EG UK Theory and Practice of Computer Graphics (2007)
Ik Soo Lim, David Duce (Editors)

Delivering Effective and Usable Interactive 3D Visualization on Lightweight Mobile Devices

I.R. Holmes and R.S. Kalawsky

Research School of Systems Engineering, Loughborough University, Loughborough, UK

Abstract

3D visualization has significantly enhanced scientists' ability to discover important new insights into their data. In this paper we focus specifically on associated HCI and human factors of remote 3D visualization. We discuss how, through adopting a user-centered design approach we have successfully engineered the Lightweight Visualization software system. This exciting HCI innovation has enabled scientists to interact with their own 3D visualization packages whenever they want and from wherever they happen to be. We describe how this advanced level of 'on the move' HCI was achieved by facilitating scientists to interact with their familiar desk-based 3D user interfaces via affordable, wireless computing devices. In particular, we highlight how the system has initially been employed within the UK e-Science RealityGrid project to support an extensively used 3D molecular dynamics user interface on the roaming, handheld PDA. Whilst it is recognized that these types of 'lightweight' devices possess limited compute, graphics and memory capability we also identify how Lightweight Visualization has overcome these serious limitations to effectively deliver usable interactive 3D visualization on the commodity mobile platform.

Categories and Subject Descriptors (according to ACM CCS): H.1.2 [User/Machine Systems]: Human factors H.5.2 [User Interfaces]: Graphical user interfaces (GUI) H.5.2 [User Interfaces]: User-centered design

1. Introduction

The vitally important role that visualization plays in scientific computing has been widely recognized ever since the late 1980's [MDB87]. The application and use of 3D visualization in modern fields such as Grid computing [FK99], and as a component of advanced HCI techniques such as computational steering [vLMvW97] has dramatically enhanced scientists' ability to discover important new insights into their data. We focus specifically on associated HCI and human factors of remote 3D visualization. In this particular application domain, HCI design is an extremely important area that has, to date, sadly often lagged behind in the drive to simply serve and project colorful imagery to the user.

In this paper we establish a set of HCI requirements for remote 3D visualization. We discuss how, through adopting a user-centered design approach we have successfully engineered the Lightweight Visualization software system. Traditionally, in order for scientists to properly interact with their 3D visualization applications they have often had to be tied, also sometimes for prolonged periods [KN04], to high-end graphics workstations. This underlies the basic need for a more flexible and convenient means of supporting 3D user interaction. Our Lightweight Visualization innovation has successfully enabled scientists to interact with their own 3D visualization packages whenever they want and from wherever they happen to be. This could be at any time of the day

Figure 1: *The RealityGrid PDA Client.*

© The Eurographics Association 2007.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

(or night) and from virtually any convenient location; especially whilst being away from the traditional 'desk' environment. We describe how this advanced level of 'on the move' HCI was achieved by facilitating scientists to interact with their familiar desk-based 3D user interfaces via affordable, wireless-enabled computing devices. In particular, we highlight how the system has initially been employed within the UK e-Science RealityGrid project to support an extensively used 3D molecular dynamics user interface on the roaming, handheld PDA (refer to Figure 1). Whilst it is recognized that these types of 'lightweight' devices possess limited compute, graphics and memory capability we also identify how Lightweight Visualization has overcome these serious limitations to effectively deliver usable interactive 3D visualization on the commodity mobile platform.

1.1. RealityGrid

We discuss the design and development of the Lightweight Visualization system primarily within the context of our work as part of the UK e-Science RealityGrid project (www.realitygrid.org). Within RealityGrid, the system has already achieved considerable initial success by underpinning the Grid-enabled interactive 3D visualization capability (as part of computational steering) of the RealityGrid PDA Client [HK06] (refer to Figure 1). The system, and the accompanying PDA GUI were both the product of a deep-track RealityGrid HCI design activity, which was initiated by a thorough human factors audit [KN04] of the project.

1.2. Related Work

The idea of scientists using lightweight mobile devices to interact remotely with their 3D visualizations is not an entirely novel concept. Several examples of successfully implemented mobile 3D GUI systems have emerged over recent years. Some of the more prominent instances have included OpenGL Vizserver [Oha99]. This system worked on the basis of intercepting the graphics library calls that a visualization application invokes on its host computer. Fully rendered 2D images (frames) were captured from the resulting graphics pipeline and served across a network for display on the user's remotely connected desktop or laptop PC. Hence, the user interface hardware did not have to possess any substantial (expensive) graphics capability in order to support 3D user interaction. Also, by externally intercepting application calls to the graphics library, the user was not required to instigate any internal modification to their visualization packages in order to use the system. The Vizserver GUI could thus by employed to support transparent, remote 3D user interaction with any OpenGL-based application.

