Integration of rendering technologies and visualization techniques to improve 3D Mobile GIS applications

M. Farnaghi [a], A. Mansourian [a], A. Toomanian [b]

[a] Dept. of GIS, Faculty of Geodesy & Geomatics Eng, K.N.Toosi University Of Technology, Vali-e-asr St., Mirdamad Cross, Tehran, Iran, P.C 19967-15433 – (farnaghi@dena.kntu.ac.ir, mansourian@kntu.ac.ir)

[b] GIS Centre, Dept of Physical Geography and Ecosystems Analysis, Lund University, Sölvegatan 12 SE-223 62 Lund, Sweden – ara.toomanian@nateko.lu.se

Workshop on quality, scale and analysis aspects of city models

Keywords: 3D visualization, GIS, Mobile computing, Tiling, Direct3D Mobile

ABSTRACT: This paper proposes integration of different technologies, techniques and standards to improve the efficiency of 3D mobile GIS applications. The paper describes different challenges that developers face while developing a 3D mobile GIS application. It then investigates different solutions to resolve the problems. Integration of client/server architecture and web services technologies together with culling methods, Level of Detail (LOD) and tiling mechanism was a proper solution that adapted and implemented. The frame rate was used as an indicator for the test. The results of the implementation and tests showed an improvement in 3D mobile GIS application when using the proposed solution.

1. INTRODUCTION

3D Mobile GIS has a variety of applications in urban management, disaster management, tourism, utility management, etc. However design and development of 3D Mobile GIS is a challenging work due to limitations of mobile devices in comparison with desktop computers:

- All mobile and handheld devices suffer from low CPU resource.
- Most of mobile and handheld devices have low memory.
- Mobile and handheld devices have limited visualization capabilities.
- 3D visualization in mobile devices needs an appropriate 3D rendering API.
- Mobile networks don't have enough band-width to transfer large amount of data.
- 3D spatial data is commonly larger in size than 2D spatial data and therefore transferring such data over networks could be a challenging work.

During the last few years, different studies have been conducted with respect to rendering and visualizing 3D data on mobile devices, emphasizing on just some aspects of the above-mentioned challenges. These studies have also rarely attended on developing an operational 3D Mobile GIS. In this respect, Lipman (2004) developed a stand-alone mobile application that renders 3D steel structures. Huang et al. (2007) developed a mobile application to render 3D objects on mobile devices. Nadalutti et al. (2006) worked on rendering of X3D content on mobile devices. Chung et al. 2009 developed a 3D virtual viewer on mobile device for wireless sensor network-based RSSI (R… S… S… I…) indoor tracking system. However, none of these solutions are applicable as a total solution for development of 3D mobile GIS applications. In fact, due to the large size of GIS data, storing 3D data on a mobile device is not a rational solution for 3D spatial data exploration on mobile devices with low memory.

Baldauf et al. (2008) presented client/server architecture to explore 3D spatial data on mobile devices. They developed a J2ME client application that could connect to the server application and retrieve required 3D data. Although, this client/server solution is more suitable than the stand-alone approaches, it is not still a proper solution for 3D mobile GIS due to the rendering mechanism which is used. Baldauf et al. (2008) utilized a server side rendering mechanism that is not a proper rendering method for wireless network environment, as will be described later in section 2.3.

Considering the above descriptions, this paper intends to propose an integrated solution to improve 3D mobile GISs by resolving the limitations of mobile devices, described earlier. To achieve the aim, the paper is organized as follow. In section 2, different technologies, standards and techniques which should be utilized and integrated for the development a 3D mobile GIS is depicted. Design and development of a prototype 3D mobile GIS, based on the proposed solutions, is explained in section 3. The last section of the paper includes conclusions and future trends.

2. MATERIALS AND METHODS

2.1 Client/Server Architecture

Despite centralized architecture in which a computer handles all processing, including input, output, data storage, retrieval, and analysis, client/server architecture is a distributed application architecture that partitions tasks or workloads between service providers (servers) and service requesters, called clients (DAA, 2009). Often clients and servers operate over a computer network on separate hardware. A server is a high-performance host that is a registering unit and shares its resources with clients. A client does not share any of its resources, but requests a server's content or service function.

