Association for Information Systems
AIS Electronic Library (AISeL)

Wirtschaftsinformatik Proceedings 2009 | Wirtschaftsinformatik

2009

AN INTERACTIVE REMOTE VISUALIZATION SYSTEM FOR MOBILE APPLICATION ACCESS

Marcus Hoffmann
Fraunhofer IGD, Darmstadt

Jörn Kohlhammer
Fraunhofer IGD, Darmstadt

AN INTERACTIVE REMOTE VISUALIZATION SYSTEM FOR MOBILE APPLICATION ACCESS

Marcus Hoffmann, Jörn Kohlhammer[1]

Abstract
This paper introduces a remote visualization approach that enables the visualization of data sets on mobile devices or in web environments. With this approach the necessary computing power can be outsourced to a server environment. The developed system allows the rendering of 2D and 3D graphics on mobile phones or web browsers with high quality independent of the size of the original data set. Compared to known terminal server or other proprietary remote systems our approach offers a very simple way to integrate with a large variety of applications which makes it useful for real-life application scenarios in business processes.

1. Introduction

Looking forward to the growing mobilization of the work force and data access, one focus of visualization these days is to use known visualization techniques on mobile devices. The main challenge in mobile computing is limitation of limited hardware resources. At the same time the data transfer rates have grown in the recent years. It is now possible to establish peer-to-peer data connections between mobile and portable devices and stationary hardware. Data flat rates for mobile access using e.g. WiFi or 3G wireless networks become more and more affordable. Looking further at this evolution, we expect the usage and acceptance of streaming data to mobile devices to grow in the coming years. Terminal server applications and proprietary remote visualization solutions tried to address these problems in the past. Nevertheless these technologies aim for specific solutions and have constraints regarding the flexible integration and functionality porting to the mobile device.

This paper presents a system that allows the user to interactively use high quality 2D and 3D graphical content with slim clients like smartphones or PDAs. The visualization system is designed to connect to a variety of applications on the server side and transmit the pre-calculated visualization data to a mobile client where the user can interact with the server-sided application in real-time. The application can use all hardware provided by the server. The mobile client gets visualization results of highest possible quality regarding to what the server application can achieve. This objective can be fulfilled through a streaming connection between the mobile client and the application located on the visualization server. Except the addressing of limited hardware possibilities of mobile devices another benefit of a remote system that has to be taken into account

[1] Fraunhofer IGD, Darmstadt, Germany

is a security issue. Transmitting pre-calculated and rendered data to a mobile device without sending the real application data is a big advantage looking at secure data transmission for applications handling confidential data on the server side. The data itself is safe behind firewalls and only used by the application on the server side. The only data sent to the client is the remote image data.

2. Related Work

2.1. General approaches

Enabling visualization on mobile devices has been a major research topic in recent years. The main goal of the research in this area was to achieve interactivity on slim devices like Smartphones or PocketPCs with very limited hardware properties regarding complex calculations. Generally there are two ways to achieve such visualizations.

The first approach aims at the development of toolkits and algorithms to provide the mobile device with highly compressed and optimized data for visualization and a framework running on the mobile device itself to render the received data sets. Here, Pulli et al. [13] or Soetebier et al. [19] have achieved very interesting results presenting a framework for mobile 3D visualization with OpenGL ES [12] and M3G [14]. With the approach in M-Loma [9] has targeted the area of geographic visualization to bring 3D city maps and landscapes to mobile phones and achieved excellent results in the field of rendering on mobile devices.

The second major approach is to outsource the calculations for visualization to a server and transmit the completed results in form of images or video streams to the thin client. Engel et al. [4], B.O. Schneider and I. M. Martin [17] or SGI's VizServer [18] have shown methods how to visualize complex 3D data on mobile devices with the focus on special applications or systems a few years ago. Other approaches described by Teler and Lischinski [22] or Soumyajit et al. [20] especially in the area of 3D remote rendering use distributed image-based transmission of parts of the 3D scene and re-arrange the images and imposters on the client-side to complete the 3D scene with pre-rendered images from the visualization server. However, these approaches target either special data structures and data sets or special applications where they integrate their remote functionality. Our approach aims for a more generalized solution in the remote visualization area to be able to use one system for a number of different applications instead of implementing specialized and proprietary solutions for each different application. Basically it does not require a specific application or server solution to work with. The application only needs the capability to capture the content that is targeted for remote visualization and an interface to listen to incoming interaction events. For some development frameworks (QT [15], MFC [10], Java [6]) this functionality is provided within a plug-in library, for others generalized methods to input the captured image data into the plug-in for further processing and transmission to the mobile client are provided.