A second, significant example of mobile 3D user interaction was demonstrated by the RAVE environment [GAW04]. In this system, ubiquitous 3D user interaction was supported across a range of high-end as well as low-end user interface devices, including the lightweight PDA. A particularly notable aspect was the system's capability to support both local and remote rendering, depending on the indigenous graphics capability of the user interface hardware.

One final example to highlight is the PDA client [LZS*03] that was developed for use with the Chromium [HHN*02] cluster-based rendering architecture. This lightweight GUI capitalized on Chromium's capability (similar to Vizserver) to intercept OpenGL calls on a host computer resource. The user was thus able to interact via the wireless PDA, just as they would through the Vizserver client with the resulting graphics stream from any OpenGL-based 3D application.

1.3. HCI Requirements Analysis

The emphasis in remote visualization has, to date, often been placed upon system capability to project remotely rendered images onto users' mobile interface devices. Equal consideration has less often been given to how the user actually needs to interact with their 3D visualizations. Yet this is a vitally important HCI design factor that should not be overlooked. The human factors audit [KN04] that was conducted as part of RealityGrid identified how the project's groups of application scientists needed to interact with their Grid-based 3D visualizations. The wide range of research activities within RealityGrid meant that a considerable number of specialized as well as bespoke 3D user interfaces needed to be employed; as opposed to the generic solution. This identified HCI requirement raised two key issues with regard to supporting lightweight 3D user interaction. Firstly, the scientists' required 3D visualization GUIs/packages could not be expected to operate in certain lightweight computing environments; the PDA, for example. Secondly, even where the required GUI could be deployed locally, perhaps onto a thin laptop PC, the sheer complexity or size of the visualization data (up to terascale simulations in RealityGrid) would be overwhelmingly too excessive for current lightweight device technology to render at interactive speeds.

The remote OpenGL approach that was implemented in Vizserver [Oha99], as well as in the PDA GUI [LZS*03] for Chromium offered a partial solution for lightweight 3D user interaction. This method showed that it could support ubiquitous mobile user interaction on both the handheld and the laptop platforms. The benefit of the user not having to modify their visualization packages in order to employ the system also made for a compelling argument. However, the OpenGL solution was found to be deficient within the context of our HCI requirements. It supported lightweight user interaction with the graphics pipeline, but not the actual underlying visualization data, or the required user interface of the scientists' applications. This deficiency raises a problem for domains such as RealityGrid, where a considerable number of specialized as well as bespoke 3D visualization user interfaces are required. The OpenGL solution also presents a particular issue for HCI models such as computational steer-

© The Eurographics Association 2007.

ing (the central theme in RealityGrid), where user interaction via direct control [vLMvW97] is a requirement.

Another important HCI requirement relates to user interaction performance. Specifically in terms of remote 3D visualization, a key factor is the amount of latency that is generated in between the user interacting and subsequently viewing the effect within the remotely rendered, served and locally displayed images. It is commonly regarded [KOC05] that a consistent response time of around 0.1s is required, if the user is to have the ideal sensation of real-time interaction with their application. However, remote 3D visualization GUI systems for the PDA, and other user interface devices of similar limited (lightweight) capability have rarely been able to achieve this level of performance. Without the capability to support efficient, and ideally real-time user interaction, lightweight UIs (in particular the PDA, mobile phone, etc.) may only realistically be regarded as toys, and not as being the beneficial 'usability improvement' technologies for real, everyday scientific application.

2. Lightweight Visualization

The Lightweight Visualization system has been engineered with the user-centered objective of satisfying the identified HCI requirements to effectively deliver usable interactive 3D visualization on lightweight mobile devices. Lightweight Visualization was primarily designed in response to the findings of the RealityGrid human factors audit [KN04], which clearly identified the need for a more flexible and convenient means of supporting interactive 3D visualization, as part of computational steering, in Grid computing environments. However, it is important to emphasize that our strategy from the outset was to design and build a generic technology to support lightweight user interaction in any field or application domain of computer visualization.

