The rapid evolution of pervasive computing technologies enables bringing virtual heritage applications to a new level of user participation. A platform for interactive virtual heritage applications integrates a high-end virtual reality system with wireless, connected portable and wearable computers, facilitating and enhancing user navigation, visualization control, and peer-to-peer information exchange.

Virtual heritage (VH) technology has developed rapidly in the last decade. Its evolution is propelled by the coming of age of several enabling technologies, namely high-speed computer networks and the Internet, high-performance computer graphics hardware and software, improved data collection, and model construction devices and applications. The diffusion of VH applications can be strategic for the economic development of many countries, such as Greece and Italy, with an unrivaled historical tradition. VH technology can make their cultural wealth accessible to the worldwide public, with obvious positive consequences on the development of tourism and further valorization of historical resources.

VH evolution has gone through several phases. In its basic form, it’s limited to simple Web-based interfaces to historical sites, museums, and collections. More advanced, second-generation VH applications exploit virtual reality (VR) technology to give users immersive experiences, such as archaeological site navigation, time and space travel, and ancient artifact reconstruction in 3D. In this context, our work focuses on the third generation of VH applications, which aims to provide an immersive and interactive user experience. Interactive VH has a long tradition, but its development has been hampered by the limited availability, poor usability, and high cost of VR interface devices used in its early embodiments.

Pervasive computing technologies have created the conditions for overcoming the limitations of early interactive VH prototypes. Our work exploits widely available portable and wearable computing platforms (PDAs and cellular phones) and software environments (Java) to create highly interactive VH applications based on popular, low-cost, wireless terminals with highly optimized interfaces. In other words, we enable the VH application user to interact in the VR environment with his or her own palmtop or mobile phone, without having to become familiar with a new I/O interface.

Furthermore, we improve interactivity by enabling not only single-user interaction with a virtual environment but also multiple-user, peer-to-peer, and many-to-many interaction in the same environment. This is the first step toward a collaborative fruition of the VH content. The applications of these ideas and technologies are numerous, including virtual museums, virtual archaeology laboratories, group exploration of virtual historical sites, games, and entertainment.

Use of ubiquitous computing and wireless technologies for interaction in immersive virtual environments (IVEs) poses many challenges in terms of real-time performance. VH applications need high interactivity in terms of subjective quality of system response observed by the users, to preserve their sense of immersion and presence. Thus, it’s important to study system reaction time to events generated by users to ensure adequate interactivity. This is not trivial when using mobile systems; wireless devices are often affected by limitations related to processing resources and network bandwidth and latency.

In this work, we describe a software and hardware infrastructure based on off-the-shelf components to support collaborative interaction in VH. We can enrich VH applications by displaying multimedia data and sharing them with other users. Furthermore, usability increases when providing effective, mobile user interfaces that target a wide range of single users and groups.

Our aim is not to offer a completely new experience but to develop an appropriate system solution for a real set of multimedia requirements, using commodity components. We describe a prototype VH application, namely a PDA-controlled navigation interface (see Figure 1), which
is also a collaborative analysis and exploration tool for the tombs on the ancient Appian Way. These applications provide a realistic test bed for performance analysis of the basic interaction functions implemented in our framework. Our quantitative analysis demonstrates that performance levels are adequate for navigation and interaction among a limited number of users. The “Background and State of the Art” sidebar describes work by other researchers.

System requirements

Our primary objective is to manage interaction and communication in a collaborative virtual environment shared by archaeologists and cultural heritage experts (but also architects, virtual model designers, art experts, computer scientists, and so on). In other words, we do not limit ourselves to simple interactions as in the case of a visitor in a VH site. We view the VH site as an active workplace, where users have ample degrees of freedom to proactively interact with the environment and to cooperate with others. To achieve this objective, we need to satisfy three fundamental technical requirements: sense of presence, multicontent cooperation, and multimedia data support.

