MULTIUSER COLLABORATION WITH NETWORKED MOBILE DEVICES

Kam S. Tso, Ann T. Tai, Yong M. Deng, IA Tech, Inc., Los Angeles, CA

Paul G. Backes, Jet Propulsion Laboratory, Pasadena, CA

Abstract

This paper describes a multiuser collaboration infrastructure that enables multiple mission scientists to remotely and collaboratively interact with the visualization and planning software, using wireless networked personal digital assistants and other mobile devices. The infrastructure includes (1) human-computer interaction components to provide natural, fast, and accurate inputs, (2) a communications protocol to ensure reliable and efficient coordination of the input devices and host computers, (3) an application-independent middleware that maintains the states, sessions, and interactions of individual users of the software applications, and (4) an application programming interface to enable tight integration of applications and the middleware. The resulting technologies not only are applicable to NASA mission operations, but also useful in other situations such as design reviews, brainstorming sessions, and business meetings, as they can benefit from having the participants concurrently interact with the software applications (e.g., presentation applications and CAD design tools) to illustrate their ideas and provide inputs.

1 Introduction

During ground operations of planetary rover and lander missions, scientists need to meet daily to review downlinked data and plan science activities. For example, scientists use the Science Activity Planner (SAP) in the Mars Exploration Rover (MER) mission to visualize downlinked data and plan rover activities during the science meetings [1]. Computer displays are projected onto large screens in the meeting room to enable the scientists to view and discuss downlinked images and data displayed by SAP and other software applications. However, only one person can interact with the software applications because input to the computer is limited to a single mouse and keyboard. As a result, the scientists have to verbally express their intentions, such as selecting a target at a particular location on the Mars terrain image, to that person in order to interact with the applications. This constrains communication and limits the returns of science planning. Furthermore, ground operations for Mars missions are fundamentally constrained by the short turnaround time for science and engineering teams to process and analyze data, plan the next uplink, generate command sequences, and transmit the uplink to the vehicle [2]. Therefore, improving ground operations is crucial to the success of Mars missions.

The multiuser collaboration infrastructure enables users to control software applications remotely and collaboratively using mobile devices. The infrastructure includes (1) human-computer interaction techniques to provide natural, fast, and accurate inputs, (2) a communications protocol to ensure reliable and efficient coordination of the input devices and host computers, (3) an application-independent middleware that maintains the states, sessions, and interactions of individual users of the software applications, and (4) an application programming interface to enable tight integration of applications and the middleware. The infrastructure is able to support any software applications running under the Windows or Unix platforms. The resulting technologies not only are applicable to NASA mission operations, but also useful in other situations such as design reviews, brainstorming sessions, and business meetings, as they can benefit from having the participants concurrently interact with the software applications (e.g., presentation applications and CAD design tools) to illustrate their ideas and provide inputs.

2 Related Work

Research efforts on human-computer interaction have shown that collaborative work can improve productivity and a number of approaches have been proposed.

An ideal interface would allow multiple people to comfortably collaborate directly on the display surface. Smart Boards [3] and other touch sensitive

surface can allow one or two people to interact directly with the large presentation screen, but other meeting participants must resort to manipulation by proxy — trying to describe which objects to manipulate and having the user in front of the screen do it.

A popular approach that allows more people to point at objects on a large screen is to use a video projection screen with a camera-tracked laser pointer [4]. The laser point on the screen is captured by a video camera, and its location is recognized by image processing techniques. The behavior of the point is translated into signals sent to the mouse input of the computer, causing the same reactions as if they came from the mouse. However, interactions using laser pointers tend to be imprecise, error-prone, and slow. Although there is effort to correct the imprecise pointing due to misaligned projector placement by automatic keystone correction [5], the time delay seriously affects the interactions.

Early approaches for multiuser inputs use expensive and special-purpose hardware or use specially constructed meeting rooms. For example, MMM (Multi-Device, Multi-User, Multi-Editor) [6] is one of the first single display groupware environments to explore multiple mice on a single display. MMM handles up to three mice. It has a single tool palette and color palette, and a separate home area to show each user's name, current color, and drawing mode. MMM only supports editing of text and rectangles, and uses color to distinguish between the users.

As the popularity of personal digital assistants (PDAs) such as the Palm OS and Pocket PC handhelds grows, researchers have investigated their use as generic input devices to enable collaborative work on custom applications. For example, the M-Pad system [7] supports multiple users collaborating with Palm OS handhelds and a large whiteboard. Each user carries their own Palm OS handheld that acts as a holding area for personal options and new content as it is created. Users can then copy this new content to the whiteboard when it is complete and ready for public viewing. The Remote Commander and PebblesDraw [8] also use

handhelds to enable multiple users to collaborate simultaneously. The Remote Commander allows strokes on the main display area of the Palm OS handheld to control the PCs mouse cursor, and for Graffiti input to emulate the PC's keyboard input. This allows multiple users to input to existing PC applications. However, since the window has only one cursor (the input focus), the users have to take turn to input to the application and social protocols are relied upon to control whose turn it is. The PebblesDraw is a custom application which allows all users to have their own cursors so that they can input simultaneously.