2.1. System Architecture

At the heart of Lightweight Visualization is a distributed system architecture comprising four distinct software components (refer to Figure 2). The user's desk-based 3D visualization GUI/package is instrumented with a small bespoke code module, called the **LViz Module** (refer to Figure 2). This provides an application interface to the required elements of the native 3D user interface. The module is responsible for invoking specific user interaction functionality from within the visualization application. It also captures the resulting graphics output by recording sequences of rendered 2D frames to persistent digital image files on the hard disk. The module receives requests for user interaction functionality, in the form of text-based messages, from a separate application called the LViz Server.

The **LViz Server** (refer to Figure 2) deploys onto the same computer resource as the user's visualization application. It communicates asynchronously with the LViz Module over a local-domain socket. The server is responsible for receiving user interaction requests from remote clients. Request messages are buffered within the server, before being forwarded on for invocation within the LViz Module. The server also encodes and serves the resulting sequences of rendered 2D frames back across a network for display within remotely connected instances of the LViz Client.

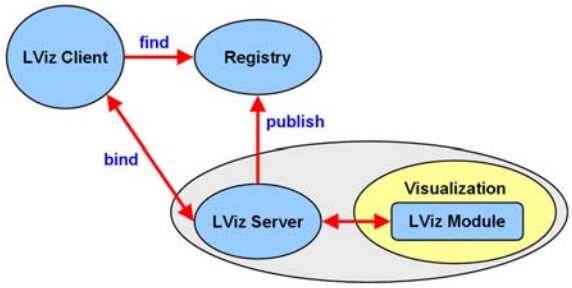

Figure 2: *Lightweight Visualization Architecture.*

The **LViz Client** (refer to Figure 2) supports the front-end to Lightweight Visualization. Its GUI is tailored specifically to suit the user's individual interaction needs, and/or the user interface of their required 3D visualization package(s). The client captures package-specific interaction commands from the user's inputs. These commands (user interaction request messages) are then promptly dispatched across a network to the LViz Server. The client also processes and displays the resulting sequences of 2D frames, which are received from the LViz Server in an asynchronous fashion. All communication (requests and images) between client and server is mediated via a fast and direct TCP/IP socket link.

The final component of the Lightweight Visualization architecture is the **Registry** (refer to Figure 2). This provides a stateful Web service to which the LViz Server publishes descriptive detail of scientists' visualizations. By querying the Registry, the LViz Client can easily find and dynamically couple to 3D visualization applications (via the LViz Server), without the user having to memorize and manually input specific details of remote network endpoints. This aspect of Lightweight Visualization was derived directly from the RealityGrid computational steering system [PHPP05].

2.2. LViz Module

A key HCI requirement specified that application scientists ideally needed to interact with the user interface from their own 3D visualization package(s). In the specific case of RealityGrid, the required user interfaces were found [KN04] to be tightly integrated into the software. This meant that an effective mechanism had to be devised for supporting users to easily integrate their own 3D visualization packages into the Lightweight Visualization environment. The approach that was implemented to achieve this effect within the RealityGrid computational steering environment was to provide an

© The Eurographics Association 2007.

API as part of the architecture [PHPP05]. Users could thus utilize the API to instrument (Grid-enable) their own codes. This solution worked well for the RealityGrid computational steering system, but it raised several key issues with regard to its prospective use in Lightweight Visualization.

In order to utilize the API system, some form of user access to the visualization code must be available. This would not always be practical for the case of Lightweight Visualization, where third-party or commercial 3D visualization packages needed to be used [KN04]. In addition, the amount of time and effort that may be expected of the application scientist (user) to instrument the more complex of visualization codes [FCJW04], could discourage or potentially even prohibit the use of Lightweight Visualization. One final point is that the Lightweight Visualization system operates on the basis of supporting tailored lightweight GUIs to required 3D user interfaces. The user interface can vary considerably between different visualization packages, so any generic solution would be deficient in this specific instance.

The approach that we adopted to support easy integration of scientists' 3D visualization packages into the Lightweight Visualization environment was discerned from [vWvLM00]. Here, it was suggested that a partial solution could be found in modern modular or extensible system architectures. In RealityGrid, a small bespoke code module had to be written [PHPP05] specifically for the modular architecture of the commercial (closed source) AVS/Express visualization package. Embedding a bespoke module was the only viable option for instrumenting AVS/Express, which needed to be used [KN04] by a number of application scientists in conjunction with the RealityGrid computational steering system. It was this same bespoke modular/extensible solution that we have carried forward into the system design of Lightweight Visualization.