Background and State of the Art

Research in innovative technologies aimed at presenting cultural heritage sites with immersive virtual reality (IVR) is active. As a representative example, researchers at the Fraunhofer Institute applied innovative visual metaphors in the IVR presentation of the cathedral of Siena. An avatar leads the visitor through the cathedral reconstructed in a passive immersive environment. Alternatively, the visitor can choose views and notable areas through a GUI that mimics an active book with multiple pages and selection buttons. One of the most important results of virtual heritage (VH) is the presentation of contents not available otherwise. In the Nuovo Museo Elettronico della città di Bologna (Nu.M.E.) project, the users are provided with simple and efficient navigation tools to navigate a virtual world in space and in time.

At the Foundation of the Hellenic World, VR is employed to create immersive, interactive, and photorealistic experiences for the general public. The VR equipment used in this exhibition was chosen with the goal of offering technologies accessible to people with different backgrounds. The main limitation of this type of application is interactivity, due to access provided via a stationary console (for example, Behr et al. use a Windows PC with a touch screen). This allows only a small group of visitors to simultaneously interact with the virtual world, while others can have only a passive experience. Moreover, VR equipment and interaction devices are expensive, cumbersome, limited by multiple cables, and not user friendly.

To overcome the limitations in interactivity in the VH environment, several researchers have explored pervasive computing technologies. The wireless network infrastructure provides untethered connection between I/O devices carried by the user and the server infrastructure (for rendering and visualization) of the virtual environment. Furthermore, pervasive computing devices are becoming widely available (noticeable in this sense are PDAs and mobile phones) and users are accustomed to their interfaces. Therefore, the objective is to exploit ubiquitous computing to enhance usability and interactivity in VR.

Examples of research efforts in this area started in 1993 when Fitzmaurice et al. verified spatial awareness and different interaction metaphors for a tracked, small, handheld monitor acting as a palmtop computer. Other work has implemented virtual handheld interaction devices, that is, physical devices for which a representation is present in the virtual environment. An example, presented in 1995, is the Virtual Tricoder, which maps a physical input device, the Logitech Flymouse, to a visual model. In 1999 the Haptic Augmented Reality Paddle (HARP) project implemented a handheld window through a physical paddle in the virtual environment. This technique is based on tracking of a small object and hiding it under a virtual representation. This provides the tactile and force feedback, and the user has the illusion of holding a handheld interaction device that has the computing power of a high-performance graphics workstation. This technique was necessary because past handheld devices didn’t have much computing power and storage capabilities. This solution isn’t the best one, however, because the interaction through virtual devices is uncomfortable. These devices can be nonintuitive and also invasive, occluding the user’s view of the virtual world.

The rapid evolution of technology has made possible some recent practical combinations of VR and real handheld devices.
Continued from p. 47

This is well illustrated in Watsen, Darken, and Capps who used a 3Com Palm Pilot to interact with a CAVE-like environment. Another example is the Java-Based Interface to the Virtual Environment (JAVE) project, which exploits a handheld tablet and wireless networking. It uses Java and accommodates custom-designed interfaces. Finally, at Siggraph 2002, researchers presented two interesting projects in this direction. First, Virtual Reality Applications Center researchers presented Tweek, a mid-dleware tool providing an extensible Java GUI interacting with VR applications. This system can run on desktop and palmtop computers in projection-based VR systems or in a 3D virtual space. The second work concerns the ARS Box and the Palmist project, a PC-based VR system combined with a CAVE-like environment, where a handheld PC works as its interaction interface. This solution offers a low-cost system and at the same time enhances interaction and content presentation through separate management of three projection screens from a unique device, a handheld PC. A limitation is that the interface isn’t platform independent, and only one person at a time can experience interactivity. The latter is the major limitation also for Tweek.

References


same time. These characteristics make CAVEs and virtual theaters suitable solutions for collaborative experiences.

A CAVE is a cube composed of display screens completely surrounding the viewer. The user can explore the virtual world by moving around. A virtual theater is a large-screen system that fills the field of view of users seated on chairs like in a real theater. In both cases the illusion of immersion is obtained wearing stereo glasses and providing the viewers with multimodal devices (trackers, gloves, and so on) to interact with virtual objects in the scene, which are calculated and projected in real time.

