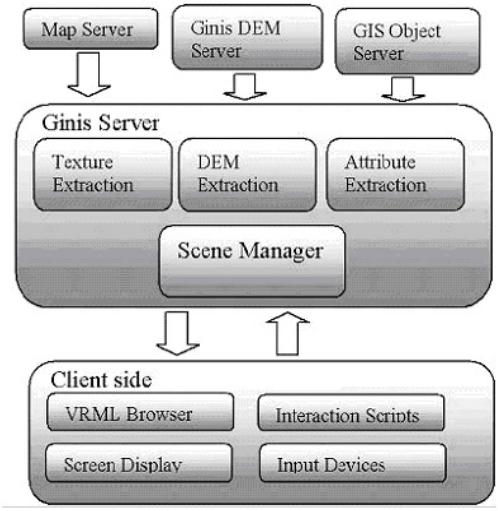

Figure 2: Ginis Web 3D Modeler framework.

Since editing/manipulating of data is one GIS principle, the GINIS Web Feature Service, complying with a Web Feature Service Implementation Specification (OGC, 2002) is a must as well. Operations of a Web Feature Service (WFS) are insert, update, delete, query and discover data. The data is represented in form of GML, another OGC standard for exchanging geo-information. Both, WMS and WFS, are based on the HTTP protocol for transferring data. In our model, GIS Object Server is compliant with a Web Feature Service standard.

Decoupling Web Feature Service and Web Map Service from our GINIS Server enables us to focus on implementation of only 3D GIS functionalities, leaving 2D GIS functionalities development on the side of the existing services. In this way, whenever we improve our WMS or WFS with a new feature, it is a very easy task to enable GINIS Server to utilize this new feature. And for a numerous cases, no change in GINIS Server was needed in order to enable new features to show in 3D GIS environment display. Interoperability and compatibility offer multiple usage of geo-information as well as the creation of added value and more convenient data updating, and thereby assure the sustainability and quality of 3D data resources (Altmaier, 2003).

The design and implementation of the operators for 3-dimensional geographical analyses are closely related to the modeling scheme of geographic objects besides the complex computational geometry algorithm itself. The more amount of data is used for modeling an object, the deeper level of detail can be obtained. But the level of detail should be traded off with the availability in managing the geographic vector data and transmission efficiency in web-based application. We used rather simple modeling scheme proposed in (Wang, 2004) and (Tao, 2004) for 3-dimensional geographic objects using mainly extrusion method. With this method, 3D objects such as road or pipeline can be simply modeled using 2-dimension vector data.

Displaying different layers of 2D GIS information does not raise an issue to be solved inside 3D visualization framework's scope, but on the Web Map Server Side. Different layers are drawn on the same surface texture, and such texture is used in terrain tile visualization. This overlay process can be done on the server, or, as we prefer, on the client side, enabling us to turn different layers on an off, at a user convenience, without having to transfer new bitmaps for every layer deselected. More of a problem presented a visualization of different layers of 3D objects that needed to be

shown together with the terrain, such as bridges, buildings, or clouds above the terrain. Spatial reference system of these objects had to be the same, and that was a major requirement for connection with a feature server. Additional 3D Features are delivered to GINIS server from a GIS Object Server. GIS Object Server is a model of a 3D feature server used to provide additional display elements. It is a requirement that these display elements are represented using international standards, like VRML/X3D, or SVG, so that incorporating these elements into a final model would not require framework rebuild, but only reconfiguring for a different use. GINIS DEM Server is another 3D feature server, but here is observed separately, because of the critical performance issues we had to solve considering 3D terrain data request from GINIS Server to a DEM Server, and more complex data management schemes needed to be implemented on the DEM Server, then on a common feature server.

Scene Manager is a central part of GINIS server, and it is responsible for real-time scene management and LOD implementation. LOD algorithm is implemented in such a way that the tile used for the area of interest is being replaced with a tile that has a higher resolution per measure unit, when the user approaches some part of the terrain (Röttger, 1998). This is done dynamically without movement interruption through the virtual scene.

