Telecommun Syst (2006) 32:181–191
DOI 10.1007/s11235-006-9137-3

3D Visualization on mobile devices

Miran Mosmondor · Hrvoje Komericki · Igor S. Pandzic

Abstract This paper discusses current status and recent advancements of 3D graphics on mobile platforms and describes open issues concerning its usage in different applications. We have treated two particular application fields. Firstly, we deal with problems of visualization of complex data structures on mobile devices. 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®[1] as a rendering engine. 3D visualization of data is generated in the form of a Virtual Reality Modelling Language (VRML) file meaning that actually any kind of 3D content written in VRML file format can be rendered on such a device. It was the result of a project the objective of which was to provide a user interface on a mobile platform displaying visualization of hierarchical Grid monitoring data. Secondly, we describe the system that brings face animation to embedded platforms. Face animation is considered to be one of the toughest tasks in computer animation today and its delivery to mobile platforms brings possibilities for development of new innovative and attractive services for the mobile market.

Keywords 3D graphics · Data visualization · Virtual characters · MPEG-4 FBA · Mobile device

1. Introduction

The last few years have seen dramatic improvements in how much computation and communication power can be packed into 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

M. Mosmondor (✉)
Ericsson Nikola Tesla, Krapinska 45, p.p. 93, HR-10 002 Zagreb, Croatia
miran.mosmondor@ericsson.com

H. Komericki · Igor S. Pandzic
Faculty of Electrical Engineering and Computing, Zagreb University, Unska 3, HR-10 000 Zagreb, Croatia
hrvoje.komericki, igor.pandzic@fer.hr

[1] DieselEngine®is registered trademark of the Inmar Software Ltd.

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. A small overview of today's mobile phones 3D graphics capabilities including software and hardware solutions is presented in the related work section.

A. 3D visualization of data

Most common definition of visualization is as "the use of computer supported, interactive, visual representations of data to amplify cognition" [3]. Authors in this work propose several major ways in which visualization can amplify cognition such as by increasing the memory and processing resources available to the users, reducing the search for information, using visual representations to enhance the detection of patterns, enabling perceptual inference operations, using perceptual attention mechanisms for monitoring and/or by encoding information in a manipulative medium. The term *visualization* was first used in 1987 in publication of the NSF report "Visualization in Scientific Computing" [5,9] but has come a long way since. There are two main domains of visualization that can be distinguished: scientific visualization and information visualization. Scientific visualization refers to visualization as a part of a process of scientific computing where it is used for computer modeling and simulation. This visualization is applied to physical data. On the other hand, information visualization refers to abstract data from other sources that are included in large and heterogeneous data collections found in business and finance, administration, digital media and other abstract concepts.

There are many ways to visualize such data on a computer. Different visualization layouts, like trees, maps, visualization rooms, and temporal dimension depend on type of visualized data. Previously mentioned physical and abstract data distinction is just one classification, but there are others like static and dynamic data, structured and unstructured data, or hierarchical and non-hierarchical data classification.

Applications that visualize data in 3D are very useful analytical tools. They provide a wide view of a database and an interactive way of work. Also, visualizing technical data in a 3D 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 large amount of data. Unique views of data are available with 3D and VRML specification features. 3D visualization of data has application in medicine, biology, physics, engineering, networking etc, and offers fascinating new opportunities in those areas.

Graphics hardware support is utilized to display even very large data sets at interactive speed. There are many visualization techniques that have been developed for PCs and workstations. However, using these same approaches for mobile devices introduces some unresolved problems.

B. Face Animation

Creating animated human faces using computer graphics techniques has been an increasingly popular research topic the last few decades, and such synthetic faces, or virtual humans, have recently reached a broader public through movies, computer games, and the World Wide Web. Current and future uses include a range of applications, such as human-computer interfaces, avatars, video communication, and virtual guides, sales persons, actors, and newsreaders.

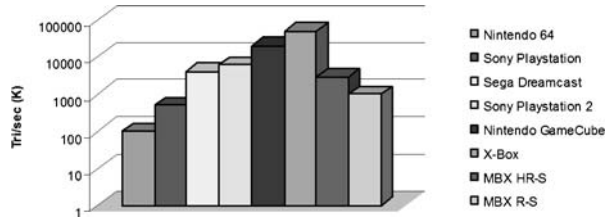

Fig. 1 MBX benchmark comparison[2]

In mobile applications, the face model can be animated using speech synthesis or audio analysis (lip synchronization) [14]. Face animation system described here is based on the MPEG-4 standard on Face and Body Animation (FBA) [13]. This standard specifies a set of Facial Animation Parameters (FAPs) used to control the animation of a face model. The FAPs are based on the study of minimal facial actions and are closely related to muscle actions. They represent a complete set of basic facial actions, and therefore allow the representation of most natural facial expressions. The lips are particularly well defined and it is possible to precisely define the inner and outer lip contour. Exaggerated values permit to define actions that are normally not possible for humans, but could be desirable for cartoon-like characters.