The satisfaction of real-time constraints is obtained through high-performance graphical libraries and VR toolkits and through graphic workstations with dedicated hardware components that relieve the main CPU from many tasks. Usually a graphical subsystem works in parallel as part of a pipeline system, interfacing itself with the CPU. This subsystem usually generates the actual data to be rendered, providing geometry operation—that is, 3D objects transformation and translation from 3D world-space coordinates into 2D screen-space coordinates or polygon subdivision into triangles. The subsystem also handles buffering and rasterization—converting polygonal scene representation to pixel representation, drawing screen-space primitives into the frame buffer, performing screen-space shading, and executing per-pixel operations. These operations guarantee real-time rendering of complex 3D scenes. To display realistic scenes, the ability to perform fast texture-mapping functions is crucial. A state-of-the-art multipipe graphics subsystem can help drive a single screen with up to 250 to 300 million triangles per second of a sustained performance and 7 to 8 billion pixels per second.

**Interaction tasks.** Navigation, object selection and manipulation, and system status changes all refer to interaction with the IVE. At the same time, these tasks enhance the sense of presence and immersion inside the virtual world. Obviously, we must respect real-time constraints, and interaction techniques should not affect navigation smoothness or the virtual world’s immediate reaction to user requests.

Navigation is one of the basic interaction tasks in an IVE. It lets users move from one place to another (travel or motion) or helps the mental or cognitive process of guiding movement (wayfinding), which could be more difficult in a virtual world compared with the same process in the real world. Navigation is the aggregate task of the cognitive element and the motor element, wayfinding, and travel. We can implement travel through passive transport; for example, tapping the place to reach on a map can cause the consequent motion toward the target, or through an active transport guiding the motion directly—for example, clicking an arrow button for each step in a certain direction. Also, we can perform maneuvering (a subset of motion relating to adjusting position, orientation, or perspective even if the observer is not traveling) through buttons for modifying angle-of-view orientation (heading, pitch, and roll). We could also provide the ability to select the speed of movement. Navigation is extensible to the ability to move not only in space, but also through time. This is a powerful feature in applications regarding VH, where the chance to visit the same environment through ages or to see the changes of an artwork through time from the original state to the restoration adds value to the reconstruction.

Object selection and manipulation are respectively the specification of virtual objects and specification of virtual object properties. For example, our interface provides object selection, rotation, and zoom in and out.

System controls refer to the possibility to change the system state or the interaction mode. For example, in our implementation we provided the ability to change the drawing mode (from wireframe to solid, or to with or without textures).

**Pervasive computing help.** Currently, while VR offers an increasing number of possibilities for experiencing new methods of interaction, it’s limited by highly specialized user terminals. This has negative consequences on learning time and usability, because interaction tools and devices require active attention, diverting from VR contents. Moreover, specialized devices (such as gloves, HMDs, BOOMs, and 3D joysticks) are often felt as invasive, cumbersome, and limiting because they’re usually wired to stationary appliances. Furthermore, advanced VR technologies are expensive and cannot benefit from the consumer market effort in optimizing product ergonomics and usability.

We can obtain important benefits, instead, with the introduction of pervasive computing techniques, which address these kinds of problems. For this reason, in our system we use wireless connectivity to achieve freedom of...
movement for the user. At the same time, personal devices (such as PDAs and, in the future, other wireless handheld devices such as cellular phones) are perfect candidates to support advanced interfaces. The main advantage of using PDAs is that they’re already in widespread use with the prospect of further diffusion in the market. The convergence of PDAs and mobile phones will result in more natural communication among people. They’ll be able to send text, images, and other multimedia data; surf the Web; listen to music; and watch a video through unique devices representing the natural transparent interfaces with digital multimedia data.

From this point of view, PDAs are suitable devices for our purposes because they’re more powerful and manageable compared to mobile phones or pagers in collecting, organizing, interacting, and managing multimedia data. Another requirement is to design the system with platform independence, to support different kinds of handheld devices with different hardware capabilities, thus enabling users to exploit their own personal device.

Multiclient cooperation
Supporting concurrent multiuser interaction for networked environments, as in a virtual theater, and providing exchange and collection of multimedia data can be of great interest in education, entertainment, and collaborative work. In the VH field, it’s particularly useful for making a hypothesis of reconstructions and to exchange multimedia data among people. These features imply the development of peer-to-peer interaction among clients and scheduling protocols to organize the priority in controlling the VR.