2.2. Yet another Remote Desktop Application?

A variety of virtual network computing (VNC) applications [16] [24] was developed in the recent years to gain remote workers access to workstations from everywhere. For every popular operating system different applications are available such as the Microsoft Windows remote desktop [10] or RealVNC [24] for windows and Linux operating systems. The VNC applications provide access to the complete desktop of the remote workstation. This paper describes a more sophisticated approach. Basically, our system will not give the mobile user access to the complete desktop with

all its functionality and access to critical system parts. Only the relevant parts of the application will be visualized for the transmission to the client-side and only these parts are available for interaction. Which parts this will be can be defined by the administrator that integrates the application with the remote system. Another issue with VNC visualization is included by the use of OpenGL or other 3D content visualization on the server environment for different reasons. Either they gain access to different users at the same time but having problems with hardware accelerated visualization technologies or they gain access to one remote user per session being able to use hardware features for the remote visualization. The reason for the problems in hardware acceleration for different users at the same time lies in the fact that such solutions make use of software emulated graphics adapters. These emulators do typically not support hardware accelerated features as needed for e.g. OpenGL. Terminal server solutions are some kind of similar to VNC solutions but terminal server solutions are optimized for special programs on the server and the client side. Furthermore real application data is transmitted between the client and server side which makes the application vulnerable for hacking or spy attacks. Since the only data this solution transmits is image data from the server to the client and proprietary interaction data from the client back to the server which only can decode the remote server we have achieved to have a very robust solution looking at security issues.

3. System Concept

The basic concept of this visualization system is the outsourcing of all calculations necessary for the visualization to a server environment. Instead of transmitting the data to the mobile client and doing the visualization calculations on the mobile device the visualization application is running on the server. The thin client only has to render 2D image data and capture interaction from the client user. The interaction data is then transmitted back to the server environment for post processing and adaptation of the current visualized scenes. *Figure 1* shows the basic communication pipeline between the application itself and the mobile clients on the other side using the render server in between. On the server side, at least two programs are running. The first one is the *application* itself which contains all the visualization functionality and technologies which has to be provided for the mobile client. Since the application is running on the server side of the overall system it can make use of all hardware advantages the server provides. Therefore very complex calculations and visualizations can be done using the latest hardware features on the server side.

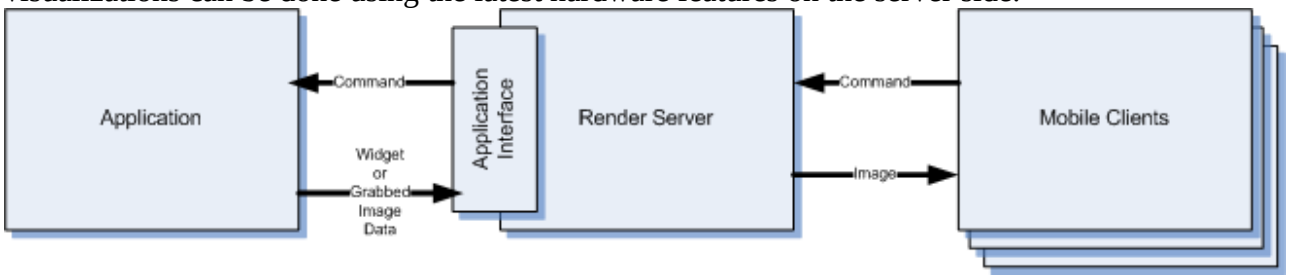

Figure 1: The complete communication pipeline.