Rather than providing a generic object that the scientist uses to instrument their visualization code, the remote architecture of Lightweight Visualization has been split into two separate entities: the LViz Server and the LViz Module (refer to Figure 2). The server constitutes 99% of the remote functionality, and is a completely generic technology. Its counterpart module is an abstract entity that, once implemented, provides the remaining 1% of the functionality. The module adheres to a specific application interface so that it can communicate correctly with the server component. Everything else about it is bespoke and tailored specifically to provide external access to the scientist's required 3D user interface. The module can be rapidly engineered in any programming language (to suit the 3D application), either by the user themselves or alternatively by a third party. A number of pre-implemented 'template' modules written in a variety of popular languages have also been included as part of the initial distribution, and these can be very quickly customized to suit individual requirements.

For 3D visualization GUIs/packages that support modular extensibility (AVS/Express, for example), the LViz Module can be easily configured and embedded as a supplementary system component. This supports a user-friendly alternative to the application scientist having to internally modify complex visualization code. The module can also be very easily configured as a shell script. This provides another user-friendly option for visualization GUIs/packages that support user interaction via standard input commands. For 3D systems where neither modular extensibility nor standard input commands are supported (bespoke codes, for example), the option still remains for the LViz Module to be configured in the package's native language and compiled into its binary. It would however also be entirely feasible to implement the LViz Module, similar to [LZS*03], as an OpenGL stream processing unit (SPU) for the Chromium system. This would not explicitly satisfy the HCI requirement for the scientist to interact remotely with their visualization user interface, but it would provide an alternative in the most extreme of cases where code instrumentation was not a justifiable option.

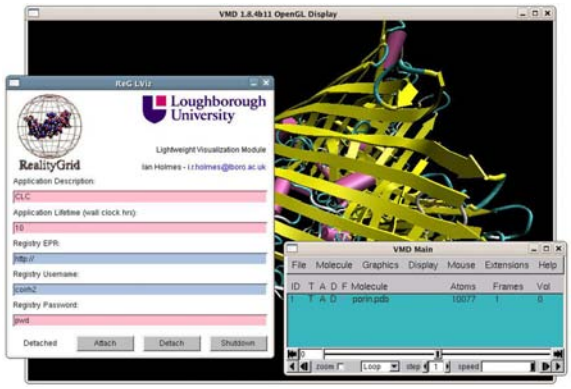

Figure 3: *LViz Module (left) in VMD.*

For the initial test case of Lightweight Visualization being deployed within RealityGrid, the LViz Module was implemented for the well established 3D molecular dynamics visualization GUI/package: VMD [HDS96] (refer to Figure 3). VMD was selected as the test package because of its extensive use [FCJW04, JCH05] within the RealityGrid project. It supported modular extensibility, and thus provided a good example of how a third-party visualization code (a particularly complex code in this instance) could be quickly and easily integrated into the Lightweight Visualization environment. When VMD was originally instrumented using the RealityGrid computational steering API, the whole process [FCJW04] took several researchers working for a considerable amount of time, and proved to be a challenging endeavor. In contrast, the LViz Module that was built for VMD comprised only around two dozen lines of high-level Tcl/Tk code (1% of functionality). The entire process of developing the module from scratch and fully integrating VMD into the Lightweight Visualization environment took one researcher

© The Eurographics Association 2007.

working for less than a day. This enforces our decision to design a split (99%/1%) generic/bespoke solution.

In our initially implemented RealityGrid project example, the LViz Module provides remote (lightweight) GUI access, via the LViz Server, to the dedicated 3D molecular dynamics user interface of the VMD package. The module can be loaded when the visualization is launched, via a command line argument. It can also be loaded at run-time via the shell, or alternatively via it's built in Tk GUI (refer to Figure 3). In the latter case the module's GUI is accessed through a menu option, which we very easily embedded into the native VMD front-end. Once the LViz Module is up and running, VMD becomes fully integrated with Lightweight Visualization. It can therefore be discovered through the Registry, and can also begin receiving remotely issued user interaction requests through the LViz Server. This same format would apply to any 3D visualization GUI/package for which the LViz Module had been implemented.

2.3. LViz Server

The LViz Server constitutes the primary component (99% of functionality) of the remote Lightweight Visualization architecture (refer to Figure 2). It is a self-contained application that deploys onto the same computer resource as the user's 3D visualization applications. The LViz Server provides two essential elements of functionality. Firstly, a message passing service that facilitates remote clients to communicate user interaction requests to the 3D visualization. Secondly, a visualization serving facility that retrieves the rendered sequences of 2D frames from the hard disk (courtesy of the LViz Module), encodes them, and serves them back across a network for display.