The development of 3D Mobile GIS applications with client/server architecture is a suitable solution to the first and second challenges, described earlier. In a client/server application some part of processing and analysis tasks are operated at the server side. Therefore, the client processes are reduced. Additionally, in a client/server application, there is no

need to store all of the required data, locally, on a client device. In fact, client application requests the server to send required data on demand. It is perceived that the client/server architecture can resolve the limitations relevant to CPU and memory of mobile devices in 3D Mobile GIS.

2.2 Web Services Technologies

The mobile networks do not always have permanent and reliable connection. Thus, the classical client/server architecture, which presumes always-on broadband communication, is rarely suitable for mobile applications. A more resilient model consists in developing a full-featured client/server application is to use Web services technologies (Badard 2006).

Web services technologies are realization of a software design pattern called service-oriented architecture (SOA). In SOA, the services are applications which present piece of functionality that fulfils users' (human or software package) requirements. Such a service is generally implemented as a course-grained, discoverable software entity that exists as a single instance and interacts with applications and other services through a loosely coupled, message-based communication model (Zimmermann et al., 2004). The service is a software entity that is available over the Internet or private (intranet) networks; uses a standardized messaging system; is not tied to any one operating system or programming language; is self-describing via a common grammar; and is discoverable via a simple find mechanism (Brown et al., 2002).

Web services are implemented using a collection of standards, including network transport protocols such as TCP/IP and HTTP, meta-language standards such as XML, service communication standards such as SOAP[a], service description standards such as WSDL[b] and service publishing and discovery standards such as UDDI[c].

In order to develop an operational server component for the 3D mobile GIS application, they should be developed as a standard service, providing interoperable interface for any other mobile application as well as a more reliable client/server interaction. Development of server components of the system as services, using web services technologies, provides the system with such capabilities.

2.3 3D Rendering on Mobile Devices with Mobile Graphic APIs

There are several approaches for the rendering of 3D data on mobile devices. They can be divided into three categories (Mikovec et al. 2006):

- Server-side or remote rendering: This is a solution in which the rendering process is performed on a server computer with powerful graphics accelerator. The rendered scenes are sent to the client application to be displayed. Sanna et al. (2004), Pasman and Woodward (2003) and Baldauf et al. (2008) have used this approach in their studies.
- Client-side or on-board rendering: This approach renders the 3D data completely on the mobile device. Zunino et al. (2003) has adapted this approach as an example.
- Hybrid rendering: The hybrid solution balances the workload between the server and the client. It is used by Hekmatzada et al. (2002) and Diepstraten et al. (2004).

Since mobile devices have only recently reached a performance that allows them to manage 3D graphics, most of the 3D rendering applications on mobile devices focuses on remote rendering rather than on-board or hybrid solutions (Nudalutti et al., 2006). The remote rendering for mobile devices needs to have a permanent and reliable wireless connection that is not always available. Additionally, considering the work load of the rendering process, the server will not be able to support too many simultaneous client applications.
The hybrid rendering methods are complicated for the implementation. As a result, in this research the 3D data are completely rendered on the mobile devices (using on-board rendering solution) through a proper Mobile Graphic API.
There are different Graphic APIs which are applicable for development of graphical applications on desktop computers. Three of the well-known graphic API for mobile devices are:

- Mobile 3D Graphics (JSR 184)
- Microsoft Direct3D for Mobile (D3DM)
- OpenGL ES

The Mobile 3D Graphics API, commonly referred to as M3G, is a specification defining an API for writing Java programs that produce 3D computer graphics. It extends the capabilities of the Java Platform Micro Edition. M3G is a completely high-level API that originated from previous APIs such as Java 3D and OpenInvendor. Commonly standardized high-level APIs are not as popular as low-level ones for writing dedicated engines, such as a game engine (Pulli et al., 2008) or a GIS engine. Hardware-accelerated low-level APIs can provide more flexible tools for development of any dedicated engine. Additionally, if the/a developer wants to create such engine using a high-level API such as M3G, the entire program should be developed in Java incurring a significant performance penalty. Therefore, in this study, we don't recommend to use M3G in 3D mobile GIS applications developments.