3.1. Render server

The render server component consists of three major parts: the *network interface*, the *image processing engine* and the *application interface*. Basically, the system provides a one to one connection from the mobile client to the remote application. However, more than one mobile client will be able to connect to the same remote enabled application. All clients connected to one instance of the application will see the same visualization and each mobile client will see the same interaction and remote steering from all connected devices.

To provide different views to different users it is possible to start multiple instances of the application on the server side and connect each different mobile participant to a different instance of the remote enabled application. Then, every mobile user will have its own visualization not shared with others and can interact with it independently. *Figure 2* shows the render server's architecture, its components and how the components communicate with each other.

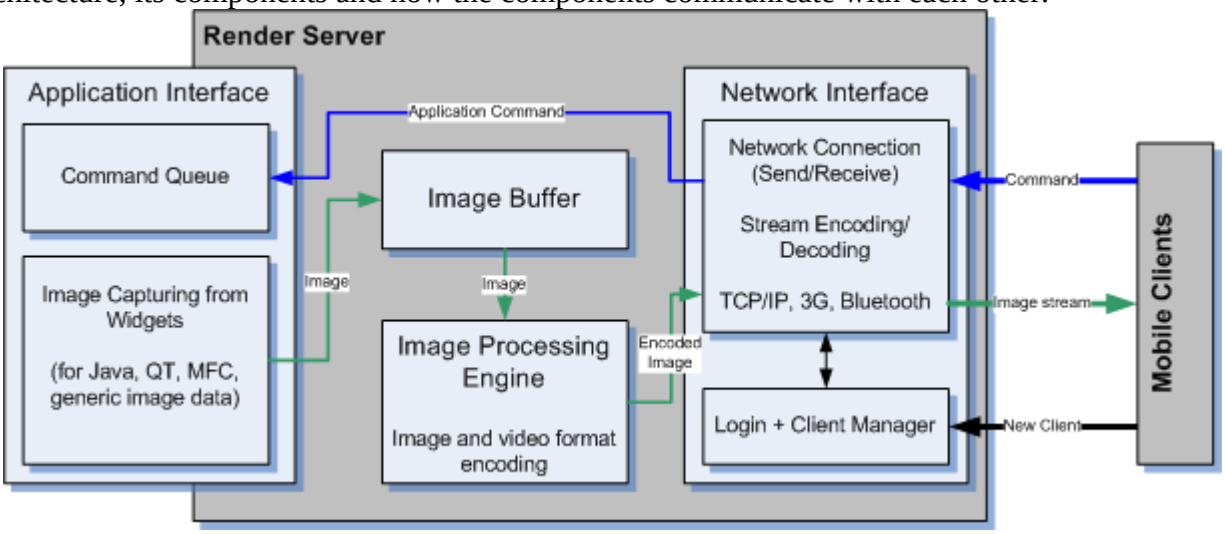

Figure 2: The render server system layout

3.1.1. Network Interface

The network interface handles the network communication between the render server and the mobile client. At its current state it supports two options: Communication over TCP/IP for LAN, wireless LAN or 3G connections and the communication over Bluetooth for short distance connections. Both protocols are integrated into the mobile client application and the render server and can be used. The *first part* of the network interface is the client manager. It provides a server socket that is listening for incoming connection requests of mobile clients. It handles the connection procedure and the management of multiple clients. This management includes the correct routing of the different clients to their according application instances for remote visualization. The *second part* is the worker area for connections to the different clients. This part of the network interface takes care of the creation of the network stream from the compressed image data incoming from the image processor. The network connection to the client runs in its own thread which grabs the image data from a buffer provided by the image processing engine every time the client asks for a new image. Another thread is running in the network interface for incoming interaction request. Here, all commands incoming from the mobile client are decoded and sent back to the application interface. The application then grabs the commands from the interface. Every application can then interpret the commands individually.