The major limitation of the existing approaches to collaborative interactions in a meeting room is that their capability is restricted to customized applications which are specially developed for collaboration. In contrast, our research aims for developing basic collaborative capabilities that can work for all existing software applications, without the need of modifying the applications. Our multiuser collaborative infrastructure works on any computer platforms that support Java, thus covering most software applications being used. Another limitation of the existing approaches is that they only allow multiple users to take turn to interact with the applications. Since our middleware maintains the states and sessions of individual users to enable multiple simultaneous inputs to different windows of the same application or different applications, we allow generic sharing of applications by multiple users. The third limitation of existing approaches is the inaccuracy and latency associated with the inputs. This research addresses the human-computer interaction issues to ensure that user inputs will be accurate and natural.

3 Multiuser Collaboration Infrastructure

The multiuser collaboration infrastructure, illustrated in Figure 1, consists of the Multiuser Collaboration Controller (MCC) software which runs on the mobile devices, and the Multiuser Collaboration Middleware (MCM) which runs on
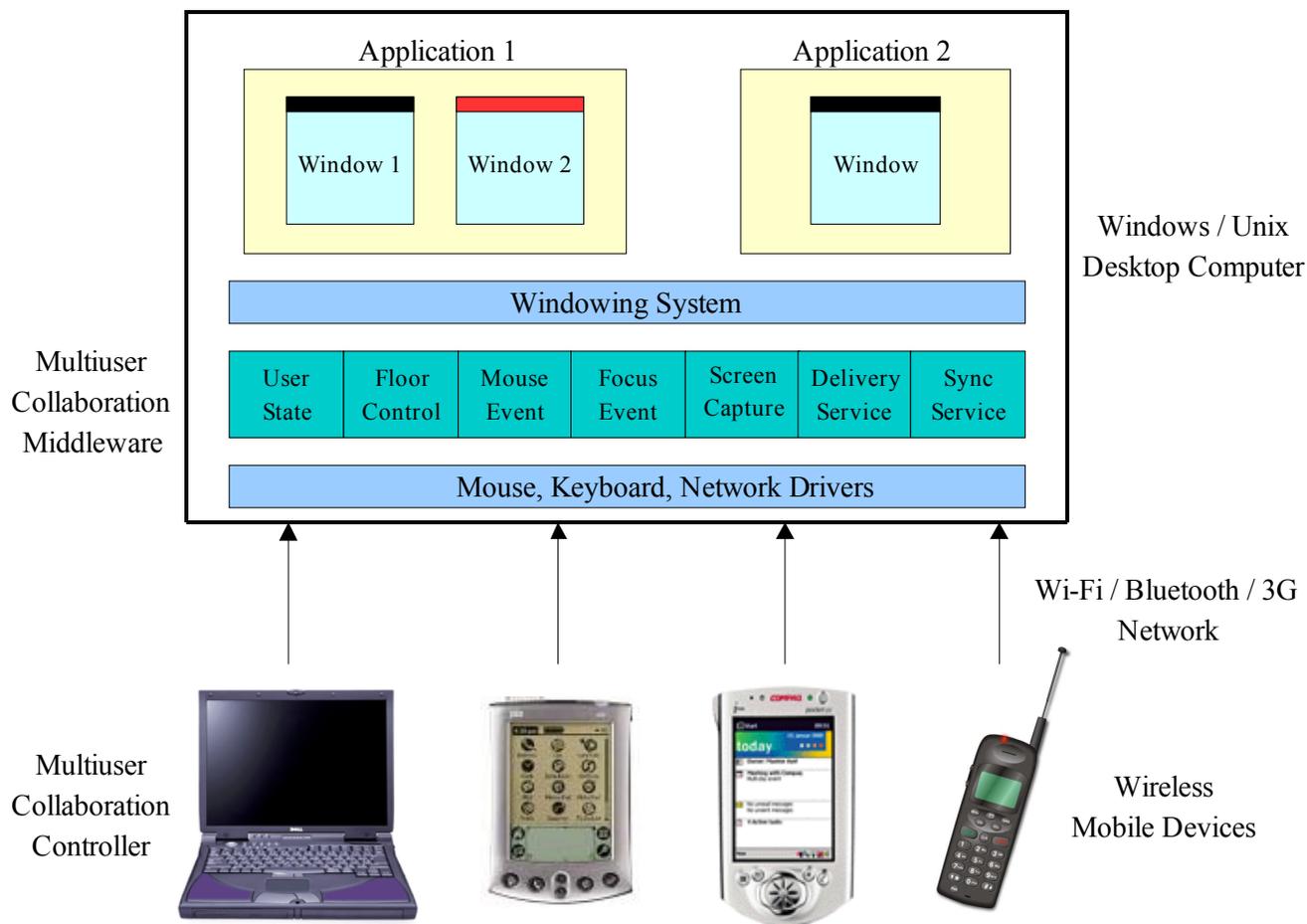
Figure 1. Multiuser Collaboration Infrastructure

the desktop computers. Together, they form the Multiuser Collaboration Suite (MCS) which enables multiple users to remotely control and collaborate on desktop software applications using their personal digital assistants (PDAs) and other mobile devices.