The LViz Server was coded in the JAVA programming language. JAVA was selected because of its intuitive threading model, as well as for its extremely useful generic digital imaging API: JIMI. JIMI supported a uniform interface to encode and decode rendered 2D frames to and from just about any digital image format imaginable. This level of comprehension was essential in order to cater to the various native image file formats of different 3D visualization packages. For example: VMD outputs rendered 2D frames to the obscure Truevision TGA (TARGA) format.

The visualization serving element of the LViz Server was a crucial HCI/system design factor. In order to satisfy our HCI requirements, Lightweight Visualization had to consistently support efficient, and ideally real-time 3D user interaction at its thin front-end. In the context of the system's initial deployment within RealityGrid [HK06], this meant underpinning fast (interactive) system response times and high frame display rates on the limited PDA device. Visualization serving latency was identified [KOC05] as the major factor impacting upon remote 3D visualization performance. Thus, within the design of the LViz Server, latency resulting from rendered frames being served to the lightweight client had to be significantly reduced, through image compression, without adversely affecting the user's interaction with the system; i.e. without adversely compromising the user-perceived quality of the locally displayed imagery. We achieved this via the implementation of an adaptive frame encoding scheme within the LViz Server (refer to Figure 4). This scheme was designed to yield a usable/interactive trade-off between network throughput and local display quality.

Figure 4: *Adaptive frame encoding in Lightweight Visualization: lossless (left) / lossy (right).*

A HCI design assumption was made that whilst the user was actively interacting with their 3D visualizations (rotating models, panning or zooming cameras, etc.) they would not need to view the resulting imagery at full resolution and in full lossless quality. Therefore, during active periods of user interaction, all rendered frames are encoded and served to the lightweight client at reduced resolution and in lossy JPEG format (refer to Figure 4). A further HCI design assumption was made that once the user completes their interaction (stops rotating the model, for example), they would then need to analyze the fine detail within the image. Thus, the final frame in any given sequence of served images is always encoded at full resolution and in lossless PNG format (refer to Figure 4). Sequences of encoded frames, along with preceding user interaction request messages are transmitted between client and server over a fast and direct TCP/IP socket. TCP/IP was selected primarily because of its speed in order to satisfy the HCI performance requirement, but also for its ability to guarantee delivery of network packets.

In addition to adaptive frame encoding, multi-threading techniques also played a crucial role within the system design of the LViz Server. The multi-threaded internal architecture of the server has underpinned several key usability features of the system. The server is capable of simultaneously receiving user interaction requests from multiple clients, whilst concurrently forwarding requests to the 3D visualization (via the LViz Module), whilst also concurrently

© The Eurographics Association 2007.

encoding and serving rendered frames. This underpins efficient system performance, which was essential towards satisfying our HCI requirements, as well as supporting a beneficial degree of collaborative user interaction. The LViz Server also, again through multi-threading, configures its image encoder to suit indigenous, individual lightweight device display properties. Hence, rendered frames are always encoded and served to the correct resolution and aspect ratio specification. This underpins the system capability to support ubiquitous user access via different types of lightweight device (thin laptop PC, PDA, mobile phone, etc.). Finally, our implementation of multi-threading has also enabled the system to facilitate lightweight 3D user interaction with a number of concurrently running visualization applications. This usability aspect is supported through a single instance of the LViz Server, which executes on a shared computer resource. The server maintains distinct sets of interacting clients for each active visualization application. This capability also extends to support heterogeneous 3D visualization packages (VMD at the same time as AVS/Express, for example).

2.4. LViz Client

The LViz Client (user interface), as with the other architectural aspects of Lightweight Visualization, posed a considerable HCI design challenge. Our HCI requirements specified that application scientists ideally needed to interact with the user interface from their own, adopted 3D visualization packages. However, an assumption was made that the visualization GUIs/packages themselves could not be deployed locally onto the lightweight device. It was therefore established that the best HCI design solution was to create the illusion of the users' familiar 3D user interfaces within the lightweight GUI; therefore supporting the required format of user interaction. Our derived and practical approach to achieve this effect was to implement an easily customizable, object-oriented front-end system (refer to Figure 5).