3.1.2. Image processing engine

In this engine the encoding of the captured raw image data into a stream able image or video format is done. Furthermore the capture functionality for selected application frameworks and technologies is integrated here. The image processing engines provides a number of different algorithms used to compress images like JPEG, J2K, PNG, GIF, Motion JPEG etc. Chapter 4 discusses in detail the utilization of image-based transfer in comparison to video streaming. The image processing engine can be extended by additional compression algorithms. The engine provides the current state of the visualization within a buffer. This buffer is updated for each change of the content in the application. Then, a new image will be encoded and stored into the buffer. From that buffer, the

network interface will grab the image and encode it into a network stream that is sent to the mobile client.

3.1.3. Application interface and integration

This is the interface for the visualization application to communicate with the render server or the mobile client on the other side of the network pipeline respectively. The application has to use this interface to the render server because it provides the application with all functionality that is necessary for the remote connection. This basic interface makes it is very simple to connect new applications to the render server functionality. The render server's application interface consists of two parts that must be connected with the application:

The first part is the *capturing functionality*. Here, the application makes use of algorithms integrated into the render server interface to capture either the complete application window or only certain parts of the application. For a number of popular development frameworks like Java, QT, MFC or OpenGL context widgets image grabbers are integrated into the application framework of the render server to alleviate the integration of the remote system for the user. The interface is designed in a way that it can be extended with more functionality regarding the grabbing of content from an application. The application can also decide to capture the content by itself and supply only raw image data. In this case, the image processing engine will be provided with the captured raw image data for further processing. The other facility of the application interface is a *command interface* which is triggered every time the mobile client sends an interaction request. This request is only an undetermined command coming from the mobile client and routed through the render server to the application interface. In the current version, the application does the command message handling, interpret the command and trigger the according functionality inside. In summary, it can be stated that the effort to integrate the remote visualization functionality into existing applications is kept very low. The developer only has to decide which part needs to be exposed to remote users and let the interface capture this part.

3.2. Mobile client

The mobile client was realized using the Java Micro Edition (JME) [5] to maximize the compatibility on different devices. Almost all currently available mobile devices are able to interpret JME applets. Furthermore, such applets can easily be integrated into a website. Hence, one mobile application can be used to connect to a large number of different remote applications using a large number of different mobile devices. This was one of the main goals of our work – to be able to have a generalized solution for mobile visualization without having too much effort to adapt every new application to the clients or, even worse, to the device it will be used on. Simply spoken, the mobile client does nothing else than rendering 2D images to its screen device and sending back interaction commands to the render server using the Java API. More specifically, the mobile client contains the following components: The *network component*, the *user interface* and the *mobile control* component. An architectural overview of the mobile client is illustrated in *Figure 3*. The *network component* contains the functionality to connect to the render server, receive the image stream and send out interaction and acknowledge commands to the render server. The send and receive functionality is running in different threads to be able to asynchronously send out commands to the visualization application on the server side. The *mobile control* part of the client captures all events coming from the mobile device and converts them into network commands. These commands are sent to the server asynchronously. Currently, the set of commands that can be sent from the client to the server is pre-defined. The *user interface* can be designed independently from the application on the server side. All the client need is an area for 2D image rendering for the

remote data. Additional interaction functionality can be integrated using interface elements like menus or buttons. The render server can capture the complete application on the server side and visualize it on the mobile client. In this case it is sufficient to capture mouse and key events for steering the application.