When several people—either close or separated in proximity—share the same workspace, they have two main needs: to be aware of other users and their actions and to actively interact with them. Awareness of others, when they’re not actively communicating with each other, is essential in shared virtual environments where only the view of one user at a time can be projected. In particular, it’s important to provide the representation of each user’s current position in the system and to make available information on what another user is doing, whether or not that particular user is currently controlling the system. Cooperating requires communication and exchange of multimedia data among clients—for example, opening private communication channels and exchanging or sharing data such as voice, sounds, music, video, graphics files, charts, and images.

Multimedia data support
Another requirement in a productive work environment is to have easy access to tools for taking notes, sending and receiving emails, or browsing the Internet. In fact, a multitude of different media is required for a more effective collaboration, because each way to present and organize data has its own special benefits. In other words, we need a portable terminal that runs an immersive application to allow friendly, active participation and to visualize and insert multimedia data and comments. We’d like such a device to provide the possibility of establishing relationships between data and models, to collect and retrieve data, and to make a more accurate analysis. Moreover, it’s useful for these terminals to provide users with access to the network database available on the Internet.

System implementation
We developed our hardware and software architecture in accordance with the requirements described in the previous section. The system is essentially composed of a server and user terminals (see Figure 2). The server infrastructure supports the visualization of the VH world, namely,

- high-resolution, wide-angle stereoscopic visualization;
- high-throughput rendering of still and motion pictures; and
accurate modeling of the environment with many levels of detail (obtained from modeling tools).

Client support is provided by pervasive computing devices, with the following characteristics:

- low weight and suitable form factor for portability and/or wearability,
- untethered connection for complete freedom of movement,
- low cost and wide availability, and
- highly familiar and rich user interfaces (including audio and video).

A wireless network connects the client and server infrastructures, providing adequate bandwidth and latency. In our case, latency should be in the tens of milliseconds range to achieve the sense of immersion, while bandwidth should be in the megabits per second range to support streaming multimedia data transfer (especially for video). Lower bandwidth (tens of Kbytes per second) can be tolerated in a reduced-functionality environment, where video streaming isn’t supported on the clients. Fortunately, the market offers technologies and devices meeting the required hardware and software specifications.

Targeted scenarios

On the software side, our system is a multi-client-server application, which provides three possible usages: single user, many to many, and peer to peer.

For single users, the current implementation provides each user entering the system with stereoscopic visualization, IVE interaction and navigation controls, and heterogeneous data storage and organization. The server infrastructure provides stereoscopic visualization. This infrastructure is composed of an immersive projection screen; real-time rendering libraries; and 3D high-definition models of environments, buildings, and objects. The system also provides the user with the ability to interact on a basic level with the IVE through a 2D GUI, which is nonintrusive because it’s displayed on the mobile terminal, apart from the 3D visualization.

Usability, which is a main requirement in VH because of the possible users’ heterogeneous background, is provided to the user via a touch screen, pen, and a simple GUI on the palmtop computer. Navigation controls include clickable arrows and zoom controls implemented through buttons and other features selectable through easy-to-use menus. Figure 3 shows a way to select objects or to highlight object details through menu items. A 2D map representing the 3D world achieves many goals. First, in wide IVE—for example, representing an ancient built-up area—a 2D map provides a quick overview of the virtual environment and its objects, helping users acquire spatial knowledge, a fundamental requirement for wayfinding. Through the map, users visualize their position and orientation in the virtual world and which regions can be reached from the current position. Moreover, by outlining objects or viewpoints on the map, the virtual world modeler can guide users to the best detailed part of the IVE, or the archaeologist can suggest the most historically relevant part of the reconstruction. Clicking on an object or a viewpoint represented on the map can have many effects: select object, change point of view, or display multimedia data related to the object on the PDA.