The users interact with MCC running on their devices to move the cursor, press mouse buttons, and input keyboard strokes to the desktop computer. The MCM running on the desktop computer receives and realizes these remote control actions. In addition, MCM provides an application programming interface (API) which can be used by application developers to provide tightly integration of the mobile devices and the applications. MCC is developed in the Java Platform, Micro Edition (Java ME) which allows it to be run on a wide variety of mobile devices such as Palm OS and Pocket PC

PDAs, Java-enabled cell phones, and Windows or Unix laptop computers. Connectivity between the mobile devices and the desktop computers is supported by Wi-Fi, Bluetooth, and 3G Mobile Phone wireless network technologies.

4 Interaction and Collaboration Techniques

A number of human-computer interaction techniques have been developed to enable natural, fast, and accurate interactions with host applications using mobile devices to accommodate the limitations on screen size and input capabilities. Other features such as floor control and user authentication have also been implemented to support collaboration on interacting with the host applications among the users.

4.1 Direct Mapping and Relative Mapping

The most basic interaction with a software application is moving the cursor to a desired location. It needs to be done quickly and accurately. Cursor movement can be accomplished by sending cursor motion commands to the host based on the tapping or dragging of the stylus on the device screen. There are two basic approaches to map the stylus tapping or dragging on the device to the cursor movement on the screen of the host computer running the software application: Direct Mapping and Relative Mapping.

In the Direct Mapping approach, each pixel on the device screen is mapped to the absolute coordinate of the computer screen. This approach allows the user to move the cursor from one location to another with a tap on the device screen. For example, tapping in the upper left corner of the device screen would move the cursor to the upper left corner of the PC screen. This approach is the fastest way to move the cursor and can be accomplished with the fewest commands between the device and the host computer.

In our initial development we implemented this Direct Mapping approach. However, our experience shows that it does not work well. The device screen (160x160 for Palm OS PDAs and 240x320 for Pocket PC PDAs) is much smaller than the computer screen (usually with a resolution of 1024x768 or 1280x1024). Thus each pixel on the device represented 6 or more pixels on the computer, making cursor positioning inaccurate. Moreover, since characters are often less than 10 pixels wide, this also makes it hard to select individual characters. Furthermore, the positions reported by the devices are jittery, varying by 1 or 2 pixels in all directions when the stylus is kept still, so the cursor jumps around the computer screen.

The other approach is Relative Mapping in that stylus movement across the device screen is mapped to an incremental movement across the computer screen, so the actual position where the user puts down the stylus is not important; just how far it is moved from the initial position. This is analogous to the small touchpad on some laptops. We implement Relative Mapping in MCC and find it provides accurate and smooth cursor movement. But there is also problem associated with using relative coordinates. It would take many strokes of the device if one needs to move the cursor from one side of the screen to the other. To mitigate this problem, we provide several buttons to control the speed of cursor movement. The 1x button maps each device pixel increment to each computer screen cursor increment, while the 4x button maps each device pixel increment to four pixel increments on computer screen. In 4x speed, moving the stylus across the device screen can almost move the cursor across the computer screen. We found that the combination of 1x and 4x cursor speeds enables the device users to achieve fast and accurate cursor movement and positioning. Figure 2(b) shows the MCC Main Panel. The grid area in the panel is used by users to stroke the stylus for cursor movement.

4.2 Mouse Buttons Simulation

Besides positioning the cursor, the mouse is also used to select and activate objects, drag and drop objects, activate and select popup menus, etc., in software applications that have a graphical user interface. The functionalities require the device be able to send *press*, *release*, *click*, and *double-click* of the left and right mouse buttons. Some Windows and Unix applications also use inputs from the middle mouse button and mouse wheel.

Most PDAs has four built-in hardware buttons to launch commonly used applications and a directional pad for cursor movement and object selection. We make use of the hardware buttons for generating mouse button events and the directional pad for mouse motion. Using this method, MCC programs hardware button #1 as the left mouse button. Mouse button events such as press, release, click, and double-click are generated when the hardware button is pressed, released, clicked, and double clicked. Mouse dragging is achieved by stroking the stylus while the button is pressed. Hardware button #2 is programmed as the right mouse button and similarly generates the right button press, release, click, and double-click. The remaining buttons are programmed for generating middle button events.