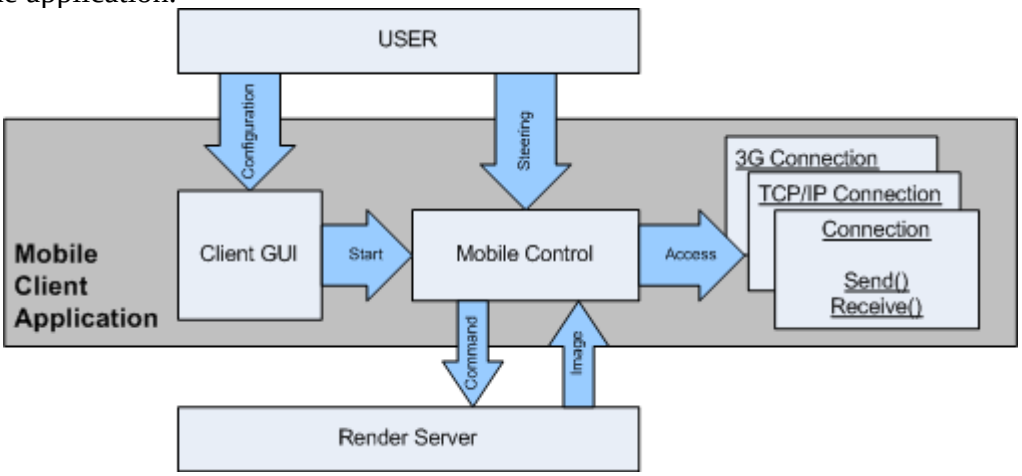

Figure 3: Mobile client architecture

Each time the mobile client has finished the rendering of a 2D image it sends out a flag to the render server for synchronization. This flag signals the render server to send the next image or pack of multiple images. This is necessary to keep the mobile application synchronized with the application connected with the render server. Since the render server can encode and send images very fast, the bottleneck of the whole system lies on the mobile client side. Most of the mobile clients have either very limited hardware resources or only access to bandwidth-limited networks. Taking these conditions into account the render server should never send more image data then necessary. To prevent the server from producing unnecessary network traffic it waits for the flag from the client to update the image stream.

3.3. Security issues

Most of the applications to be used with the remote clients are located behind firewalls. Therefore, it is necessary to establish a communication instance in front of the firewall to be able to establish a secure connection to the render server plug-in of the application in the back of the firewall. For that reason a small Java application was developed which is running on a server in the so-called demilitarized zone (DMZ). This server is running in front of the firewall and will be able to accept most incoming connection and communication attempts, but it is not able to send data through the firewall to the secure area. In the secure area another server is running. An implicit advantage of remote visualization systems in general is the fact that only image data is transmitted. Even if an attacker would hijack the communication between the client and the render server he would only get rendered images from the application on the server side. The raw – and possibly confidential – data is located behind a firewall on a server that can not be accessed. Since the communication protocol uses proprietary commands that can be defined by the user it is very hard to tunnel the firewall using our remote protocol.

4. Discussion

This chapter will discuss our approach to stream interactive visualization data from the client to the server. First of all, the remote client gets its data image by image. This means that the server grabs

the content and saves every image into a buffer. From the buffer a server thread creates a network command that will be transmitted to the remote client. An obvious alternative that comes to mind is to use a video stream. There are three reasons why we do not use a video stream. The first thing to take into account is the ability of the client to *decode data*. Decoding of video streams requires CPU power that is a scarce resource on mobile hardware. Especially taking into account that the device does not only have to listen to the video stream but furthermore has to mange interaction commands from the user makes such a solution even less useful. The second issue is the ability of the server to *encode video streams in real-time*. We have tested several encoders for different video streams like e.g. MJpeg or Mpeg4 [8]. These tests clearly showed that a standard computer will not be able to encode a video stream in real-time from an image chain provided frame-by-frame The best result reached frame rates of around 2 to 5 when using software encoders and the latest standard CPU hardware available. Even the use of multi-core CPUs does not give any advantages to the frame rate that can be provided by the server, because the image compression for a network stream can not be parallelized to be more efficient. The third point to take into account is mainly a *synchronization* issue and this is the major reason for deciding against video stream based visualization. Video streams with high compression rates typically consist of enclosed sequences of images. Within these sequences, single images are compressed using techniques that differ from format to format. But generally on the decoder side these sequence blocks must be decoded at once because the single images inside the sequence have certain dependencies on their parents. However, the current sequence has to be completely processed by the client before the client can react to the latest interaction. Therefore the user would have the experience that the remote visualization does not react immediately to its interaction requests. Especially when navigating in 3D scenes with fluent animations it makes a huge difference if a camera movement appears in real-time to the user when he started it or if the user has to wait a second until his interaction appears on the remote device. In summary, for our application scenario an on demand single image transmission is the most adequate solution.