2. Related work

Today's mobile phones with color displays and more powerful processors are becoming more and more capable of presenting and rendering 3D graphics, with performances improving from one day to another. First attempts to implement 3D graphics accelerators on mobile phones have already been made. Mitsubishi Electric Corp. announced first 3D graphics LSI core for mobile phones called Z3D in March 2003. Since then numerous other manufacturers have introduced their own 3D graphics accelerators with performances attaining today's top 3D benchmarks employed for desktop computers.

For example, in Fig. 1 benchmark comparison of ARM's graphics acceleration solutions is shown. ARM's acceleration solutions are based around Imagination Technologies' PowerVR MBX cores.

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

There has also been a lot of research on computer graphics for mobile platforms that aim to reduce resource usage while maintaining a high quality of rendered image. Some advanced solutions have already been implemented in real graphic hardware. Example are Bitboys new graphics processors, called G32, G34, and G40, that have, among others, implemented new texture compression called PACKMAN (Ström and Akenine-Möller (2004)) targeted for mobile phones.

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. The efforts made by the Khronos Group were clearly some of the biggest steps toward this direction. The Khronos Group is the industry consortium founded by a number of leading media-centric companies in January 2000. One of open standard APIs they have created is the OpenGL ES (OpenGL for Embedded Systems)

[2] Source: Ashley Stevens, *ARM 3D Graphics Solutions* (http://www.arm.com/pdfs/MBX%203D.pdf)

announced in July 2003. It is a royalty-free, cross-platform low-level API for full-function 2D and 3D graphics on embedded systems.

Another important effort was Mobile 3D Graphics API (JSR-184) specification developed under the Java Community Process. It was published by the end of October 2003. JSR-184 (also known as M3G) is a high-level API for Java mobile 3D graphics and it is designed to provide an efficient 3D graphics API suitable for the J2ME platform, and, in particular, for the CLDC/MIDP profile. OpenGL ES low-level API and M3G high-level API represent de facto industry standards in area of 3D graphics API for mobile devices.

In the field of scientific research on visualization, there has been a lot of work. As mentioned, our project concentrated on providing a user interface on a mobile platform displaying visualization of hierarchical Grid monitoring data [19]. Visual interfaces for information retrieval are a growing research area that attempts to provide visual depictions of very large information spaces. However, almost all of the visualization approaches have been developed for PCs and workstations.

[18] present the Multiplatform Universal Visualization Architecture (MUVA), a collection of software modules that allow visualization of the same data across a wide range of platforms, from workstations to mobile phones, while automatically adapting the visualization and delivery modes to the particular platform. Collaboration in the process of MUVA creation was the primary motivation for our prototype application development.

[7] present one of the few works that deal with the problem of the visual representation of intrinsic structures of complex information spaces on mobile handhelds. They have described two techniques for hierarchy visualization, called *Magic Eye Vi*ew and *Rectangle View* that were adopted for these devices. In both methods a 2D approach was used although the original *Magic Eye View* technique uses 3D *Focus & Context* visualization. In their opinion, real interactive 3D visualization exceeded the capabilities of pocket-sized devices so they had customized this technique in order to use it on mobile handhelds.

We will show that 3D visualization with interactive frame rates can be achieved with large amounts of displayed data. Furthermore, rich interaction functionalities such as 3D transformations (zoom, rotation or translation), folding and unfolding subtrees, changing parameter values etc. are especially convenient for such handheld devices with small display screen and thus are implemented in our system, despite the fact that this was intentionally neglected in above mentioned Karstens, Kreuseler, and Schumann (2003).

Regarding face animation system on mobile device, no previous similar research work in this area was found.

3. Implementation of 3D visualization

To dynamically generate 3D visualization of data on mobile devices, a prototype application was built [11]. This application was able to produce and efficiently render 3D content converting from an extended VRML format. The application high-level UML Use Case diagram is shown in Fig. 2.