Microsoft Direct3D Mobile is a low-level API that provides support for 3D graphics applications on mobile devices, (Direct3D, 2009). It is a subset of Direct3D API found on Microsoft Windows-based desktop systems. Microsoft Direct3D Mobile is optimized for use on embedded systems.

OpenGL ES is a cross-platform API for 3D graphics on embedded systems - including consoles, phones, appliances and vehicles. It consists of well-defined subsets of desktop OpenGL, creating a flexible and powerful low-level interface between software and graphics acceleration, (OpenGL ES, 2009).

During the last decade, various researchers such as (Miszalok, 2009; Roy, 2002 and Akenine-Moller et al., 2008) compared these technologies from different point of views. As a result of these studies as well as the investigations of the present research, the selection of one of these two technologies for the implementation, generally relates to the experience of development team and the implementation platform of the end-user device. It is expected to address the third and the fourth challenges by use of Direct3D Mobile or OpenGL ES..

[a] Simple Object Access Protocol or Service-Oriented Access Protocol
[b] Web Service Definition Language
[c] Universal Description, Discovery, and Integration

2.4 Rendering Techniques

While this paper proposes on-board rendering approach, it is necessary to reduce the workload of mobile devices by adapting proper rendering techniques. In this section some 3D rendering techniques are discussed which can be used to improve the performance of 3D applications.

2.4.1 Viewing frustum culling

In 3D space the viewpoint can only see a partition of space that is called frustum. Geometries that are not within the inner space of this frustum are not visible to viewpoint and can be ignored (Chung et al., 2009). The act of determining the objects that entirely lie outside of the viewing frustum and ignoring them from rendering process is called viewing frustum culling (Akenine-Moller et al., 2008).

2.4.2 Back-face culling and Occlusion culling

It is impossible to directly see the back side of an object, when it is modelled as a solid object. Additionally it is not possible to see the objects that are entirely behind other opaque objects. So, to save time, back-facing geometries and occluded geometries can be ignored to be rendered. This technique is known as back-face and occlusion culling (Akenine-Moller et al., 2008).

2.4.3 Multi-resolution rendering

Complex geometries can be simplified into some levels and depend on the criteria like viewpoint location, simplified copy geometry could be sent to rendering stage.

2.4.4 Level of Detail (LOD) and Tiling

The term level of detail (LOD) refers to the capability of providing and using different representations of spatial object, at different levels of accuracy and complexity, depending on specific application needs (Danovaro et al., 2006). A large dataset can be tiled into different levels of a pyramid. Each level of this pyramid covers the same area of the source dataset. Each level of this pyramid contains a simplified version of data with a pre-calculated Level of Detail. Tiling of data with level of detail can help the application to access the large datasets partially and on-demand. In other words, using tiling with level of detail each user of the system just needs to receive appropriate tiles of data in proper level to gain required data. This can prevent the application from loading large size datasets.

It seems that the combination of culling methods and multi-resolution rendering can significantly improve the performance of client application by addressing the third and the fourth challenges. However, using multi-resolution increases the size of the stored data on mobile device and considering the low storage size of mobile devices this can be a problem. To solve this problem a tiling mechanism along with level of detail (LOD) can be used. Using these mechanisms, a large dataset is divided into smaller parts at different levels of a pyramid. At each level, level of detail is calculated based on LOD strategy of user. Then the tiled dataset is served through the server components over the communication infrastructure. Mobile device can establish a connection to the web service and receive its data partially. In other words, this research proposes applying culling methods and multi-resolution models along with tiling mechanism and LOD for developing 3D mobile GIS applications.

Additionally tiling mechanism can effectively respond to low band-widths of mobile networks and large size of 3D data which are the fifth and sixth challenge, described earlier. Using the tiling mechanism, client application always receives the optimum part of data and therefore the data transfer between client and server will be efficient.

2.5 3D Graphic formats and GIS formats

3D mobile GIS applications should support vector, raster and elevation data. Vector data should be stored and transferred in a 3D capable format. The format should be also a GIS format to be able to carry the topology data of vector datasets. Therefore standard 3D formats such as VRML and X3D are not applicable in this context. Evaluation of different spatial data formats proves that currently GML[a] is one of the most popular and complete formats that can be used to store and transfer 3D spatial data for 3D mobile GIS.

