IEEE MELECON 2004, May 12-15, 2004, Dubrovnik, Croatia

3D Visualization of Data on Mobile Devices

Miran Mošmondor, Hrvoje Komerički and Igor S. Pandžić
Department of Telecommunications
Faculty of Electrical Engineering and Computing
University of Zagreb
Unska 3, HR-10000 Zagreb, Croatia
{miran.mosmondor, hrvoje.komericki, igor.pandzic}@fer.hr

Abstract — **This paper discusses current status and recent advancements of 3D graphics on mobile platforms and describes an application that is used for 3D visualization of data on such platforms. The implementation of a 3D visualization renderer on the Symbian platform for mobile devices is written as a C++ application and based on the DieselEngine as a rendering engine. 3D visualization of data is generated in the form of a VRML (Virtual Reality Modelling Language) file so this application can also be considered as a VRML browser. This means that actually any kind of 3D content written in VRML file format can be rendered on such a device. The implementation was tested on the Sony Ericsson P800 mobile device. It was the result of a project whose main objective was to provide a user interface on a mobile platform displaying visualization of hierarchical Grid monitoring data.**

I. Introduction

A. *3D visualization of data*

There are many ways in which to display data on a computer. Most often grids, forms, printed reports or charts are used. Quite new is a possibility to display data in three-dimensional space – as a 3D scene. Applications that visualize data in 3D are very useful analytical tools. They provide a wide view of a database and an interactive task of work. Unique views of data are available with 3D and VRML specification [3] features. 3D visualization of data has application in medicine, biology, physics, engineering, networking etc, and offers fascinating new opportunities in those areas.

Visualizing technical data in a 3D type environment can be beneficial to interpretation and in the discovery of new data relationships not easily apparent just from two-dimensional visualizations. Recent improvement of graphic workstations allows the frequent use of 3D visualization of information. 3D visualization has a great advantage of treating complex information as well as a large amount of data. State-of-the-art visualization techniques allow you to gain detailed insight into data. Graphics hardware support is utilized to display even very large data sets at interactive speed.

Our work concentrates on providing a user interface based on 3D graphics on a mobile platform. The developed application is designed for the Sony Ericsson P800 mobile device, and was in this particular implementation used to visualize Grid network monitoring data. Input for the application was provided by the MonALISA system [16], a distributed monitoring system used in this case to monitor AliEn Grid sites (http://alien.cern.ch/). Cone Trees [18] were used to create an appealing interactive 3D visualization for hierarchical data structure. The first prototype was implemented on desktop computers and an IPAQ PDA. It uses Shout3D technology to produce applet that runs on Java Virtual Machine of host device and generates 3D visualization from WRL file. In this way no plug-in is required. WRL is basically an enhanced VRML file. However, this technology is not possible on mobile devices because Java Virtual Machine on mobile platforms is not yet able to efficiently render this kind of contents.

B. *3D graphics on mobile platforms*

The last few years have seen dramatic improvements in how much computation and communication power can be packed into such a small device. Despite the big improvements, the mobile terminals are still clearly less capable than desktop computers in many ways. They run at a lower speed, the displays are smaller in size and have a lower resolution, there is less memory for running the programs and for storing them, and you can use the device for a shorter time because the battery will eventually run out.

Rendering 3D graphics on handheld devices is still a very complex task, because of the vast computational power required to achieve a usable performance. With the introduction of color displays and more powerful processors, mobile phones are becoming capable of rendering 3D graphics at interactive frame rates.

First attempts to implement 3D graphics accelerators on mobile phones have already been made. Mitsubishi Electric Corp. [11] announced their first 3D graphics LSI core for mobile phones called Z3D in March 2003. Also other manufacturers like Fuetrek [12], Sanshin Electric [13], Imagination Technologies [14] and ATI [15] published their 3D hardware solution for mobile devices a few months after.

Mobile terminals are already in many ways a match or even superiors to today's 3D gaming consoles. For example, in Figure 1 benchmark comparison of ARM's graphics acceleration solutions is shown. ARM's acceleration solutions are based around Imagination Technologies' PowerVR MBX cores.