As it can be seen in the diagram, how data is generated and collected is outside of this system scope. It is only important that is in form of the VRML file. After VRML parsing, this 3D scene is optimized for representation on mobile device. Limited processor power, memory constraint and small display size of such devices are all taken under consideration when performing this optimization. In this way, for example, picking sensors are presented as 2D object, while the data structure itself remains in 3D format. Afterwards, user can manipulate with this optimized 3D

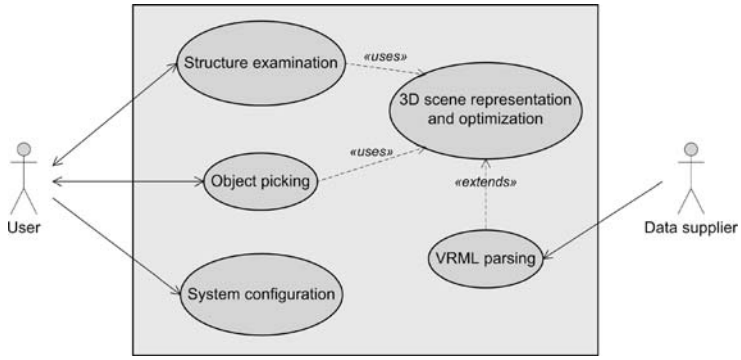

Fig. 2 Application UML Use Case diagram

scene through its user interface. It can perform structure examination, object picking, or it can configure some specific system parameters.

From this high level description of the system several important system functionalities could be identified. First is converting 3D content from extended VRML format to DieselEngine®3D scene format in VRML parser. This VRML parser connects high-level interface of VRML format and low-level DieselEngine®API.

Second functionality that needed to be implemented was interaction. The module for interaction implements navigation and manipulation with the camera functionalities and enables selecting objects in the scene. There are three basic types of navigation: rolling, moving and zooming.

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

A. VRML Parser

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

Building a complete VRML parser is a long process. However, for 3D visualization of data complete parser is not needed. We have implemented a parser that enables rendering simple 3D content from VRML file. Beside support for primitive objects like box, sphere, cylinder and cone it has support for arbitrary 3D mesh shapes that are defined with *IndexedFaceSet* VRML 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 in Table 1.

B. 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 (Fig. 3).

Table 1 Supported nodes and affiliated fields list compliant to VRML97 specification

Node type	Node name	Supported fields
Grouping	Group	children
	Transform	center, children, hidden, rotation, scale, translation
Common	Shape	appearance, geometry
Sensor	TouchSensor	enabled
Geometry	Box	size
	Cone	bottomRadius, height, side, bottom
	Cylinder	bottom, height, radius, side, top
	IndexedFaceSet	coord, coordIndex, texCoord, texCoordIndex, color, normal
	Sphere	radius
Geometric Properties	Color	color
	Coordinate	point
Appearance	Appearance	material, texture, textureTransform
	Material	ambientIntensity, diffuseColor, emissiveColor, shininess, specularColor, transparency
Interpolators	CoordinateInterpolator	key, keyValue
	OrientationInterpolator	key, keyValue
Bindable	Background	skyColor
	Viewpoint	fieldOfView, orientation, position

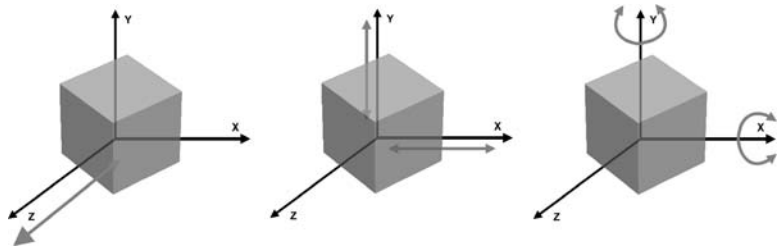

Fig. 3 Navigation types: zoom, move, and roll

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.

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.

We used a rendering method (Akenine-Möller and Haines (2002)). Each of these methods tests 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.

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 (Ritter (1990)) that creates a near-optimal bounding sphere. If there is a multiple intersection, a value in Z-buffer is compared and the nearest point is selected.

C. Case study

Our work concentrated on providing a user interface on a mobile platform displaying visualization of hierarchical Grid monitoring data with 3D Cone Tree technique [16]. Cone Tree is an interactive visualization technique suitable for hierarchical structures, also known as tree structures. Tree layout is an innovative graphic which can be two or three-dimensional and is especially used for structured and hierarchical data. Hierarchies are collections of data nodes where each node has a unique parent (node above it in the hierarchy), but may have many siblings (nodes below it in the hierarchy). In general, the nodes and the links between them can have multiple attributes. The entire structure of the hierarchy and its encompassing relations are also usually relevant. Tasks can be applied to a single node, a link, a collection of nodes, or even to the entire structure. A hyperbolic view can be used to centre the interesting node.