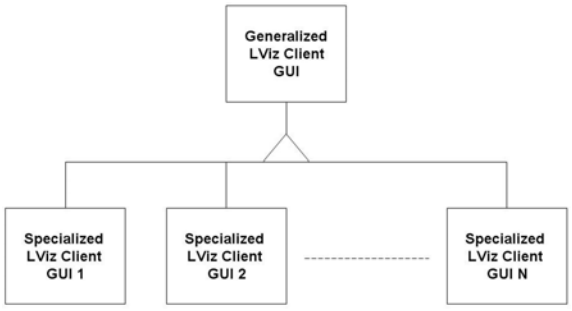

Figure 5: *LViz Client OO GUI system.*

Within the LViz Client *oo* front-end system, a generalized GUI class (refer to Figure 5) provides all of the essential 'client-side' Lightweight Visualization user interaction functionality. An instantiated generalized GUI object supports the retrieval and display of visualization application details from the Registry. It facilitates the establishment of the network connection to the LViz Server (the user's selected visualization), along with all subsequent user interaction request dispatching operations. The generalized GUI also supplies all of the essential, autonomous capability for receiving and displaying sequences of remotely rendered 2D frames onto the lightweight device display screen.

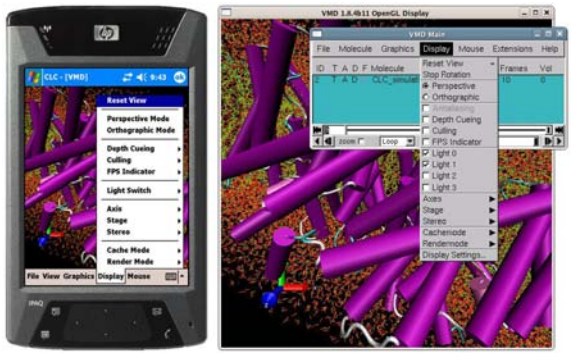

Figure 6: *VMD user interface (LViz Client) in the RealityGrid PDA Client (left) and the native VMD front-end (right).*

From the template of the generalized GUI, a series of specialized GUIs (refer to Figure 5) can be rapidly constructed. A specialized LViz Client GUI inherits all of the generic functionality from the base class. This equips any user interface with the capability to interact with the Lightweight Visualization environment. All that remains is to simply drag and drop the required 'knobs and dials' (input widgets) to create the illusion of the native visualization user interface (refer to Figure 6). Each tailored component of the specialized GUI (buttons, menus, sliders, etc.) is assigned with a specific user interaction request message (text string). These requests will be dispatched to the LViz Server whenever the user triggers the corresponding interaction event ('clicking' a button, for example). The practice of inheriting functionality from the base class in order to tailor a specialized LViz Client GUI may at first seem complicated. In reality it is an extremely quick and straight forward process for anyone with a modicum of basic skill in using a visual IDE.

To support the creation of the RealityGrid PDA Client, the *oo* LViz Client GUI system has initially been implemented in the Microsoft .NET Framework (C#). This facilitated a PDA front-end to the VMD package (refer to Figure 6) to be rapidly constructed within the Visual Studio IDE. In this instance the specialized LViz Client was tailored with the required set of menu options, which were representative of those found within the native VMD GUI (refer to Figure 6). The display object (PictureBox) was configured so that the user was able to rotate, pan, zoom, etc. simply by dragging the PDA stylus across the screen. The client was also configured to incorporate a series of dialog forms that provided

© The Eurographics Association 2007.

lightweight user access to some of the more advanced and required user interaction capabilities of the VMD package.

2.5. Initial Timings

To record initial performance data from our Lightweight Visualization system, an accordingly adapted version of the RealityGrid PDA Client was constructed. The GUI was installed onto a Dell Axim X3i PDA (refer to Figure 4), which comprised a 400Mhz CPU, 64MB of memory, and a standard QVGA (240*320) display. The LViz Server and VMD were hosted onto a standard Linux PC with a 2.0GHz processor, 256MB of memory, and a commodity GeForce4 graphics card. The Registry and its required elements of the RealityGrid environment, which did not contribute towards the performance evaluation but were however required, were hosted onto a separate Linux PC. VMD was loaded with a molecular structure that comprised approximately 50000 atoms, and is shown in Figures 3 and 4. Networking was supported via built-in 802.11b Wi-Fi capability on the PDA device, and a wireless Internet access point on the Loughborough University (100Mbps) campus network.