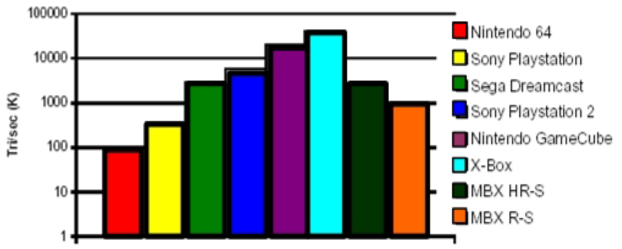

Figure 1. MBX benchmark comparison

We can see that performance of an ARM10 or ARM11 procesors with MBX HR-S clocked at or near its maximum clock rate is close to the Dreamcast and PlayStation2.

Beside hardware solutions, other important thing for 3D graphics on mobile devices is availability of open-standard, well-performing programming interfaces (APIs) that are supported by handset manufacturers, operators and developers alike. When we started working on this project no implementation of standard 3D APIs for mobile platforms were available. The reason is that OpenGL ES (OpenGL for Embedded Systems) [5] was published only recently and Mobile 3D API for J2ME (Java 2 Micro Edition) [6] is yet in final stage of standardization.

Sony Ericsson P800 [10] mobile device does not have any kind of hardware support for 3D graphics, but it is powerful enough to render 3D content without it. The idea was to build an application that will be able to produce 3D content based on WRL files. In this way maximum flexibility is enabled because application uses the same WRL file that is used to produce 3D visualization of data on desktop computers using Shout3D technology as mentioned before. So if there is a change in structure of 3D visualization of data only the module that produces WRL files has to be changed.

The first thing that had to be implemented was a WRL parser. To render 3D content on the P800 mobile device, the DieselEngine [7] was used. However, DieselEngine has low-level API and, among other things, the module for interaction with 3D objects also had to be implemented. This interaction regards navigation and selecting objects in the scene.

II. Required Tools

For developing a C++ application with DieselEngine on P800 mobile device the following SDKs (Software Development Kits) and tools were needed:

- Symbian UIQ v7.0 SDK [8]
- Metrowerks CodeWarrior for Symbian OS [9]
- DieselEngine SDK version 1.30 or higher
- P800 Connectivity tools

Symbian UIQ v7.0 SDK enables developers to write applications in C++ and/or Java. The SDK comprises development tools including the UIQ Emulator, selected source code, programming examples, documentation and other system components. The SDK enables application development for any UIQ based device. Programs are initially developed and then debugged on the UIQ Emulator (hosted on a Windows-based PC), and then rebuilt for a native platform and uploaded to a UIQ-based device. The SDK also includes deployment tools which allow developers to package applications for convenient delivery to end-users.

To run UIQ SDK for Symbian OS v7.0 Metrowerks CodeWarrior for Symbian OS is needed. The CodeWarrior IDE provides a set of tools for developing computer software. Using the IDE, program, plug-in, library, or other executable code can be developed to run on a wide variety of computer systems using different programming languages. The CodeWarrior IDE includes compilers, linkers, a source-code browser, a debugger, an editor, and a rapid application development tool set.

DieselEngine is software support for creating 3D applications on various platforms. It is a collection of C++ libraries that helps build applications with 3D content on mobile devices. However, DieselEngine has a low-level API (Application Program Interface) so high level modules had to be implemented.

III. Implementation

To dynamically generate 3D visualization of data on the P800 mobile device, a C++ application needed to be built. This application will be able to produce 3D content reading it from a WRL file. A module that generates the WRL file is already implemented for solution with Shout3D on desktop computers. The system architecture is shown in Figure 2.