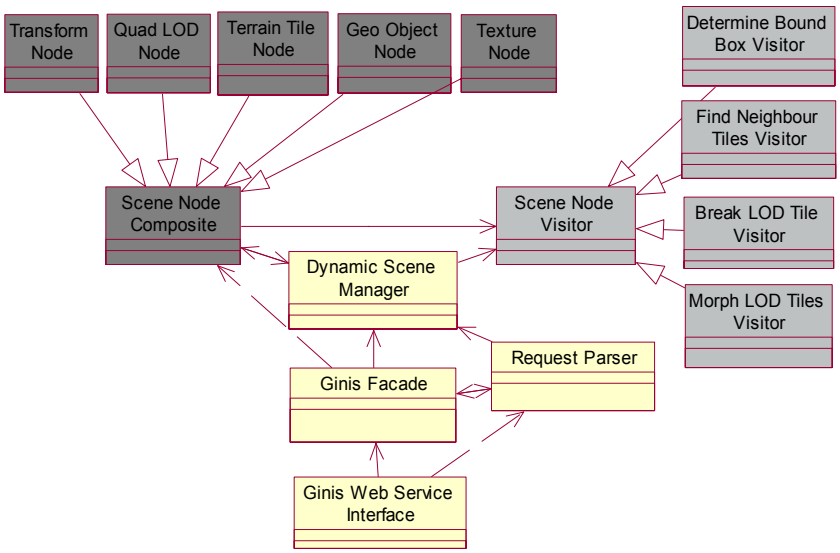

Figure 3: GINIS Scene Manager object model (subset).

In figure 3 is shown an overview of a subset of C++ classes from GINIS Server framework, in order to show basics of proposed architecture of a framework. By using *Composite* pattern in modeling Scene objects, we have followed already established standards in implementing Scene Manager applications. *Visitor* pattern is used to implement different operations on the scene objects, without the need to burden scene objects with different algorithms and functionalities (Vlissides, 1999). Scene object are a representation of an actual 3D objects and transformations, and are all serialized as a VRML/X3D objects, simplifying transformation from dynamic data model to VRML/X3D data to be sent to the client. Serialization itself is done using VRML/X3D templates, so that the transformation can be done independent of the modeling style a particular client requires.

Interaction with the user is done on the client side by using JavaScript programming, and request from the user are modeled in the following forms:

1. User's avatar moves to a different location, in which case we use bounding boxes sensors to detect the position of the avatar.
2. User's avatar approaches some object at a smaller distance, in which case we use proximity sensors and bounding boxes. In this case we need to invoke generation of a terrain tile of a finer grain resolution, and invoke LOD functionality
3. User clicks on some object, thus requesting additional information about this object. Such requests are object-specific, and are processed together with WMS and WFS.
4. User chooses a set of layers to be displayed. Whether 2D or a 3D layer is selected, coordinates of a user's avatar position are transferred as a part of a query, and the resulting object is included in the scene.

Additional query scenarios can be modeled easily into our framework, by adding functionalities on both the server side (expanding the server framework), and the client side, using JavaScript. Events distribution architecture on the client side is shown in figure 4.

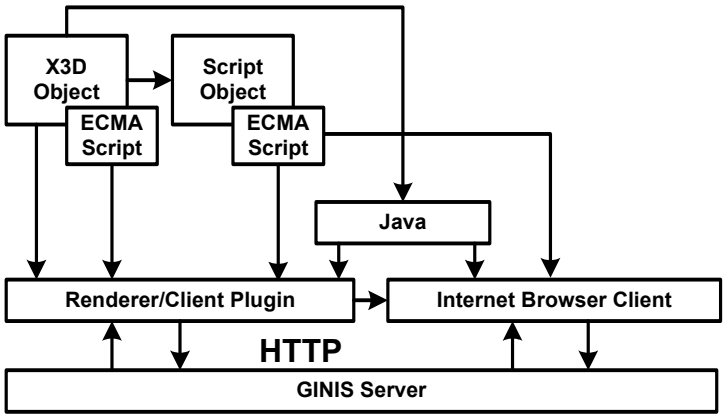

Figure 4: Client-side model.