We have tested a variety of compression formats like jpeg (with different quality settings from 30% up to 100% quality), png, bmp or zip on a dual core processor machine. Here, the jpeg and png compression have proved to be the most useful for the proposed system. Most mobile devices are able to decompress jpeg or png using their hardware resources which makes the decompression procedure very fast on the client side. On the server side png requires more CPU resources than jpeg but none of these two compression methods takes the CPU to its limits when not exceeding image sizes of around 1024x768 pixels. This scalability issue is obviously a minor problem on smartphones or very slim devices with small displays but has to be considered when talking about remote solutions for web environments. Approaches to address especially the support of large solutions are described in chapter 6 in the future work part.

The hardware requirements on the server side for the support of a large number of simultaneous users are the biggest scalability challenge. After all, the more independent visualizations the user of such a system wants to provide, the more instances of the application have to run on the server simultaneously and the more hardware performance is required to achieve reasonable frame rates. This is, of course, hardly a new problem, but rather a well-known issue that has been researched intensively by the network and service management community [1]. Depending on the application, the number of clients and the request rate it is possible to find the most cost-effective solutions. One major scenario for our application is to connect multiple remote clients to the same visualization. In that scenario only one application runs on the server side and multiple users are connected to that single server application. Since the render server buffers the captured images, this buffer can be used to broadcast the current image to every connected client, which causes only low additional server load.

5. Application Areas

Due to the flexibility of the introduced system, the fields of application are numerous. The generic client implementation and the usage of the JME platform allow the client application to be ported to a variety of mobile devices as well as used in a web browser. The simplicity and flexibility of the render server interface allows the integration of the technology into a large variety of applications in the computer graphics area.

Figure 4: The application FinMotion and a game client with captured graphics content on a mobile phone.

The render server application interface was integrated into three completely different applications: One written in C++ using the QT platform that uses a scatter plot visualization of financial data called FinMotion [23], one gaming application using OpenGL rendering, and a Java application that visualizes cloth simulation in 3D and in real-time [7]. Especially the gaming application and the cloth simulation application make exhaustive use of hardware resources on the server side. The cloth simulation tool, which is already integrated into a CAD software for the garment industry, runs on computers with at least dual core CPUs and the latest graphics hardware.