Raster data should be stored and transferred through an image format that does not allow spatial and quality loss. Additionally, the raster format should be capable of carrying spatial reference. This paper proposes tagged image format (*.tif) as a proper format for raster data, due to it's capability of carrying spatial reference information.

In order to store and transfer elevation data, triangulated irregular network (TIN) data is stored in X file which is a well known format in 3D visualization world. X file format is a binary format and optimum for storage and transferring 3D data.

3. SYSTEM ARCHITECTURE AND A PRACTICAL TEST

This section describes the design, implementation and test of a prototype 3D mobile GIS application, based on the solutions described earlier.

3.1 Architecture

The proposed system has a client/server structure where the overall architecture is illustrated in figure 1. The system has four main subsystems: (i) the manager subsystem responsible for pre-processing 3D data and tiling input data into the server data repository, (ii) the server subsystem which loads 3D tiled data from the data repository and responds to the client, (iii) the communication layer that implements the communication mechanism, and (iv) the client application that connects to the server and receives the data and renders 3D geodata.

[a] Geography Mark-up Language

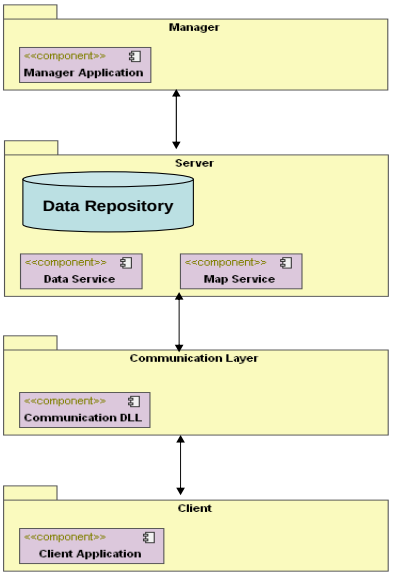

Figure 1 Overall system architecture

3.2 Manager Subsystem

The manager subsystem is responsible for pre-processing data and tiling specified datasets. In order to tile spatial data, a tiling mechanism is developed that tiles each dataset in a pyramid in WGS 1984 geographic coordinate system. Each level of this pyramid covers an area of 180°×360° in latitude and longitude directions. In each level (n) there are $2^n \times 2^n$ tiles of data.
Minimum number of levels can be computed based on resolution of initial dataset through Eq. 1.

$$N = \text{ceiling}(\log_2^{\frac{360}{R \times T}}) \quad \text{(Eq. 1)}$$ [a]

The tile number (row and column of the tile) of desired coordinate is calculated through Eq. 2.

$$i = floor(\frac{(180+\lambda)\times 2^n}{360}) \quad \text{(Eq. 2)}$$ [b]

$$j = floor(\frac{(180-\varphi)\times 2^n}{360})$$

The coordinate of top left corner of each tile can be computed through Eq. 3.

$$\lambda = \frac{360\times i}{2^n} - 180 \quad \text{(Eq. 3)}$$ [c]

$$\varphi = 180 - \frac{360\times j}{2^n}$$

There is a convention for naming of each tile in each level. The manager subsystem is developed as a desktop application that helps to manipulate data in data repository. The application can access to data repository through the LAN[d].

3.3 Server

The server consists of a data service and a map service. These two services are completely developed based on web services technologies using Microsoft ASP.NET Web Services Technology. The map service is responsible for presenting general setting for the client application such as layer names and rendering conditions. The data service is responsible for transferring requested data to the client. The data service delivers appropriate tiles to the client based on the client's needs.

The server contains a data cache that manages frequently requested data on the server. There is a method that synchronizes the cached data with the source data. This method cleans the cached data if the main data is changed or deleted. In addition, there is a client side cache which is implemented on the communication layer.

3.4 Client

Client application is responsible not only for the interaction with the users but also the data presentation to them. The client application is able to establish connections to the server using communication platforms that are provided by the mobile device. Once the client application establishes a connection, it creates a communication thread and sends a map request to the server and receives an XML file that contains application settings. After applying those settings, the client creates another communication thread and sends viewing parameters to the server and receives the appropriate tiles of data. The data will then be rendered on screen.