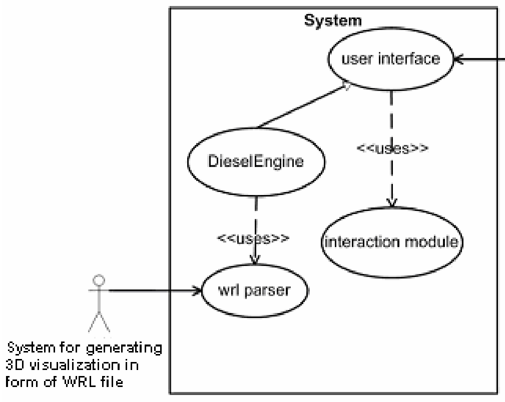

Figure 2. Use case diagram of solution

Main task of WRL parser is to convert 3D content from WRL file to Diesel3D scene format. In this way WRL parser is a bridge between high-level interface of WRL format and low level DieselEngine API. WRL parser is being summoned on the initialization of Diesel application.

The user interface enables input and output operations with the application. It consists of a button for exit, button for selecting navigation type, info panel and 3D visualization itself. It uses module for interaction to navigate and select objects in scene.

The module for interaction implements navigation and manipulation with the camera and enables selecting objects in the scene. There are three basic types of navigation: rolling, moving and zooming.

In the next chapter each of these modules is explained in detail.

A. Wrl Parser

The WRL parser is the most important part of application. It is the bridge between high-level interface of WRL format and low level DieselEngine API. Its main task is to covert 3D content from WRL file to Diesel3D scene format.

Building a complete WRL parser is a long process. However, for 3D visualization of data complete parser is not needed. We have implemented parser that enables rendering simple 3D content from WRL or VRML file. Beside support for primitive objects like box, sphere, cylinder and cone it has support for arbitrary shapes that are defined with *IndexedFaceSet* nodes. In addition, it is also capable of applying textures from jpg, bmp or gif file format. Even simple animation is enabled using *CoordinateInterpolator* and *OrientationInterpolator* nodes. Detailed list of supported nodes and fields is given bellow.

- root nodes: Transform (center, children, hidden, rotation, scale, translation), Group (children)
- children nodes: Background (skyColor), Shape (appearance, geometry), TouchSensor (enabled), Viewpoint (fieldOfView, orientation, position)
- other: Appearance (material, texture, textureTransform), Box (size), Cone (bottomRadius, height, side, bottom), Coordinate (point), Cylinder (bottom, height, radius, side, top), IndexedFaceSet (coord, coordIndex, texCoord, texCoordIndex, color, normal), Material (ambientIntensity, diffuseColor, emissiveColor, shininess, specularColor, transparency), Sphere (radius), Toggle (route), CoordinateInterpolator (key, keyValue), OrientationInterpolator (key, keyValue)

B. User Interface

The user interface enables input and output operations with the application. It consists of a button for exit, button for selecting navigation type, info panel and 3D visualization itself. (Figure 3)

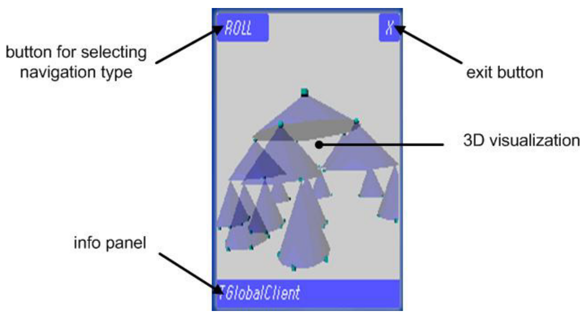

Figure 3. User interface

The user interface uses an interaction module for determining camera position and object selection.

C. Interaction

The module for interaction implements navigation and manipulation with the camera and enables selecting objects in the scene.

1) Navigation

For manipulation with the camera, basic matrix transforms were used. Using and combining these matrices navigation types *roll*, *move* and *zoom* were implemented (Figure 4).

Roll is a navigation type that rotates objects around x- and y-axis in a camera coordinate system. In this way a centre of a rotation is the centre of a camera coordinate system. However, position of an object does not have to be in a centre of a camera coordinate system, or size of the object can be changed. For example, when we pick a sensor another level of data is displayed. So, every time this happens, a new centre of rotation is being calculated and the translation is used to properly align a centre of rotation.