The directional pad is used to generate mouse motion events. When it is pressed on the left, a mouse motion event is generated for cursor movement in the negative x direction. The number of pixels to move depends on how long the pad is
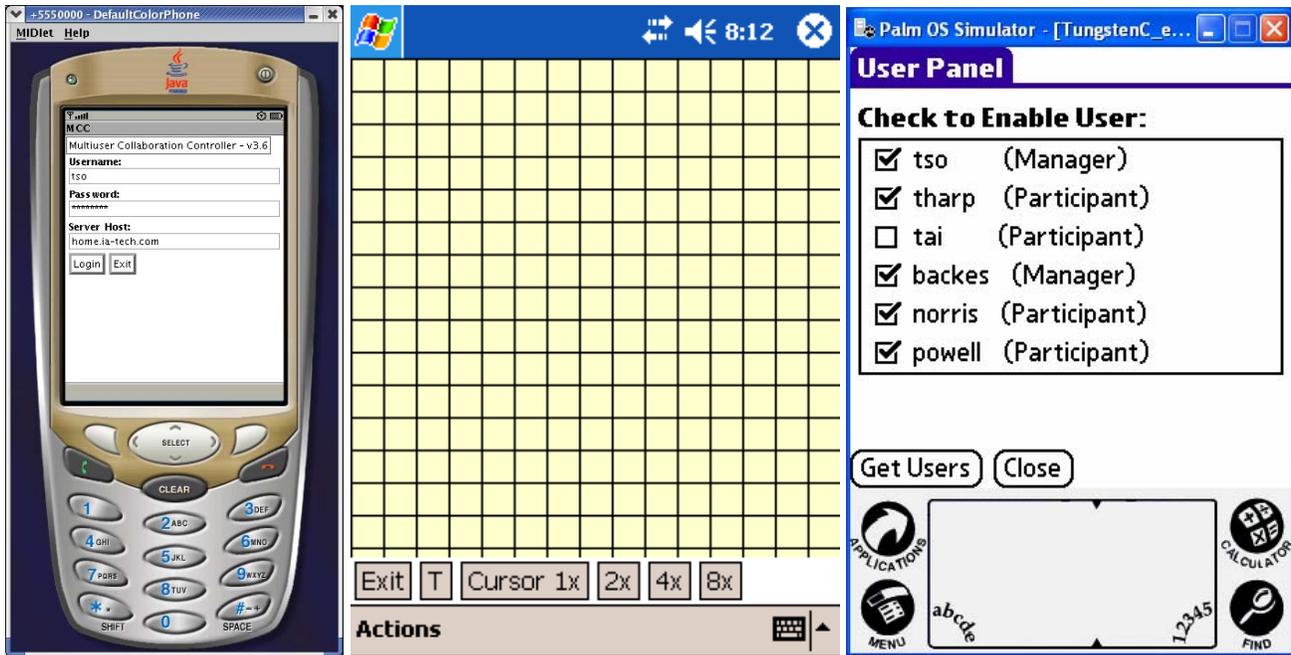
(a) Logon Panel	**(b) Main Panel**	**(c) User Panel**

Figure 2. MCC Panels on Different Mobile Device Platform

pressed. The longer it is pressed, the faster the cursor will be moved. If the directional pad is clicked, only one pixel will be moved. This enables very accurate movement of the cursor. Similarly, when the directional pad is pressed toward the right, up, and down, it generates positive x, negative y, and positive y motion, respectively. We also programmed the press of the pad at its center as a left mouse click. This can be conveniently used for object selection.

4.3 Online and Offline Text Input

We developed two methods for the user to input text to the host applications: Online-Text and Offline-Text Input.

In Online-Text Input, the user either uses the built-in keyboard or the PDA virtual keyboard to enter text directly to the host application. Every key the user inputs to the device is sent to the host application as a single character. Editing keys, such as Backspace, are also sent out as they are entered. But some key sequences, such as Shift-A, are processed locally. They are sent as a single character, 'A' in our example, instead of two keys. The Online-Text Input method is good for sending

editing keys. But it is not good for long text input when multiple users are interacting with the host application. Every time a key is entered, MCM needs to change the input focus of the host application to that of the text-input user. If another user is also working on the application, the input focus, and even the active window if they are not working on the same window, will keep changing.

The contention problem created by simultaneously inputs can be alleviated by the Offline-Text Input method. In Offline-Text Input, the user opens a Text Box in MCC to compose text input. During text composition, the user can use keys such as Backspace, Delete, Arrows to edit the text. After the user finishes composing the text, the complete text string is sent to MCM by pressing the OK button. The Offline-Text Input method allows users to compose text input on their own devices locally without interfering with the host application. Therefore, it is a fast and efficient way to input a long text. However, since the editing keys are used to edit text locally, there is no way to send editing keys to the host application for changing

the text that has already been input. Thus a combination of Online-Text and Offline-Text Input methods can best serve text inputs.

4.4 Commands Look Ahead and Collapsing

Messages carrying the Mouse Motion events are sent from the MCC to MCM at a very high rate continuously as the user strokes the stylus on the device screen. The high message rate ensures smooth cursor movement. However, when MCM is run on a relatively slow computer, Mouse Motion messages will not be processed fast enough and results in having the messages accumulated in the buffer. When this happens, the cursor movement on the host computer will lag behind the stylus movement.