Enabling interactive viewing, zooming, expanding and collapsing of parts of the structure enhanced the user interface. 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" [4]. 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 Fig. 4.

The developed application is designed for the mobile devices with Symbian operating system, and was in this particular implementation used to visualize Grid network monitoring data. 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." (Buyya (2002)). Input for the application was provided by the MonALISA system (Newman et al. (2003)), a distributed monitoring system used in this case to monitor AliEn Grid sites (http://alien.cern.ch/).

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 extended VRML (.wrl) file. In this way no plug-in is required. WRL is basically an enhanced VRML file. However, this technology was not possible on mobile devices

Fig. 4 Cone tree displayed on SE P800

because Java Virtual Machine on mobile platforms was not able to efficiently render this kind of contents.

4. Implementation of face animation player

The player is essentially an MPEG-4 FBA decoder. When the MPEG-4 Face Animation Parameters (FAPs) are decoded, the player needs to apply them to a face model. Our choice for the facial animation method is interpolation from key positions, essentially the same as the morph target approach widely used in computer animation and the MPEG-4 FAT approach. Interpolation was probably the earliest approach to facial animation and it has been used extensively. We prefer it to procedural approaches and the more complex muscle-based models because it is very simple to implement, and therefore easy to port to various platforms; it is modest in CPU time consumption; and the usage of key positions (morph targets) is close to the methodology used by computer animators and should be easily adopted by this community.

The way the player works is the following. Each FAP (both low- and high-level) is defined as a key position of the face, or morph target. Each morph target is described by the relative position of each vertex with respect to its position in the neutral face, as well as the relative rotation and translation of each transform node in the scene graph of the face. The morph target is defined for a particular value of the FAP. The position of vertices and transforms for other values of the FAP are then interpolated from the neutral face and the morph target. This can easily be extended to include several morph targets for each FAP and use a piecewise linear interpolation function, like the FAT approach defines. However, current implementations show simple linear interpolation to be sufficient in all situations encountered so far. The vertex and transform movements of the low-level FAPs are added together to produce final facial animation frames. In case of high-level FAPs, the movements are blended by averaging, rather than added together.

Due to its simplicity and low requirements, the face animation player is easy to implement on a variety of platforms using various programming languages. The face animation player on Symbian platform (Figure 5) for mobile devices is based on DieselEngine®. Because of low CPU time consumption and low memory requirements, MPEG-4 FBA decoder can be used on mobile device. Most important issues concerned rendering 3D graphics on mobile device. For that purpose DieselEngine®was used. It is a collection of C++ libraries that helps building applications with 3D content on various devices. DieselEngine®has low-level API

Fig. 5 3D face model preview on Symbian platform

(Application Program Interface) that is similar to Microsoft DirectX and high level modules had to be implemented. The most important is VRML parser that is used to convert 3D animatable face model from VRML format to Diesel3D scene format (DSC). Other modules enable interaction with face model like navigation, picking and centering.

Appliances of such face animation player are numerous. For example, it is crucial part of a prototype system that allows mobile subscribers to communicate using personalized 3D face models created from images taken by their phone cameras [10].

5. Results

For testing performances we expand our prototypes 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 several thousands triangles per second can be rendered. Although DieselEngine®is 3D polygon-based renderer, performances do not 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 single sample object and results showed couple of hundred more 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.

We have also tested face animation application on Sony Ericsson P800 mobile device with various face models as well. Interactive frame rates were achieved with models containing up to several hundreds polygons. Generic face model that is used in our prototype system uses approximately two hundred polygons and its performances are shown in Fig. 6. After animation has started, in this case in 21st second, frame rate drops to average of 10 frames per second (FPS), but this is still relatively higher than the considered minimum for interactive frame rates on mobile devices.

6. 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 could be achieved with relatively large amount of displayed data. This

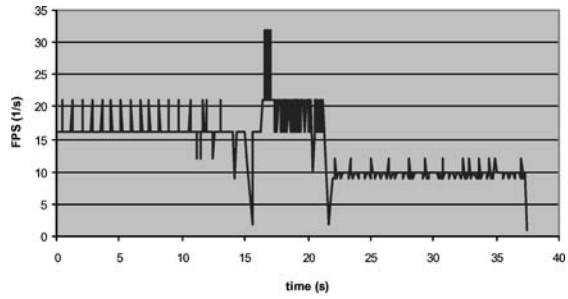

Fig. 6 Generic face model performances on SE P800 mobile device

solution can also be considered as a simple VRML browser so its application is much greater. We have gone a step further and developed a facial animation player as defined in MPEG-4 Face and Body Animation (FBA) standard. There, a simple VRML browser was 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 see how 3D graphics as well as 3D visualization of data is becoming more and more as common on mobile devices as on desktop computers.