SYSTEM PERFORMANCES

A sample of client-side display rendering, obtained after a selection of geographic area of interest is shown in figure 5. Our system enables both the semantics and spatial navigational interface capabilities. On the left side of the client browser window, user can select different layers for which he or she has interest in. Choosing a layer is independent of the nature of the layer data, whether the layer is presented as a two-dimensional bitmap layer overlaid on an existing layers or the layer presents a set of 3D objects that should be integrated in a 3D scene. Navigation in a 3D environment is done through a VRML/X3D browser interface, using fly, examine and walk movements. The implemented LOD algorithm enables shorter period for delivery of data for a certain area that is currently presented to the user. Deeper investigation of that area triggers the activation of additional object with a higher level of detail, and these detailed objects replace the low-level of detail objects through a seamless transition. This seamless transition is enabled through a caching scheme where the client always loads one level of detail ahead, in a background process, while a user avatar approaches some terrain tile.

Table 1 shows average measured frame rates with and without LOD functionality, and also with textures switched ON and OFF. Measured terrain sample had 60,000 height points, and was textured with 1024x1024 24-bit detailed terrain bitmap. The test PC configuration was Athlon 1 GHz CPU, 512 MB DDR RAM, GeForce 2 graphics card, Windows 2000 OS, MS Internet Explorer 6.0, and

Cortona VRML browser (ParralelGraphics, 2004). It can be observed that we gained 18 frames per second with the textures rendering off, which can be considered a small gain, but makes almost 50 percent gain with detailed terrain textures rendering, and together with optimized bandwidth requests justifies the introduction of LOD algorithms.

	Detailed Textures OFF	Detailed Textures ON
LOD switched OFF	70	23
LOD switched ON	88	34

Table 1: Measured average frame rates with detailed textures included and excluded.

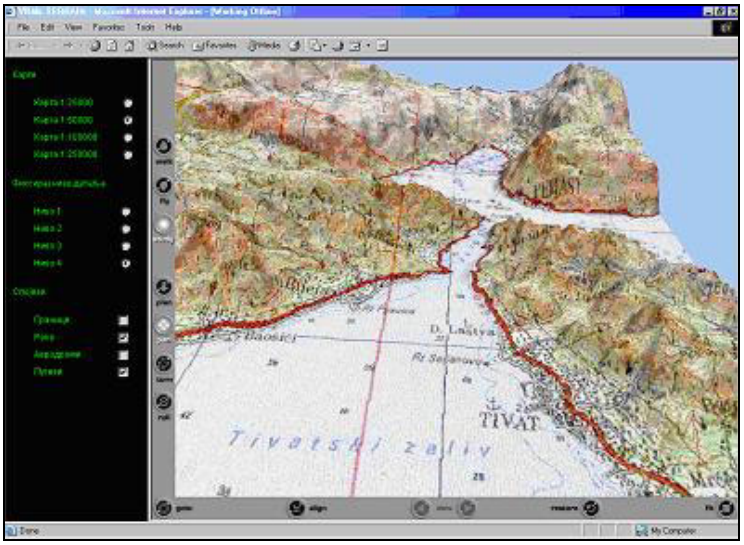

Figure 5: Screenshot from the Ginis Web 3D Modeler client.

CONCLUSION

The ways of developing a system that has all the benefits of the two worlds: organized geographical and georeferenced information capability of a full-featured GIS system, and an easy-to-use three-dimensional presentation of GIS information are explained in this paper. There are numerous applications of such system. System like this can be used at a public-place automatic-information desk, or, with a recent increase in small mobile devices performance, it could be easily executed on any mobile device that has a ported version of X3D browser. Java and JavaScript programming languages enable us to expand existing capabilities of X3D, and are powerful tools for design of user interaction system and performance optimization algorithms. GINIS Web 3D Modeler framework is a step toward designing a truly powerful interface to existing GIS datasets, available to larger population of users. VRML is an open standard, but recently, X3D as an XML-compatible standard has gained its position in a WWW community. X3D is by all concepts similar to VRML, and also has same basic nodes with same structuring rules, and it is an easy task to convert existing VRML-based systems in a way to support X3D. The only disadvantage of X3D is the slow response of VRML browser manufacturers. Broad implementation of X3D is the first step in developing this system as a full-featured Web service.