Mean	Min	Max	Variance	Std. Deviation
0.229	0.214	0.242	0.04	0.006

Table 1: *System response time (lossless)(seconds).*

Mean	Min	Max	Variance	Std. Deviation
0.16	0.14	0.194	0.089	0.009

Table 2: *System response time (lossy)(seconds).*

Mean	Min	Max	Variance	Std. Deviation
13.28	12.22	14.16	0.09	0.29

Table 3: *Frame display rate (fps).*

Three performance measurements were recorded, and the results are shown in Tables 1, 2 and 3. The first two measurements recorded user interaction response time. In both cases the ensuing latency was measured in seconds, in between the action of the user issuing an interaction request from the PDA, and event of the device display screen being subsequently updated with the corresponding rendered image. In the first measurement (refer to Table 1) the LViz Server encoded and served back a single lossless PNG image. In the second instance (refer to Table 2) a single lossy JPEG was returned. This was so as to factor the adaptive encoding capability of the server into the performance evaluation. The third performance measurement (refer to Table 3) recorded frame display rate on the PDA device. The action of the user dragging the PDA stylus across the screen to rotate the model was simulated within the GUI. This triggered the server side of the Lightweight Visualization architecture to capture, encode and serve back a sequence of 100 rendered frames (99 lossy and 1 lossless). The resulting frame display rate was measured in frames-per-second (fps) as each of the 100 images was received and projected onto the PDA screen. All three measurements were repeated 100 times in order to record a representative distribution of performance data.

3. Conclusion and Future Work

The goal of the research that we have presented in this paper was to deliver effective and usable interactive 3D visualization on lightweight mobile devices. We specified several HCI requirements, which were derived primarily from the findings of the RealityGrid human factors audit. The application scientist needed to interact via the lightweight GUI with the user interface from their own, adopted 3D visualization package(s). The system also needed to support interactive performance at its lightweight front-end, irrespective of limitation in device hardware capability.

The RealityGrid human factors audit identified that different groups of users needed to interact with different 3D visualization packages. The Lightweight Visualization environment was designed to provide the optimal compromise to the generic solution, which was found to be deficient in this particular instance. The majority of the system architecture (99%) was designed and built as a generic technology, for use within any application domain of 3D visualization. The remaining 1% of the system was left open-ended, and could thus be easily customized or 'tailored' to suit individual user requirements.

Special emphasis within our system design was placed upon minimizing the level of difficulty that would be involved for the user to implement the customizable aspects of the Lightweight Visualization environment. The LViz Module was designed to support quick and easy instrumentation of any visualization package; especially where modular extensibility or user interaction via standard input commands was supported. This was demonstrated in the highlighted example of VMD, where only a miniscule amount of effort was required to integrate this particularly complex visualization code into the Lightweight Visualization system. The *oo* LViz Client GUI system also supported quick and easy 'drag and drop' construction of tailored lightweight 3D user interfaces. This capability was demonstrated in the example of the RealityGrid PDA Client, where a series of input widgets were simply 'bolted on' to the PDA front-end in order to rapidly construct a handheld version of the VMD user interface.

To achieve the ambitious performance aspect of our HCI requirements, a sophisticated combination scheme of adaptive frame encoding and multi-threading was implemented within the LViz Server. The effect was to significantly reduce visualization serving latency within the system, which was identified as the major factor impacting upon remote 3D visualization performance. This was confirmed by the consistently good initial timings that we recorded on the extreme lightweight example of a limited PDA device.

© The Eurographics Association 2007.

Our future research is currently focused in several main areas. The Lightweight Visualization system and the RealityGrid PDA Client will both shortly undergo a thorough user evaluation within the research groups of the RealityGrid project. We are also investigating several optimizations to the system. These will include tying the adaptive frame encoder into varying levels of wireless network performance, as well as developing the *oo* GUI system for the standard Linux and Windows laptop/desktop PC platforms. Security is also a key concern. The RealityGrid PDA Client employed X.509 certificates to support user authentication and secure user interaction in the RealityGrid computational steering environment. This security provision will shortly also be extended into the Lightweight Visualization system.