We use the commands look ahead and collapsing technique to mitigate the cursor lagging problem. When MCM processes a Mouse Motion message, it looks ahead of any pending messages queued in the buffer. It reads all pending Mouse Motion messages sent by the same device and combines the move increments in the messages together to perform a single cursor move. This technique solves the cursor lagging problem, although the cursor will be seen to be a little jittery.

4.5 Floor Control for Multiuser Collaboration

Without floor control, all connected users can move the cursor, send mouse button events, and enter texts at their own will. A rogue user can dominate the control by moving the mouse continuously, or more seriously, sending inputs that could cause the application to act incorrectly. Even in a cooperative environment, it is desirable to be able to allow or disallow certain users for remotely controlling the applications when collaboration of a specific group of users is needed, or when the work load of the host computer becomes too heavy.

We developed floor control for coordinating multiuser collaboration. We first classify the users into two types: managers and participants. Participants can interact with the host application only when they are enabled by the manager. The Manager has the power to enable and disable application interactions of selected participants by

using the MCC User Panel, as shown in Figure 2(c).

When the User Panel opens, MCC gets the list of users who are currently participating in the remote control from MCM. The User Panel displays the names of users and their user types. The checkbox next to each of the user names shows whether the user is enabled for control or not. Although all users can open the User Panel to view the list of users and their status, only the managers can click the checkbox to enable or disable the participant users for control. When the checkbox of a user is clicked, a message is sent to MCM to enable or disable the control of that user. If a user is disabled, the requests from that user for cursor movement, mouse click, and key input will be ignored. The User Panel also provides a "Disable All" checkbox so that the manager can use it to enable or disable all the participant users.

4.6 User Authentication and User Type

Authentication is used to provide security and identify the MCM users. Before a user can participate in multiuser collaboration, the user needs to log onto MCM using the MCC Logon Dialog, as shown in Figure 2(a). The username and password are sent to MCM which authenticates the identity of the user against the password file. This avoids unauthorized users to participate in multiuser collaboration. The password file also contains the user type of the user. MCM can then use the user type information to support floor control.

5 Multiuser Collaboration Middleware

The Multiuser Collaboration Middleware (MCM) is the layer of software that runs on a desktop computer to enable multiple users to collaborate on applications running on the computer remotely using their mobile devices. MCM accepts connections from the mobile devices, maintains the states, sessions, and interactions of the individual users to support multiuser collaboration.

MCM is developed in Java and as a result it can be run on any host computers that support

the Java Platform, Standard Edition (Java SE), which include PCs, workstations, and servers running operating systems such as Windows, Mac OS X, Linux, and Solaris. MCM has also been developed to be application-independent in that it can support any software applications running on the host computers.

Figure 1 shows the services provided by MCM to the mobile devices and the host applications. Service requests from the mobile devices include cursor movement, mouse button events, input focus, user state, data delivery, and data synchronization. The service requests from the mobile devices are contained in messages with a specific format designed to be efficient in transmission and processing. After the request is received, MCM interacts with the operating system, windowing system, and applications on the host computer to realize the service requests.

5.1 Connection Service

When MCM starts, it spawns a thread listening to a stream socket port for connection from the mobile devices. The multi-threaded design allows MCM to service requests from other mobile devices while accepting new connections.

Once a connection is accepted, a device thread will be spawned and dedicated to handling the communications and serving the requests of that connected device. The device thread is responsible to send and receive messages with the device. It also closes the socket and cleanup its state when device disconnection is detected. Commands look ahead for speed up the handling of multiple pending requests and byte-order conversion for handing devices of different processor architectures are also handled by the device thread.

As MCM uses the stream socket TCP/IP protocol, it can accommodate different network protocols at the link and physical layers. Therefore, the devices can communicate with the host computer with wired Ethernet connection, Wi-Fi wireless connection, or Bluetooth point-to-point network connection.

5.2 User Management

MCM maintains and manages the states of the users connected for multiuser collaboration. It

supports the user logon process by authenticating the user and password against the stored password file. It also retrieves from the password file the user type of the user. MCM can also be run with the option not to authenticate users. In this case, MCM just uses the password file to retrieve the user type. In public meetings with a large number of participants and security is not a concern, MCM can also be run without a password file. In this case, anyone can log on and each is assigned as the Manager user type.

After a user is logged on, MCM assigns the connected user a unique User Id. This User Id is used to identify the user in future interactions. MCM maintains a list of User State objects for all the connected users. When the user logged on, MCM creates a User State object for the user. The User State object contains the user's name, User Id, connection, and current cursor location, active window, and input focus, and a timestamp of the last message the user sent. Since each MCC/MCM message includes the User Id in its header, MCM uses it to locate the user's User State and uses the state information to support multiuser collaboration.