BIBLIOGRAPHY

Altmaier, A. and Kolbe, H. T., 2003, Applications and Solutions for Interoperable 3D Geo-visualization, Proceedings of the Photogrammetric Week 2003, Stuttgart, Germany.

Carey R. and Bell G., 1997, The Annotated VRML 2.0 Reference Manual. Addison-Wesley.

De Vries, M. and Zlatanova, S., 2004, Interoperability on the Web: the case of 3D geo-data, in: P. Isaias, P. Kommers and M. McPherson (eds.), Proceedings of the IADIS International Conference e-Society 2004, Avila, Spain, July 16-19, pp. 667-674.

Held, G., Rahman, A. A. and Zlatanova, S., 2004, Web 3D GIS for urban environments, Proceedings of the International Symposium and Exhibition on Geoinformation 2004 (ISG2004), Kuala Lumpur, Malaysia.

Nebiker, S., 2002, Design and Implementation of the High-Performance 3D Digital Landscape Server 'DILAS', Joint International Symposium on Geospatial Theory, Processing and Applications, Ottawa, Canada.

OGC, 2001, Web Map Service Implementation Specification. (http://www.opengis.org/docs/01-068r2.pdf).

OGC, 2002, Web Feature Service Implementation Specification. (www.opengis.org/docs/02-058.pdf).

ParallelGraphics, 2004, Cortona VRML Client, (http://www.parallelgraphics.com/products/cortona/).

Rančić D., Dimitrijević A., Milosavljević A., Mihajlović A. and Kostić A., 2003a, Virtual GIS for Prediction and Visualization of Radar Coverage. Proceedings of the Third IASTED International Conference on Visualization, Imaging, and Image Processing - VIIP 2003, Benalmadena, Spain, September 8-10, pp. 970-974.

Rančić D. and Đorđević-Kajan S., 2003b, MapEdit: solution to continuous raster map creation, Computer and Geosciences, Vol. 29, No. 2, Elsevier Science, pp.115-122.

Reddy, M., Leclerc, Y. G., Iverson, L., Bletter, N. and Vidimce, K., 1999, Modeling the Digital Earth in VRML, Proceedings of SPIE - The International Society for Optical Engineering. Volume 3905, pp. 113-121.

Röttger S., Heidrich W., Slusallek P. and Seidel, H., 1998, Real-Time Generation of Continuous Levels of Detail for Height Fields. In V. Skala, (ed.) Proceedings of WSCG '98, pp. 315-322.

Stoter, J. and Zlatanova, S., 2003, 3D GIS where are we standing? Joint Workshop on Spatial, Temporal and Multi-Dimensional Data Modeling and Analysis, Quebec city, Canada.

Tao, C. V., Fei, C. et al. 2004, A Web-Based Geotechnical Data Sharing and Analysis System, Geoinformatics, International Journal for Geographic Information Systems, Submitted.

Vlissides, J., 1999, Tooled Composite, C++ Report, September 1999.

Walsh A. and Bourges-Sévenier M., 2001, Core Web3D. Englewood Cliffs, NJ: Prentice-Hall.

Wang, Y., Bao, Q., and Tao, C.V., 2004, GeoServNet 3D Analyst: Enabling Web-based 3D Visualization and Analysis, Master Thesis. Department of Earth and Space Science and Engineering, Faculty of Pure and Applied Science, York University.

Web3D Consortium, 2004, (http://www.web3d.org/).

Zlatanova S., Rahman A. and Pilouk M., 2002, 3D GIS: current status and perspectives. Proceedings of ISPRS, 8-12 July, Ottawa, Canada.