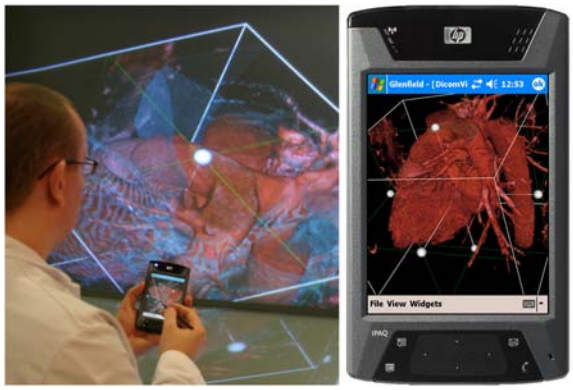

Figure 7: *Future application of Lightweight Visualization to support cardiac diagnosis at the Glenfield Hospital.*

In addition to our work within RealityGrid, we have also recently engaged into a collaboration with our local University Hospitals of Leicester NHS Trust Glenfield Hospital. This collaborative endeavour is extending the user trial of Lightweight Visualization into the clinical visualization domain. For this purpose, a specialized LViz Client GUI for the PDA has already been constructed (refer to Figure 7). The PDA GUI is being used, on an evaluation basis, to aid in a collaborative clinical diagnosis process by supporting a roaming, lightweight 3D user interface to a stereoscopic cardiac data (MRI/CT scan) visualization package (refer to Figure 7). The stereoscopic visualization package was also developed at Loughborough University, albeit separately from our Lightweight Visualization system, and is currently also undergoing a similar, initial evaluation for a prospective future clinical trial at the Glenfield Hospital.

References

[FCJW04] FOWLER P. W., COVENEY P. V., JHA S., WAN S.: Exact calculation of peptide-protein binding energies by steered thermodynamic integration using high performance computing grids. In *Proc. 3rd UK e-Science All Hands Meeting* (2004).

[FK99] FOSTER I., KESSLEMAN C.: *The Grid: blueprint for a new computing infrastructure*. Morgan Kaufmann, 1999.

[GAW04] GRIMSTEAD I. J., AVIS N. J., WALKER D. W.: Rave: Resource-aware visualization environment. In *Proc. 3rd UK e-Science All Hands Meeting* (2004).

[HDS96] HUMPHREY W., DALKE A., SCHULTEN K.: Vmd - visual molecular dynamics. *J. Molec. Graphics 14* (1996), 33–38.

[HHN*02] HUMPHREYS G., HOUSTON M., NG R., FRANK R., AHERN S., KIRCHNER P. D.: Chromium: a stream-processing framework for interactive rendering on clusters. In *Proc. ACM SIGGRAPH'02* (2002).

[HK06] HOLMES I. R., KALAWSKY R. S.: The realitygrid pda and smartphone clients: Developing effective handheld user interfaces for e-science. In *Proc. 5th UK e-Science All Hands Meeting* (2006).

[JCH05] JHA S., COVENEY P. V., HARVEY M. J.: Spice: Simulated pore interactive computing environment. In *Proc. ACM/IEEE Conf. Supercomputing* (2005).

[KN04] KALAWSKY R. S., NEE S. P.: *e-Science RealityGrid Human Factors Audit - Requirements and Context Analysis*. Tech. rep., Loughborough University, 2004.

[KOC05] KALAWSKY R. S., O'BRIEN J., COVENEY P. V.: Improving scientists' interaction with complex computational-visualization environments based on a distributed grid infrastructure. *Phil. Trans. R Soc. A: Mathematical, Physical and Engineering Sciences 363*, 1833 (2005), 1867–1884.

[LZS*03] LAMBERTI F., ZUNINO C., SANNA A., FIUME A., MANIEZZO M.: An accelerated remote graphics architecture for pdas. In *Proc. 8th Int. Conf. on 3D Web Technology* (2003).

[MDB87] MCCORMICK B., DEFANTI T., BROWN M.: Visualization in scientific computing. *Computer Graphics (SIGGRAPH '88) 22*, 8 (1987), 103–111.

[Oha99] OHAZAMA C.: *OpenGL Vizserver White Paper*. Tech. rep., Silicon Graphics Inc., 1999.

[PHPP05] PICKLES S. M., HAINES R., PINNING R. L., PORTER A. R.: A practical toolkit for computational steering. *Phil. Trans. R Soc. A: Mathematical, Physical and Engineering Sciences 363*, 1833 (2005), 1843–1853.

[vLMvW97] VAN LIERE R., MULDER J. D., VAN WIJK J. J.: Computational steering. *Future Generation Computer Sytems 12*, 5 (1997), 441–450.

[vWvLM00] VAN WIJK J. J., VAN LIERE R., MULDER J. D.: Bringing computational steering to the user. In *Proc. Scientific Visualization Conf. (DAGSTUHL '97)* (2000), Hagen H., Nielson G. M., Post F. H., (Eds.), IEEE Computer Society, pp. 304–304.

© The Eurographics Association 2007.