The client application is developed using Microsoft .NET Compact Framework. The programming language was C#. The client application runs on most Windows CE based platforms such as Microsoft Windows Mobile Operating System.

The problem of rendering 3D graphics on mobile devices is simplified by Direct3D Mobile and OpenGL ES. In this research, the client application favours the Direct3D Mobile for rendering 3D data on mobile device. The client uses a culling method that substantially reduces the geometry sent to the rendering stage. In fact, before sending geometry to rendering stage, two methods begin to work: (1) viewing frustum culling method, (2) occlusion and back-face culling method. The former method prevents the selection of objects which are not within viewing frustum of viewpoint. The latter method prevents selection of the objects which their line of sight to the viewpoint is blocked with other objects. After applying these methods, proper geometries are selected and will be sent to rendering stage.
Additionally, a multi-resolution mechanism is implemented in the client application that supports focusing on a given area of the scene for detailed extraction of the relevant geometry.

3.5 Communication Layer

The communication between the client and the web services (server) is handled by the communication layer on top of the HTTP protocol. The layer implements some classes for communicating with the map service and data service.

---

[a] N symbol is used to denote minimum number of levels. R symbol is used to denote the resolution of input data. T symbol is used to denote the specified tile size.

[b] λ symbol is used to denote longitude of desired coordinate. φ symbol is used to denote latitude of desired coordinate. n symbol is used to determine the level in the pyramid. i and j symbols define the row and column of the tile in level n in which the desired coordinate will be.

[c] i and j symbols define the row and column number of the tile in level n. n symbol is used to determine the level in the pyramid. λ symbol is used to denote longitude of top left corner of the tile. φ symbol is used to denote latitude of top left corner of the tile.

[d] Local Area Network

This layer contains a data cache that stores requested data on the client side. This cache periodically checks the size of cached data and deletes part of them if the size exceeds a specified value. Additionally this cache contains a synchronization method that checks to see if the data is synchronized with the source data on the server based on the metadata of tiles.

3.6 Evaluation and Test

To test the system, the client application ran on a Dell Axim X51 PDA and PocketPC 2003 emulator. The server ran on a typical desktop personal computer. A X * X Km^2 area of Tehran, the capital of Iran, was selected as the case study and its 3D data including elevation data as TIN, building block data as volumes and road data as surfaces, were entered to the system. Maximum number of triangles viewed is at most 2000 for the specified data when the viewpoint is close to the map and less than 1200 when the viewpoint is far from the map. The scene resolution changes progressively during navigation.
To evaluate the system, frame rate was selected as an indicator of performance of the system. Frame rate (frequency), is the measurement of the frequency at which an imaging device produces unique consecutive images called frames. Frame rate is most often expressed in frames per second (FPS). Larger frame rate shows that the processes to generate frame in a visualization application are done faster and this means that the application is working more efficiently. To test the system, a path was determined on the test data. The viewpoint was moved through the path and frame rate of the client application were measured. **Error! Reference source not found.**3 and **Error! Reference source not found.** are two plots that show the frame rate of 3D GIS data rendering on client application in two different modes.

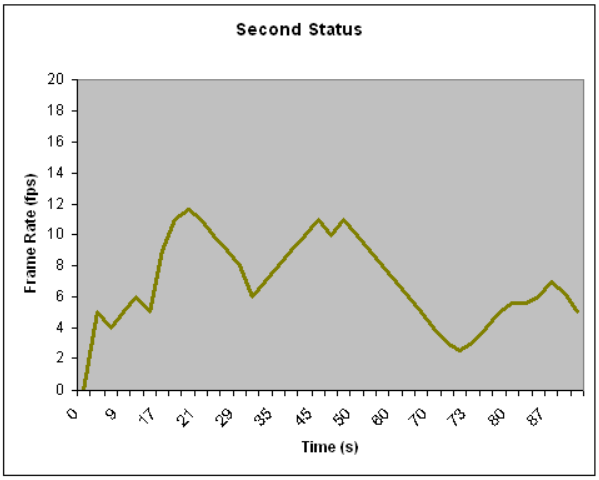

Figure 2 Frame rate of rendering the test data when culling, multi-resolution, tiling, and LOD is turned off and all the data is saved on client application