5.3 Mouse/Keyboard Control

MCM services requests of mouse and keyboard actions from the connected users. The mouse events include 1) cursor movement, 2) button press, release, drag, double-click events of the left, right, and middle mouse buttons, 3) single character of keyboard input, and 4) multiple character string of keyboard input.

MCM makes use of the Java Robot API to realize the requests. The Robot API is a standard feature of the Java Platform. It provides a programmatic interface to generate native system input events. Using the API to generate input events differs from posting events to the AWT event queue or AWT components in that the events are generated in the platform's native input queue. For example, *Robot.mouseMove()* will actually move the mouse cursor instead of just generating mouse move events. MCM realize the user's mouse and keyboard requests by invoking the appropriate methods of the Robot API.

5.4 Window/Focus Management

MCM manages the active window and input focus associated with each connected user to support multiuser collaboration such that inputs from one user will not be entered into another user's working window. This feature only supports Java applications. The Java platform provides APIs for listening to window and focus events fired during execution. To make use of this feature, the application needs to instantiate the *MCMiddleware* object which will register itself to listen to all window and focus events. When the application changes its active window or input focus, an event will be generated and sent to *MCMiddleware* object. MCM will then be able to identify the nature of the event and save the active window or focused widget to the state of the user who initiated the event. Future requests from the same user will then be forwarded to the active window and input focus associated with the user.
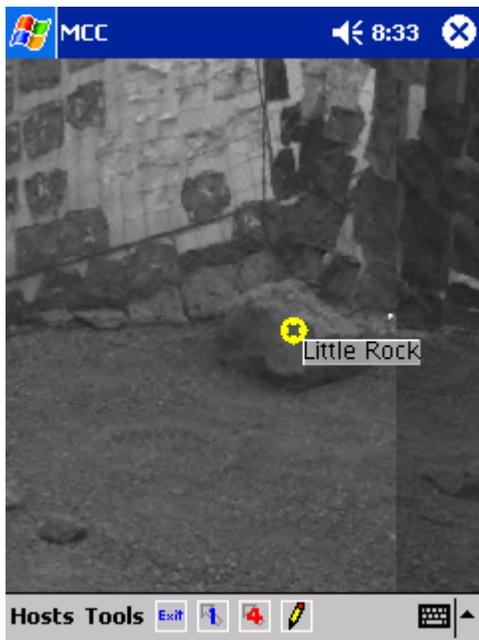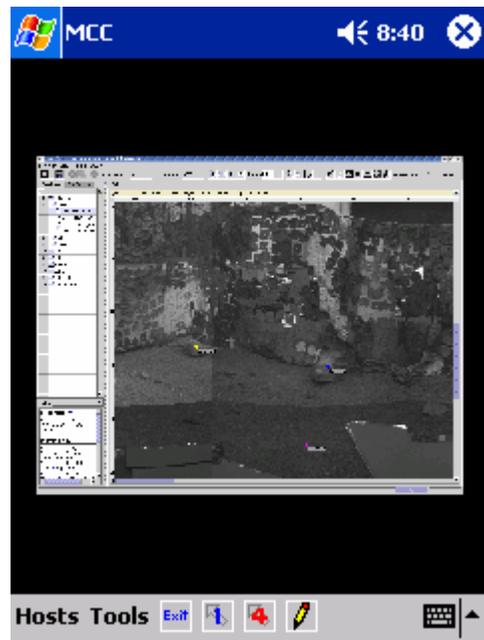
(a) Full Fidelity   (b) Whole Screen

Figure 3. Screen Capture

5.5 Screen Capture

Another feature that is important to mission scientists is the ability to capture a portion or the whole screen of the desktop and display it on the device. To capture the screen, the user first moves the host computer cursor to the location where the screen is to be captured. The user can then select one of the Screen Capture menu items to make the request. There are five menu items, each for a different scaling factor. The "1x Image" menu item requests MCM to capture the screen in the full fidelity without scaling. The same desktop screen pixels will be displayed on the device, as shown in Figure 3(a). The images can be clearly viewed on the device without any degradation. However, since the PDA has a small screen, only a small area of the desktop can be displayed.

Other menu items are provided to capture the screen with different scaling factors. They capture a larger desktop screen area and shrink it to fit into the device screen. For example, the "2x Image" captures an area of 480x640 on the desktop screen and is shrunk by a factor of 2 to display on the device 240x320 screen.

The "Whole Screen" menu item captures the whole desktop screen and the image will be scaled to fit into the device screen. Figure 3(b) is a snapshot of MCC which is displaying the whole desktop screen. The whole SAP Downlink Browser can be seen on the device, even though without the details. The "Whole Screen" capture is useful to the user for an overview of what the desktop computer is running. The user can use it to quickly move the desktop cursor to a location at which the user taps the stylus under cursor direct mapping. As discussed in Section 4.1, Relative Mapping is usually used for cursor movement. When the desktop screen is displaying the captured desktop screen, the mode for cursor movement is changed to Absolute Mapping. This allows the user to tap on the object seen on the mobile device to move the cursor on the desktop directly to that object.