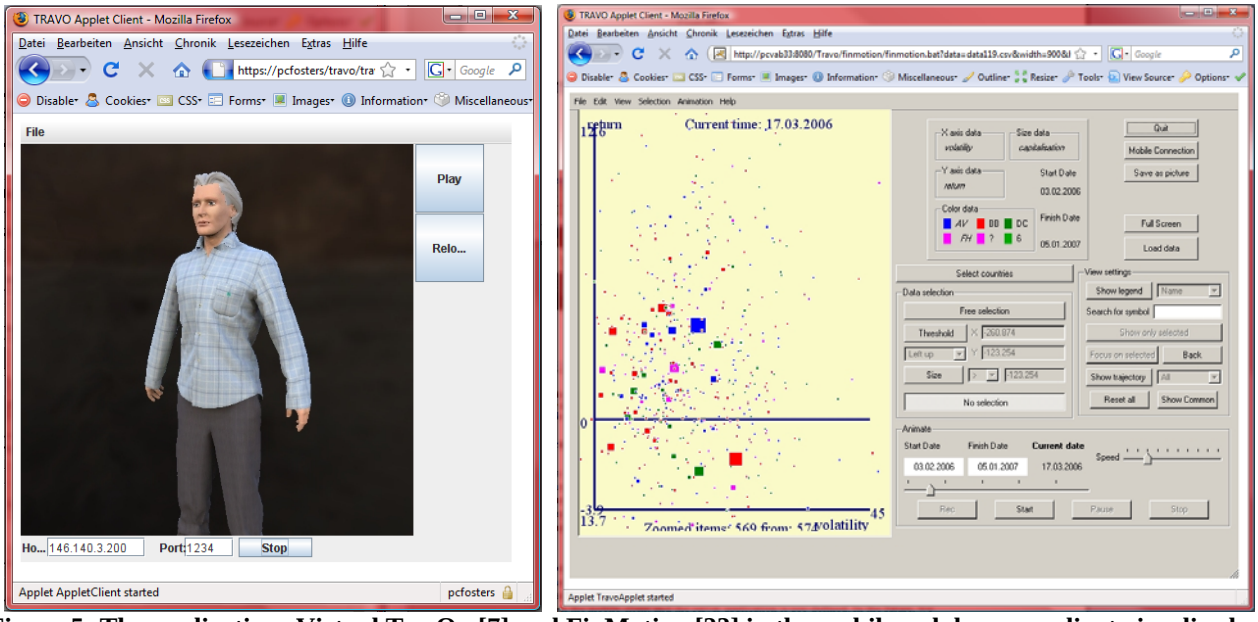

Figure 5: The applications Virtual Try On [7] and FinMotion [23] in the mobile web browser client visualized as remote applications. The cloth simulation application was slightly adapted to enhance the functionality while the scatter plot application is captured as a complete application window.

All three applications are connected to our remote visualization system by integrating the render server's application interface. On the client side, the same mobile client with only small modifications is used for all applications. As the mobile client is written with JME it can be used on a mobile phone as well as in a desktop environment. For the mobile client only the graphics content of the applications was captured, for the web version either the complete application (for FinMotion) or the rendering part of the application (for the cloth simulation). All applications run on the mobile client and can be used interactively. The game client allows a landscape walkthrough and basic interactions with the character, the cloth simulation allows to chose different garment setups that are simulated in real-time on the server side and visualized on the mobile client. In FinMotion only the functionality of browsing through the financial data sets is mapped onto the mobile client for smartphones, while the complete application functionality is available in the web browser version. Again, all different applications written with different toolsets or languages are integrated into our remote visualization system with low effort all using the same render server and all using the same remote client.

5.1 Business Impact

General trends in the field of mobile technologies show that bandwidth and coverage of available connection networks were continuously increasing for many years. It can also be anticipated that this trend will continue and accelerate in the coming years with technologies like WiMAX and LTE [1]. Providing a visualization technology that enables the user to remotely access data sources and applications will enable new workflow concepts independent from office locations and times.

We see our technology emerging from a flat rate evolution that will provide cost-effective and high-bandwidth access to business resources. The technology itself can provide the necessary graphics functionality of a remote workplace of the future that not only depends on voice and email access but also on the remote availability of visual information. The possibility to use applications from everywhere and to remotely make informed decisions will certainly save cost and time, and will support decision making processes for a variety of businesses and work flows.

6. Conclusion and Future Work

This paper introduces a system for remote visualization of graphical content. Our approach enables the integration of the technology in a variety of applications. Thus, with low additional effort application contents can be made available in mobile and web environments. This paper discusses the architecture of the system and its components, the necessary data formats and transmission techniques, and has shown several integration scenarios.

Regarding future work, we plan to include a better policy management in the render server to handle interaction requests from different users for the same visualization. Currently, the most recent interaction request is executed. Furthermore, the command set used for interaction between the mobile client and the server application is pre-defined. In the future, we will develop an interaction system that allows the definition of generic commands that will be interpreted later in the application that is connected to the render server on a mobile client. Another planned enhancement is the support for DirectX applications. DirectX provides options for widget capturing that can extend the existing interface and improve the overall performance of the capturing algorithms. Furthermore, a client-side DirectX implementation could improve the performance especially for the web clients. Finally, there are ways to improve the transmission of the image data by splitting the images into parts that are frequently updated and parts that are updated very rarely.

Those rarely updated parts of the overall image can be transmitted on demand. Parts of the application that are continuously updated have to be transmitted permanently.

Acknowledgment

We thank the Heinz-Nixdorf-Stiftung in Paderborn, Germany for funding the research project TRAVO, which formed the basis of this technology.