Move and *zoom* are navigation types that basically translate objects parallel to x-, y- and z-axis of a camera coordinate system.

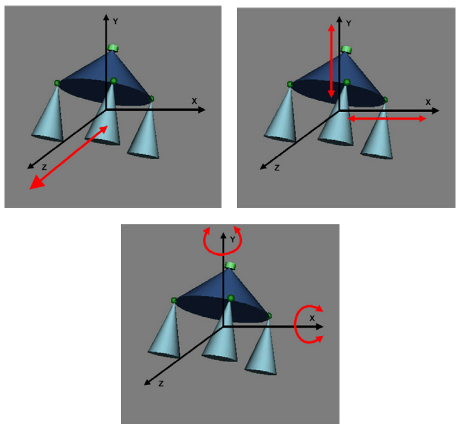

Figure 4. Navigation types: zoom, move, and roll

2) Picking

Picking or selecting objects in 3D scene with a device pointer is not as easy as it seems at first glance. There are several ways of selecting objects like ray casting, color coding, name lists etc [1]. We used a rendering method. Each of these methods test some kind of intersection with 3D objects. DieselEngine does not have any function for testing intersections so that has to be done manually.

In the rendering method each 3D object, or to be precise, every triangle of its face mesh is transformed into screen coordinates. So, intersection is tested in a 2D coordinate system. Testing intersection with each triangle of the object is time-consuming and to speed things up bounding volumes were used. In this case we used a bounding sphere. Instead of testing intersection with a couple of dozen triangles for each object, intersection is tested with the one bounding sphere what is very simple to implement using a rendering method. However, the bounding sphere formation itself is not as clear-cut because its centre and radius have to be determined in such a way that with minimum radius all vertices are inside of the bounding sphere. For primitive objects like sphere, box, cone or cylinder this is simple to determine. (Figure 5)

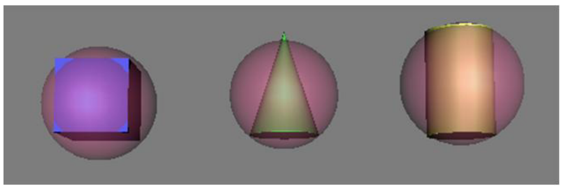

Figure 5. Approximation of primitive objects with sphere when testing intersection

For other complex objects or face meshes there are a number of algorithms that perform this task, and these have speed versus quality tradeoffs. We used the Ritter's algorithm [2] that creates a near-optimal bounding sphere. If there is a multiple intersection, a value in Z-buffer is compared and the nearest is selected.

IV. Case Study

We tried to provide a user interface on a mobile platform displaying visualization of hierarchical Grid monitoring data on the P800 mobile device. What is Grid is best presented in the next sentence: “Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed "autonomous" resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements.” [20].

Grid monitoring data is visualized using the 3D Cone Tree technique [19]. Cone Tree is an interactive visualization technique suitable for hierarchical structures. The user interface is enhanced by enabling interactive viewing, zooming, expanding and collapsing of parts of the structure. A formal user study using a cone-tree-based file system visualization showed that although Cone Trees are not suitable for all tasks, users “were enthusiastic about the cone tree visualization and felt it provided a better ‘feel’ for the structure of the information space”[18]. The root of the network hierarchy is located at the tip of a transparent cone. When a level in the hierarchy is expanded (on user click), its children nodes are distributed at equal distances around the base of a cone as shown on Figure 6.

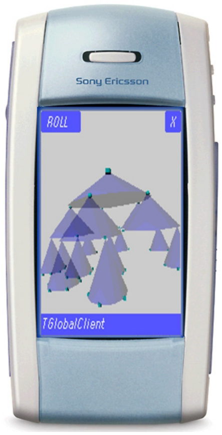

Figure 6. Cone tree displayed on P800

A. Results

For testing performances we expand our application with additional functions. First one measures time between frames and calculates frames per second (fps) parameter and the other one calculates and displays the number of triangles rendered in current frame. For Cone Tree structure tested on P800 mobile device, results showed that up to 15 000 triangles per second can be rendered. Although DieselEngine is 3D polygon-based renderer, performances don't depend only on the number of displayed polygons, but also on the number and type of objects in a scene. For example, we also tested the application with a face model and results showed up to 16 000 rendered triangles per second. The reason for this is that the Cone Tree structure consists of up to hundreds of objects, each with its own material, vertex buffer etc. so memory requirements are greater.