Figure 3. Examples of the GUI. (a–b) Ancient Appian Way Project showing menus that highlight object details, arrows and buttons to manipulate the object, and a 2D map to select models. (c) PDA GUI for navigation inside the Curia Iulia (created by the University of California, Los Angeles, Cultural VR Lab) representing user positioning through viewpoint.
PDA use enables implementation of advanced features such as the ability to

- take notes in a text editor;
- access the Internet through a wireless connection; and
- show text, videos, images, and sounds inherent to the 3D models.

This is extremely useful in the cultural heritage field. For example, archaeologists can use a PDA to directly insert and catalog notes and physical evidence in situ during the excavation process. Then, through the same device they can manage the virtual experience, facilitating the process of establishing relationships between data and models, collecting and retrieving data, and making a better archaeological analysis. For example, while analyzing a 3D model of an artifact on the main display, archaeologists can compare it with a short movie of the real artifact directly on the PDA without occluding the 3D model visualization.

For many-to-many interaction, we designed the system considering the multiclient presence and cooperation inside the IVE. During the process leading to the final 3D models of the virtual reconstruction of cultural heritage materials, we integrated many fields of expertise to avoid the risk of realizing misleading VH. The recreated VEs must be connected with all the background research material collected by all the people connected to the virtual reconstruction activity, helping to establish cooperation among archaeologists, computer scientists, historians, humanists, and so on. Thus, the resulting models are not only realistic, but also historically accurate.

In our system users can select an object, rotate it, and zoom in and out. While manipulating objects they can show and comment on the reconstruction details to exchange ideas and suggest corrections or add-ons. All types of data and materials can be displayed on each person’s PDA without disturbing the immersive shared projection. Only one user at a time controls the camera’s point of view because the scene is shared. The same rule applies to object manipulation. The client who controls the camera isn’t necessarily the one who is interacting with a certain object. In this case, the set of possible interactions is limited to changes in attributes—for example, the user can highlight the object or change its color, but can’t zoom in or out, a typical task needing camera control. Thus, the first person who gains camera or object control maintains it until she or he explicitly releases it. Each user is provided with a 2D map of the position of all clients connected to the system, letting them track the status of each client’s activities and know who is the master. Clicking on a master icon corresponds to opening a dialog box to request control. The master can then either accept or reject the request.

In a team of collaborators using peer-to-peer communication and, in particular, in a multidisciplinary application field such as VH, it’s possible that user subgroups or single users need information not useful or interesting to the rest of the group. Hence, we gave our system the ability to establish a private communication channel, implemented through a whiteboard displayed on the PDAs of the users involved in the communication (see Figure 4). As stated previously, PDAs use intrusive interfaces diverting the users from the visualization. The user who decides to open a private channel with a peer user sends her or his request to the other client by, for example, clicking on the other’s representation on the 2D map. Depending on the answer, the communication will be held or not held.

Design

The system design takes into account platform heterogeneity (typical of a distributed system), independence from visualization software, and network communication efficiency. Concerning the first characteristic, we wrote the various application components in their highest level in Java and exploited distributed computing libraries. Java, in fact, guarantees platform independence. We can benefit in the future from cutting-edge programming solutions for small devices coming from the Java development community.

With reference to independence from visualization software, the system performs rendering by calling graphical and visualization libraries.
To achieve independence from the specific set of libraries, the application associates a numerical tag to each interaction task (navigation controls, object manipulation, and so on). The tag identifies the remote procedure to call, transparently from the procedure implementation. Thus, changing the graphical library requires only minor changes to the code.

As a third point, targeting good network performance is critical in real-time environments, where unforeseen delays, packet loss, and so on can affect networking. Our system uses TCP/IP for transmitting packets reliably. Due to tags, communication among system components is efficient because it’s reduced to the exchange of a limited number of bytes for each interaction request. Our tests prove the achievement of appropriate timing. Moreover, to achieve efficient network communication, we performed many optimizations of communication patterns. An example is the technique applied for managing consistency maintenance among the clients and with the server. Verifications showed that having all clients poll the server for updates is faster than having the server send update events to each client as new events occur. In the latter solution the update is a sequential task, while in the former it’s concurrent.

We divided our distributed system’s architecture into modules, and the communication among them is mainly based on Java Remote Method Invocation (RMI).