6 Integration of Mobile Devices to Host Applications

MCM provides an application programming interface to support the use of MCM services by the
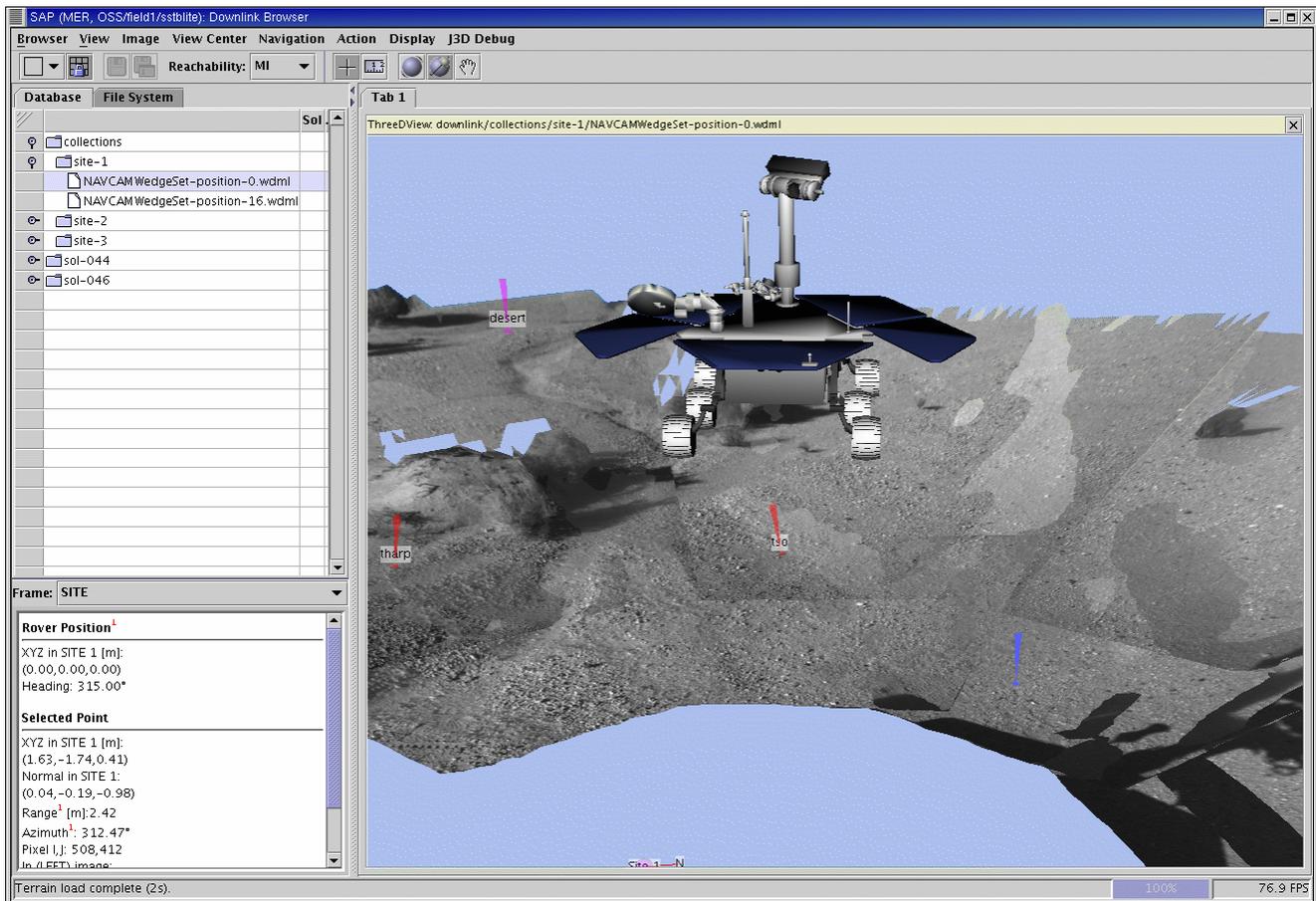

Figure 4. Navigation in the 3D View

applications. Using the API, we integrated MCM with a development version of the Science Activity Planner (SAP) and demonstrated multiuser collaboration. SAP is the primary science operations tool in the Mars Exploration Rover (MER) mission for scientists to visualize downlink data and specify desired uplink activities [1]. The integration of MCM to SAP enables the MER scientists to use their PDAs or laptops to interact with SAP during science planning.

The MCM API includes the constructors and several service methods. Before an application can make use of the MCM services, it needs to instantiate a *MCMiddleware* object. Several

constructors available to instantiate the *MCMiddleware* object for specifying a different socket port and a different debug level. The MCM methods are used to start and stop of MCM services, listen to MCM events, get a list of currently connected users, and identify the user who sends the last message for remote control, etc.