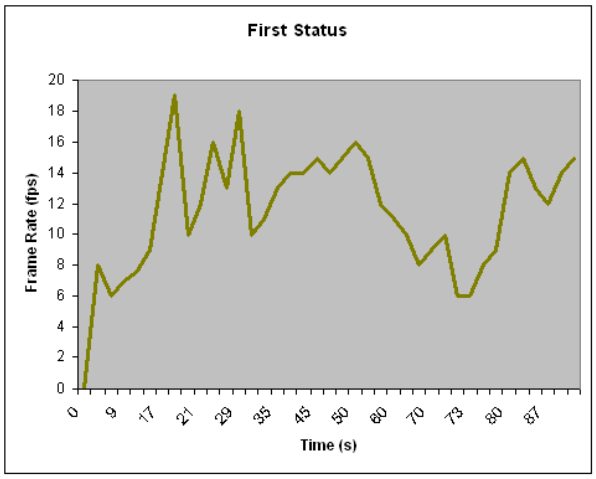

Figure 3 Frame rate of the test data when culling, multi-resolution, tiling, and LOD is turned on and data is progressively transferred to the client application by the server

Error! Reference source not found. shows the plot of frame rate of the client application when culling, multi-resolution, tiling, and LOD is turned off and all the data is saved on the client application. In fact figure 2 is a plot of a stand-alone application, without any of the recommended rendering techniques and tiling mechanism. Figure 3 shows the plot of the client application when these methods are all turned on and the data is retrieving from the server on demand. The plots show that the frame rate in the second situation is better than the first one. In the second plot, maximum frame rate reaches to about 17 fps[a] but in the first plot it doesn't exceed the 11 fps. Additionally in the second plot, the average frame rate is equal to 7.068 while in the first plot the average frame rate is equal to 11.803. It shows that the proposed architecture along with the recommended technologies and solutions can obviously improve the performance of 3D mobile GIS applications.

4. CONCLUSION AND FUTURE WORKS

This research intended to propose a 3D mobile GIS application that can operate on mobile devices in a proper manner. In order to reduce the work load of the 3D mobile GIS application, a client/server architecture based on web services technologies was recommended as suitable architecture for the system. The client/server architecture also addresses the problem of low storage size of mobile devices by storing large datasets on the server.

Considering the low band-width of mobile network a tiling mechanism was developed to split large datasets into various tiles in different levels of a pyramid. Each level of the pyramid contains a generalization of source data with a calculated level of detail. Tiled data are sent to client application by the server of the system. This approach allows the users to visualize and manipulate large GIS data on restricted mobile devices.

The server components of the system are developed as web services based on web services technologies standards, therefore these services can support any other applications on mobile devices or personal computers that need to access the tiled data of the system.

The system evaluated through a performance test which performed using frame rate indicator. The evolution show that

[a] frame per second

proposed architecture and utilized technologies can significantly improve the performance of the 3D mobile GIS applications.

For the future work, improvement of the user interface of the system and also evaluation of the system in an operational environment is considered. This implementation also can be used as a base for development of egocentric GIS applications or augmented reality.

5. REFERENCE

Akenine-Moller T., Haines E., Hoffman N., 2008. Real-Time Rendering, L K Peters Ltd.

Badard T., 2006, Geospatial Service Oriented Architectures for Mobile Augmented Reality, Proceedings of the 1st International Workshop on Mobile Geospatial Augmented Reality, Banff, Canada, p.73-77.

Baldauf M., Fröhlich P., and Musialski P., 2008, A Lightweight 3D Visualization Approach for Mobile City Exploration, Proceedings of TIPUGG, First International Workshop on Trends in Pervasive and Ubiquitous Geotechnology and Geoinformation GIScience conference

Brown, A., Johnston, S., Kelly, K., 2002. Using Service-Oriented Architecture and Component-Based Development to Build Web Service Applications. A Rational Software White Paper, Visited on: http://www.ibm.com/developerworks/rational/library/510.html, last access: June 2, 2009.