We organized the server architecture (see Figure 5a) into the following modules:

- scheduler;
- avatar manager, which provides updates of the status of users interacting inside the VE; and
- theater manager, which is employed in the virtual theater scene updates procedure and invokes the graphical APIs needed to modify the VE.

The scheduler is in charge of accepting new clients into the system. The avatar manager manages the avatars to periodically update the status. When a new visitor subscribes, the server instantiates an avatar, assigning a unique identifier to it, giving the client and avatar a reference to each other. The theater manager module is in charge of retrieving the position and orientation of each avatar and passes this information through the Java Native Interface (JNI), the high-performance graphical APIs (for example, Multigen-Paradigm Vega and SGI OpenGL Performer) needed to perform actions or other C/C++ functions specifically created to modify the scene and the application’s graphical state.

We organized the client architecture (see Figure 5b) into the following core modules:

- screen manager,
- event manager, and
- communication manager.

We implemented the screen manager to update and handle the GUI on the mobile platform (for example, the PDA). The user interacts directly with the GUI, generating events. The events are passed to the event manager, which translates them into actions for both the communication manager and the screen manager. In fact, the GUI reconfigures itself depending on the general system state, for example, displaying the correct part of the 2D map in which the user is navigating, the icons which indicate other users, or enabling only the controls given to a certain user at a specific moment. This reconfiguration is often needed, especially when the server sends data to refresh the local user copy. At the same time, however, the communication manager interacts with the event manager. The communication manager communicates with the resident objects on the server through remote procedure calls. In practice, the communication manager is responsible for the calling of proce-
dures and functions corresponding to the events that a user generates. Usually the communication manager handles the remote reference of her or his own avatar and recovers the theater manager’s remote stub when the user wants to obtain virtual theater control.

Testing the system: A case study

We’ve applied the IVH system presented here to many VH scenarios. One such scenario is the reconstruction of eight tombs along the Ancient Appian Way. We exploited this particular implementation to test the system in terms of usability and performance. The implementation provides an immersive virtual environment for collaborative work in cultural heritage and archaeology fields featuring these advanced characteristics:

- shared visualization in an immersive environment,
- peer-to-peer communication,
- availability of all needed multimedia data, and
- user friendliness to help nonexpert users.

In a typical virtual visit along the Ancient Appian Way (see Figure 1) through the PDA, the user can navigate a 2D map, select the tombs (see Figure 3a), and manipulate them. The user can switch the visualization from among different reconstructions (see Figure 6a), while the system contextually displays text (see Figure 6b), images, and other multimedia data on the PDA. Moreover, the user can exchange data with other users (see Figure 4b). Each client can identify the virtual position on the 2D map of the other users along the Appian Way. Clicking on a colored dot, which represents another user, opens a private communication channel with that person (for example, a whiteboard session as in Figure 4b). While the user interacts with VR models through our GUI, many different applications can be loaded concurrently and independently on the PDA, for example, to retrieve, display, and modify data collected and stored during the excavation.

Performance assessment

One of the main requirements for an effective VR system is real-time performance. Real-time interaction and rendering increase the sense of immersion and presence inside the VE. Since the part of the system responsible for real-time rendering is a commercial platform for professional use, the real-time graphics performance is fully reached. Thus, our tests are aimed at verifying the communication among system components to guarantee an efficient multi-client and peer-to-peer interaction in small user groups. In fact, our system isn’t targeted for applications needing fast interaction in massively charged networks such as multiplayer networked games. Because RMI has nonnegligible overhead with a standard TCP/IP connection, we tested the average time needed by the RMI to call a method on a remote host through the wireless network to verify whether the system matches the requirements.

System hardware setup

The interactive, multiuser system is a combination of two main parts. The CINECA Virtual Theater (see http://www.cineca.it) consists of

- an SGI Onyx2 engine with three graphic pipelines, each with 64 Mbytes of texture memory and 8 MIPS R10000 processors;
- a cylindrical screen;
- three projectors;
- active stereo shutter glasses; and
- consoles to control the visualization.