7. References

[1] 3GPP LONG TERM EVOLUTION. http://en.wikipedia.org/wiki/3GPP_Long_Term_Evolution
[2] COMMUNICATIONS MAGAZINE, IEEE. Volume 45, Issue 4, April 2007
[3] J. DIEPSTRATEN, M. GORKE, T. ERTL, Remote line rendering for mobile devices, 2004, In IEEE Computer Graphics International (CGI)'04
[4] K. ENGEL, O. SOMMER, AND T. ERTL. A Framework for Interactive Hardware Accelerated Remote 3D-Visualization. In Proceedings of EG/IEEE TCVG Symposium on Visualization VisSym '00, pages 167–177,291, May 2000.
[5] JAVA PLATFORM, MICRO EDITION (JAVA ME). http://java.sun.com/javame/index.jsp
[6] JAVA TECHNOLOGY. http://java.sun.com/
[7] K. KHAKZAR, R. BLUM, J. KOHLHAMMER, A. FUHRMANN, AN. MAIER, AX. MAIER. Interactive Product Visualization for an In-store Sales Support System for the Clothing Retail. In: HCI International 2007. Proceedings and Posters [DVD-ROM]: With 8 further Associated Conferences. Berlin, Heidelberg, New York : Springer Verlag, 2007, LNCS 4557, pp. 307-316.
[8] MOVING PICTURES EXPERTS GROUP. http://www.chiariglione.org/mpeg/
[9] M. NURMINEN. M-Loma – A Mobile 3D City Map. Proceedings of the eleventh international conference on 3D web technology, 2006
[10] MICROSOFT FOUNDATION CLASSES (MFC). http://msdn.microsoft.com/en-us/library/d06h2x6e(VS.80).aspx
[11] Microsoft Remote Desktop Protocol (RDP). http://www.microsoft.com/technet/prodtechnol/Win2KTS/evaluate/featfunc/rdpfperf.mspx
[12] OpenGL ES. http://www.khronos.org/opengles/
[13] K. PULLI, T. AARNIO, K. ROIMELA, J. VAARALA. Designing Graphics Programming Interfaces for Mobile Devices, I*EEE Computer Graphics and Applications Volume 25, Nr 6*, 2005
[14] K. PULLI, T. AARNIO, T. MIETTINEN , K. ROIMELA, J. VAARALA. Mobile 3D Graphics with OpenGL ES and M3G. San Francisco : Elsevier, Morgan Kaufmann, 2008. (The Morgan Kaufmann Series in Computer Graphics). - ISBN 978-0-12-373727-4
[15] QT. http://trolltech.com/products/qt/
[16] T RICHARDSON, Q STAFFORD-FRASER, K R WOOD, A HOPPER, Virtual network computing. IEEE Internet Computing 1998
[17] B.O. SCHNEIDER, I. M. MARTIN. An Adaptive Framework for 3D Graphics over Networks, Computers & Graphics, Vol. 23, No. 6, 1999.
[18] SILICON GRAPHICS, Inc. OpenGL Vizserver 3.0 – Application-Transparent Remote Interactive Visualization and Collaboration, 2003. http://www.sgi.com/.
[19] SOETEBIER, H. BIRTHELMER, J. SAHM. Client-Server Infrastructure for Interactive 3-D Multi-User Environments. In: Skala, Vaclav (Ed.) ; European Association for Computer Graphics (Eurographics): Journal of WSCG Volume 12 No. 3, 2004. Proceedings. Plzen : University of West Bohemia, 2004, S. 387-394.
[20] SOUMYAJIT DEB, P. J. NARAYANAN. RepVis: A Remote Visualization System for Large Environments. Proceedings of the Workshop on Computer Vision, Graphics and Image Processing (WCVGIP), Feb. 2004, Gwalior, India, pp. 54--57.
[21] S. STEGMAIER, M. MAGALL ´ON, AND T. ERTL. A Generic Solution for Hardware-Accelerated Remote Visualization. In Proceedings of EG/IEEE TCVG Symposium on Visualization VisSym '02, 2002.
[22] E. TELER D. LISCHINSKI, Streaming of Complex 3D Scenes for Remote Walkthroughs, Computer Graphics Forum, 2001, pp. 17-25
[23] T. TEKUSOVÁ, J. KOHLHAMMER. Applying Animation to the Visual Analysis of Financial Time-Dependent Data. IEEE International Conference on Information Visualization (IV), 11. 2007, Zurich, Switzerland, pp. 101-108
[24] VIRTUAL NETWORK COMPUTING OVERVIEW. http://en.wikipedia.org/wiki/Virtual_Network_Computing