6.1 Navigation in the 3D View

The 3D View of the SAP Downlink Browser displays the 3D terrain and rover, as shown in Figure 4. Targets, features, and clicked points are also displayed. MCM enables the scientists to use their devices to navigate within the 3DView, such as *fly-over* or *walk-through* the 3D terrain. Navigation is achieved by pressing the device hardware buttons and at the same time dragging the device screen with the stylus.

6.2 Designation of Targets

MCM enables the scientists to designate targets in SAP with their names attached to the targets. This feature allows the scientists to point out objects in the panorama for discussion. Figure 4 shows two targets selected by the users and a feature.

6.3 Remote Plan Generation

The SAP Uplink Browser is used to create and edit activity plans. The left side of the Uplink Browser is an uplink selection tree that allows the user to load a previously saved plan. A plan is opened from the selection tree by double-clicking on it. The right side of the Uplink Browser displays the activities specified in the plan, and additional views such as resource consumption plots. The MCM enables the scientists to interact with the Uplink Browser remotely using their devices to select plans, enter activities, and specify arguments.

6.4 Multiple Cursors

Moving the cursor in the desktop computer remotely is the most used MCM service. However, the window system, whether it is Window Manager under Microsoft Windows or X Window under Unix, only displays a single cursor on the computer screen. As a result, after a user places a cursor on an object, the cursor will be moved away when another user sends a mouse move message. This is undesirable because the users cannot see their current cursor locations and they could lose track of what they were doing.

MCM enables multiple cursors to be displayed by maintaining the state of the individual users after they have connected to the middleware. The user state contains the user's name, User Id, connection, last cursor location, active window, input focus, and a timestamp of the last message the user sent. Since each MCC message includes the User Id in its header, MCM can use it to locate the user's state and use the state information to support the display of multiple cursors. When a user sends a mouse move message, MCM records the new cursor location and the user state and displays a cursor, labeled with the name of the user, on that location.

7 Conclusions

This paper describes the development of a multiuser collaboration infrastructure which enables the scientists to remotely interact and collaborate on the software applications using personal digital assistants and other mobile devices during mission planning.

With the availability of low cost, light weight, and small size mobile devices and wireless networks, meeting rooms have become technology rich. And very often, meetings are conducted with a software application as the focus of discussion. It could be the Microsoft PowerPoint in presentation, a software whiteboard in brainstorming, or a CAD tool in design review. The multiuser collaboration infrastructure enables multiple users to share, interact, and collaborate on one or more software applications in these technology-rich spaces.

MCC benefits from being developed in Java ME because Java is widely supported by the wireless service providers and cellular handset manufacturers. The strong Java support enables MCC to run on the next generations of cell phones. As the wireless service providers support *over-the-air* provisioning, a service that allows the end users to add functionality, like games, utilities, and business applications, directly "over-the-air" to the cell phones, MCC can be distributed by these providers as a service. We have also implemented

an MCM Java applet which allows it to be downloaded and run on any web browser.

References

[1] J. S. Norris, M. W. Powell, M. A. Vona, P. G. Backes, and J. V. Wick, "Mars Exploration Rover Operations with the Science Activity Planner," in *Proceedings of IEEE Aerospace Conference*, (Big Sky, MT), Mar. 2005.

[2] F. P. Seelos *et al.*, "FIDO Prototype Mars Rover Field Trials, May 2000, Black Rock Summit, Nevada," in *Proceedings of the 32nd Lunar and Planetary Science Conference*, (Houston, TX), Mar. 2001.

[3] SMART Technologies, *SMART Board 580*, http://www.smarttech.com/.

[4] C. Kirstein and H. Mueller, "Interaction with a projection screen using a camera-tracked laser pointer," in *Proceedings of the International Conference on Multimedia Modeling (MMM 98)*, (Lausanne, Switzerland), pp. 191–192, Oct. 1998.

[5] R. Sukthankar, R. Stockton, and M. Mullin, "Automatic keystone correction for camera-assisted presentation interfaces," in *Proceedings of the Third International Conference on Advances in Multimodal Interfaces (ICMI 2000)*, (Beijing, China), Oct. 2000.

[6] E.A. Bier and S. Freeman, "MMM: A User Interface Architecture for Shared Editors on a Single Screen," in *Proceedings of the ACM Symposium on User Interface Software and Technology*, pp. 79–86, 1991.

[7] J. Rekimoto, "A Multiple Device Approach for Supporting Whiteboard-Based Interactions," in *Proceedings of CHI 98: Conference on Human Factors in Computing Systems*, (Los Angeles, CA), pp. 344–351, Apr. 1998.

[8] B. A. Myers, R. C. Miller, B. Bostwick, and C. Evankovich, "Extending the Windows Desktop Interface with Connected Handheld Computers," in *Proceedings of the 4th USENIX Windows Systems Symposium*, (Seattle, WA), pp. 79–88, Aug. 2000.

25th Digital Avionics Systems Conference
October 15, 2006