The second part is the infrastructure for connecting the portable terminals to the network (the base station, wired to the CINECA’s LAN, and the network interface cards) and the portable...
terminals themselves. The software’s server side runs on the system’s server part (the CINECA Virtual Theater based on the SGI Reality Center) while the client software runs on the wireless pocket PCs.

Specifically, we chose the Compaq iPAQ as portable terminals. The model on which we tested our application has a 320 × 240 resolution thin-film transistor color screen, 32 Mbytes of RAM, 16 Mbytes of ROM, and runs Windows CE 3.0. At the start of this project we had the fastest processor available on the market (a 206-MHz Intel Strong Arm 32-bit RISC). iPAQs are expandable with the needed hardware to connect through various communication interfaces (such as IEEE 802.11b, General Packet Radio Service, Global System for Mobile Communications, and Bluetooth).

We equipped the terminals with a Cisco wireless card to exploit the IEEE 802.11b standard wireless LAN interface working at 11 Mbits per second. This choice is reasonable because our application is meant for an indoor environment and the costs of the infrastructure were reasonable (that is, no additional costs during connection time).

System software setup

On the server side, we loaded the Java Runtime Plug-in v1.3.1 for SGI IRIX on top of the IRIX 6.5.14 operating system. The Java application calls the high-performance multiprocess rendering graphic libraries (Vega, Performer, and OpenGL) through JNI, which links to an intermediate C layer. On the client side, the Java Virtual Machine we used is Jeode Runtime from Insignia Solution, optimized for the Windows CE platform. This JVM supports Java Abstract Window Toolkit classes and RMI and JNI technologies. Windows CE provides the possibility of developing software using specific APIs to write code for PDA devices. We benefit from this when using JNI in our Java application to call the Windows CE API to concurrently run another application, such as an Internet browser, calling it directly from our system (selecting an item on a menu in our GUI).

Experimental results

We ran experimental tests to validate the system in terms of client side usability and interaction, the place at which the system is most limited in hardware and software resources. In this implementation, a client interacts with the shared IVE by sending requests or events to a server. Some events, such as those related to the awareness of other clients, have to be distributed to all clients. Our first test studied this processing activity: essentially we examined how the server manages the distribution of an increasing number of events per second as the number of clients goes up. The scenario we considered was the movement of a client on the 2D map in the GUI (on the PDA). In this case, the server manages the event to display the motion on all the clients’ 2D maps. The data packet sent on the network is the set of coordinates indicating the new position. We considered two procedures for managing the process:

- **Sequential version.** Any update event generates a thread on the server, which is in charge of sequentially contacting every client to refresh its status.

- **Concurrent version.** Any update event changes each client’s status on the server. On each client a dedicated thread concurrently queries the server for status update.

Both solutions provide an acceptable delay due to the small size of the packet sent (see Figure 7). Our system implemented the second solution because it provides better overall performance. In addition, the plot shows that the concurrent implementation maintains an update time of less than 100 milliseconds with more than 15 users. This is a satisfactory delay for an interactive application. In general, a recommended timing of less than 150 to 200 ms must be achieved when a system provides visual and auditory feedback.8

We performed our second test, designed to assess wireless network performance, by calling a huge number of dummy functions (mainly to send nonsignificant packets back and forth). We ran the test on a number of different platforms to compare the results and verify system portability (see Figure 8). The mean system response time of under 50 ms is maintained when the packets
 exchanged are less than 10 Kbytes and under 250 ms for packets of 80 to 100 Kbytes. These results are more than acceptable. We found unusual behavior in the form of a significant drop in performance when using packets of around 1,000 bytes on the PDA. Our hypothesis is that this is an artifact of the pocket PC JVM and might be due to the marshalling process inefficiency.

A final test attempted to evaluate what happens when multiple peer-to-peer communication channels are in use. In particular, the test measured the mean time needed to send an array of Java point type elements (indicative size was 9 bytes × array size) from one client to the others. We varied the number of clients and the array dimensions. Each client tries to write concurrently to the others 50 times.