Chung W. Y., Lee B. G., and Yang C. S., 2009, 3D Virtual Viewer on Mobile Device for Wireless Sensor Network-based RSSI indoor tracking system, Journal of Sensors and Actuators B 140, 35-42.

DAA: Distributed Application Architecture, Sun Microsystems, http://java.sun.com/developer/Books/jdbc/ch07.pdf. Retrieved on 2009-06-16.

Danovaro E., De Floriani L., Magillo P., Puppo E., and Sobrero D., 2006, Level-of-detail for data analysis and exploration: A historical overview and some new perspectives, Journal of Computers & Graphics, Vol. 30, p-334-344.

Diepstraten J., Gorke M., Ertl T., 2004, Remote Line Rendering for Mobile Device, CGI '04: Proceedings of the computer graphics international. Washington, USA: IEEE Computer Society.

Direct3D Mobile, http://msdn.microsoft.com/en-us/library/aa452478.aspx, last access: June 15, 2009.

Hekmatazada D., Meseth J., Klein R., 2002, Non-photorealistic Rendering of Complex 3D Models on Mobile Devices, Proceedings of eighth annual conference of international association for mathematical geology, Vol. 2, p. 93-98.

Huang J., Bue B., Pattath A., and Ebert D. S., 2007, Interactive Illustrative Rendering on Mobile Devices, IEEE Computer Graphics and Applications, v.27 n.3, p.48-56.

Lee K., 2007, 3D Urban Modelling and Rendering with High Resolution Remote Sensing Imagery on Mobile 3D and Web 3D Environments, Proceedings of Urban Remote Sensing Joint Event, p. 1-5.

Lipman R. R., 2004, Mobile 3D Visualization for Steel Structures, Journal of Automation in Construction, Vol. 13, p. 119-125.

Mikovec Z., Cmolik L., Kopsa J., and Slavik P., 2006, Beyond Traditional Interaction in a Mobile Environment: New Approach to 3D Scene Rendering, Journal of Computers & Graphics, Vol. 30, p. 714-726.

Miszalok, V., (2009), OpenGL and DirectX, http://www.miszalok.de/Lectures/L05_OpenGL_DirectX/OGL_DX_english.htm, last access: June 29, 2009.

Nadalutti D., Chittaro L., and Buttussi F., 2006, Rendering of X3D Content on Mobile Devices with OpenGL ES, Proceedings of Web3D 2006, p. 19-26.

OpenGL ES, http://www.khronos.org/opengles/, last access: June 15, 2009.

Pasman W., AND Woodward C., 2003, Implementation of an Augmented Reality System on a PDA, In ISMAR'03: Proceedings of the second IEEE and ACM International Symposium on Mixed and Augmented Reality, IEEE Computer Society, Washington, DC, USA, 276.

Pulli K., Aarnio A., Miettinen V., Roimela K., and Vaarala J., 2008, Mobile 3D Graphics with OpenGL ES and M3G, Morgan Kaufmann Publishers, USA.

Roy, P., (2002), Direct3D vs. OpenGL: Which API to Use When, Where, and Why, http://www.gamedev.net/reference/articles/article1775.asp, last access: June 29, 2009.

Sanna A., Zunino C., AND LAMBERTI F., 2004, A Distributed Architecture for Searching, Retrieving and Visualizing Complex 3DModels on Personal Digital Assistants, International Journal of Human Computer Studies, Vol. 60, p. 701–716.

Thomson, R., (2009), Direct3D vs. OpenGL: A Comparison, http://www.xmission.com/~legalize/d3d-vs-opengl.html, last access: June 29, 2009.

Zimmermann O., Tomlinson M.R., Peuser S., 2004. Perspectives on Web Services: Applying SOAP, WSDL and UDDI to Real-World Projects. Springer-Verlag, Berlin Heidelberg New York.

Zunino C., Lamberti F., and Sanna A., 2003, A 3D Multi-resolution Rendering Engine for PDA devices, Proceedings of seventh world multi-conference on systemics, cybernetics and informatics (SCI03), Vol. 5, p. 538-542.