Nevertheless, with these performances simple 3D scene from WRL or VRML file can be rendered at acceptable frame rates.

V. Conclusion and Future Work

Rendering 3D graphics on handheld devices is still considered a formidable task. In this paper we presented a solution for 3D visualization of data on mobile devices and we showed that interactive frame rates can be achieved with large amount of displayed data. This solution can also be considered as a simple VRML browser so its application is much greater. Work is ongoing on the development of a facial animation player as defined in MPEG-4 Face and Body Animation (FBA) standard [4]. There, a simple VRML browser will be used to determine static geometry of the head model. Other applications include product visualization, multimedia presentations, entertainment and educational titles, and even shared virtual worlds. We can conclude that 3D graphics as well as 3D visualization of data will soon be as common on mobile devices as on desktop computers.

Acknowledgment

This work resulted as a part of FER/ETK Summer Camp 2003 Workshop.

References

[1] T. Akenine-Möller, E. Haines, “*Real-Time Rendering*”, A K Peters, Natick, Massachusetts, 2nd ed., 2002.

[2] J. Ritter, “*An Efficient Bounding Sphere*”, in A. S. Glassner, ed., *Graphics Gems*, Academic Press, pp. 301-303, 1990.

[3] ISO/IEC 14772-1:1997, Virtual Reality Modeling Language (VRML), http://www.web3d.org/technicalinfo/specifications/vrml97/

[4] ISO/IEC 14496 - MPEG-4 International Standard, Moving Picture Experts Group, http://www.cselt.it/mpeg

[5] OpenGL ES Overview, http://www.khronos.org/opengles/

[6] The Java Community Process(SM) Program - JSRs Java Specification Requests - detail JSR# 184, www.jcp.org/jsr/detail/184.jsp

[7] Inmar Software Ltd., DieselEngine SDK, http://www.inmarsoftware.com

[8] Symbian Developer Software Development Kits, http://www.symbian.com/developer/sdks.asp

[9] Metrowerks Home, http://www.metrowerks.com/mw/default.htm

[10] The Sony Ericsson P800, http://www.sonyericsson.com/P800/

[11] Mitsubishi Electric Corp., http://global.mitsubishielectric.com/

[12] Fuetrek , http://www.fuetrek.co.jp/english/home.html

[13] Sanshin Electric, GSHARK Technology, http://www.gshark.com

[14] Imagination Technologies, PowerVR, http://www.powervr.com/

[15] ATI Technologies Inc., http://www.ati.com/index.html

[16] H. B. Newman, I. C. Legrand, P. Galvez, R. Voicu, and C. Cirstoiu. MonALISA: A Distributed Monitoring Services Architecture. In *Proceedings of 2003 Conference for Computing in High Energy NuclearPhysics*, La Jolla, California, 8pp, 2003.

[17] OLIVE - On-line Library of Information Visualization Environments, http://www.otal.umd.edu/Olive/

[18] A. Cockburn, B.McKenzie, *An Evaluation of Cone Trees*, In People and Computers XV (Proceedings of the 2000 British Computer Society Conference on Human-Computer Interaction.) University of Sunderland, p.p. 425-436. Springer-Verlag, 2000.

[19] G. G. Robertson, J. D. Mackinlay, and S. K. Card, Cone trees: Animated 3D visualization of hierarchical information. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, p.p. 189–184, ACM Press, New York, USA. 1991.

[20] R. Buyya, “Gridbus Technologies for Service-Oriented Cluster and Grid Computing”, 2nd IEEE International Conference on Peer-to-Peer Computing, Linkopings, Sweden, September 2002