Figures 9 and 10 show the results of tests performed with one, three, five, and ten clients attempting to send between 30 to 1,000 points. This test differs from that relating to Figure 7, as the size of data packages varies and the communication is between clients. According to some authors, end-to-end delay should not exceed 1,000 ms and a delay of less than 200 ms is recommended for tasks needing tight synchronization. The results indicate that the mean time becomes excessively long when the number of clients is greater than three to five and the data packages contain more than 70 to 100 points. This is still an acceptable result in VH workgroups, but is likely to be a major limitation to adding concurrent features to the PDA-based part of the application. The current system uses Java RMI and it’s possible that we might obtain performance improvements by adopting other communication paradigms, as RMI is a high-abstraction-level programming paradigm and limits protocol-level optimizations.

In conclusion, quantitative results demonstrate that the system is adequate for interaction with VH for small user groups (four to twenty) depending on the usage of peer-to-peer interaction features, which are the most limiting in terms of performance. Therefore, we’ve achieved our objectives for our system. This evaluation is confirmed by user feedback collected informally during demonstration sessions at CINECA, where the virtual theater based on SGI Reality Center technologies and the PDA setup described previously is used. Users accessing the demonstration sessions had different technological backgrounds, ranging from extensive computer literacy to no technical experience. All of them considered the PDA and its GUI easy to use as a wireless interaction device and appreciated the connections offered between content on the PDA and the 3D world. Some users provided sugges-
tions for future development of the system or possible features to add to the GUI. Future work will address a formal usability evaluation study.

Future work

Future work will explore different directions. First, we’re defining a set of APIs to simplify the process of application development and to improve multiuser communication. Furthermore, we’ll develop enhanced multiuser features to augment peer awareness of other peers and peer-to-peer exchange of multimedia data. We plan to add the ability to configure each client’s status, letting the clients choose the degree of awareness that others have of them. We have many technical and social implications to consider regarding awareness concerns. We’ll combine this work when we optimize the system’s performance.

We’re already exploring multimodal interaction to combine the benefits of more than one modality of I/O and to study the challenges of integrating different interaction devices. Currently, we’re integrating novel sensors into our system, starting with accelerometers to experiment with gesture interfaces.

Acknowledgments

We thank Marco Gaiani (Politecnico di Milano), scientific director of the Ancient Appian Way 3D Web Virtual GIS project, who supported our research by giving us models and contents on which we based our case study and Bernard Frischer, head of the Cultural VR Laboratory at the University of California, Los Angeles, for letting us use the Curia Iulia model.

References


Elisabetta Farella is a PhD student at the Department of Electronics, Computer Science and Systems at the University of Bologna, Italy. She is also a research consultant at the CINECA Supercomputing Center. Her research interests include VR systems, cultural heritage, pervasive computing, and human-computer interaction. Farella has a BS in electrical engineering from the University of Bologna.

Davide Brunelli is a PhD student at the Department of Electronics, Computer Science and Systems at the University of Bologna. His research interests include ubiquitous computing oriented to cooperation through mobile devices and development of graphic applications for portable systems. Brunelli has a BS in electrical engineering from the University of Bologna.

Luca Benini is an associate professor at the Department of Electronics, Computer Science and Systems at the University of Bologna. He also holds visiting researcher positions at Stanford University and Hewlett-Packard Laboratories, Palo Alto, California. His research interests include all aspects of computer-aided design of digital circuits, with special emphasis on low-power applications, and in the design of portable systems. Benini has an MS and a PhD in electrical engineering from Stanford University. He is an IEEE senior member.
Bruno Riccò is a full professor at the University of Bologna. His research interests include solid-state devices, microelectronics, integrated circuit design, evaluation and testing, multimedia systems, and human–computer interaction. Riccò has a PhD from the University of Cambridge, UK. He is an IEEE Fellow.

Maria Elena Bonfigli is a technical researcher at CINECA Supercomputing Center. Her research interests include VR and human–computer interaction applied to cultural heritage. Bonfigli has a BS in computer science and a PhD in history and computing from the University of Bologna.

Readers may contact Elisabetta Farella at the Dept. of Electronics, Computer Science and Systems, University of Bologna, Viale Risorgimento, 2, 40136, Bologna, Italy; efarella@deis.unibo.it.