Acknowledgment This work resulted as a part of FER/ETK Summer Camp 2003 Workshop. The facial animation aspects of the work are done in collaboration with Visage Technologies AB, Linköping, Sweden.

References

1. Akenine-Möller, Tomas, Haines, Eric, *Real-Time Rendering*. A K Peters, Natick, Massachusetts, 2^{nd} ed (2002).
2. Buyya, Rajkumar, *Gridbus Technologies for Service-Oriented Cluster and Grid Computing*. Keynote Presentation at 2nd IEEE International Conference on Peer-to-Peer Computing, (Linkopings Sweden, 2002).
3. Card, K. Stuart, Mackinlay, D. Jock, Schneiderman, Ben, *Readings in information visualization: Using vision to think*. 1999 MORGAN Kaufmann publisher Inc, (San Francisco California, 1999).
4. Cockburn, Andy, McKenzie, Bruce, *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, published by Springer-Verlag, (2000) pp. 425–436.
5. DeFanti, A. Thomas, Brown, D. Maxine McCormick, H. Bruce. *Visualization: Expandig scientific and engineering research opportunities*. IEEE Computer, 22(8) (1989), pp. 12–25.
6. Escher, Marc, Pandzic, S. Igor, Magnenat-Thalmann, Nadia, *Facial Deformations for MPEG-4*. Proceedings of Computer Animation 98 IEEE Computer Society Press, (Philadelphia, 1998) pp. 138–145 USA.
7. Karstens, Bernd, Kreuseler, Matthias, Schumann, Heidrun, *Visualization of complex structures on mobile handhelds*. Proceedings of IMC'2003, International Workshop on Mobile Computing, 8pp, Rostock, Germany, (2003).
8. Lavagetto, Fabio, Pockaj, Roberto, *The Facial Animation Engine: towards a high-level interface for the design of MPEG-4 compliant animated faces*. IEEE Trans. on Circuits and Systems for Video Technology, 9(2)(1999) pp. 277–289.
9. McCormick, H. Bruce, DeFanti, A. Thomas, Brown, D. Maxine. (eds.), *Visualization in scientific computing*. Computer Graphics, 21(6) (1987) (entire issue).
10. Mosmondor, Miran, Kosutic, Tomislav, Pandzic, S. Igor, *LiveMail: Personalized Avatars for Mobile Entertainment*, The Third International Conference on Mobile Systems, Applications, and Services, Seattle, (USA, 2005) pp. 15–24.
11. Mosmondor, Miran, Komericki, Hrvoje, Pandzic, S. Igor, *3D Visualization of Data on Mobile Devices*. The 12th IEEE Mediterranean Electrotechnical Conference – MELECON 2004, (Dubrovnik, Croatia, 2004) pp. 645–648.
12. Newman, B. Harvey, et al, *MonALISA: A Distributed Monitoring Services Architecture*. Proceedings of 2003 Conference for Computing in High-Energy and Nuclear Physics (CHEP 03) on DVD, Jun 2003, CHEP-2003-MOET001, 8pp, La Jolla, (California, 2003).

13. Pandzic, S. Igor, Forschheimer Robert. (eds.), *MPEG-4 Facial Animation – The standard, implementations and applications*, John Wiley & Sons (2002).
14. Pelachaud, Catherine, Badler, I. Norman, Steedman, Mark, *Generating Facial Expressions for Speech*, Cognitive Science, 20(1) (1996), pp. 1–46.
15. Ritter, J., *An efficient bounding sphere*. In Graphics Gems, Andrew S. Glassner, Ed., chapter V, pages 301–303. Academic Press, (San Diego, CA, 1990).
16. Robertson, G. George, Mackinlay, D. Jock, Card, K. Stuart, *Cone trees: Animated 3D visualization of hierarchical information*. Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology, pp. 189–184, ACM Press, (New York, USA, 1991).
17. Ström, Jacob, Akenine-Möller, Tomas, *PACKMAN: Texture Compression for Mobile Phones*, Poster at SIGGRAPH 2004.
18. Skorin-Kapov, Lea, et al., *Multiplatform Universal Visualization Architecture*, The Second International Conference on Mobile Multimedia, (Bali – Indonesia, 2004) pp. 15–24.
19. Skorin-Kapov, Lea, et al., *Interactive Visualization of Grid Monitoring Data on Multiple Client Platforms*, European Grid Conference, (Amsterdam – The Netherlands, 2005).