ABSTRACT

We report on the design and implementation of tools to enable distributed computer supported cooperative work (CSCW) within distributed virtual environments (DVEs). The current implementation allows the user to interact with remote participants using public bulletins, private e-mail, real-time chat sessions, shared viewpoints, recorded video, and annotations (notes) posted within the DVE. We transmit this data between application computers using experimental Distributed Interactive Simulation (DIS) protocol data units and non-DIS data streams. Achieving these capabilities required developing means for display of information sent by other users, a means for dispatching information to users, and development of experimental DIS protocol data units.

1. INTRODUCTION

When commanders and their staffs operate in the real world, they collaborate. As they work together on a problem, they share and communicate important information. Because of the pace of operations in the modern battlespace, the rate of information flow is rapid. The pace of information flow within military operations creates an environment that can overwhelm the individual with information. In this type of environment, as in others where information overload impedes mission accomplishment, realistic training modeled on real world circumstances can be effective in preparing individuals and groups to deal with analogous real world situations. However, within current distributed virtual environment (DVE) applications each participant operates in isolation and lacks facilities for sharing information with other participants without leaving the DVE application and using supplementary means. Therefore, because the facilities for providing information do not exist and the resultant interactions are incorrect, the participants are encumbered with unnecessary work and they do not operate within a realistic training environment.

Our goal is to assist users in maintaining their focus on the battlespace environment by equipping DVE applications with collaborative tools. This paper reports on our progress in developing collaborative tools for use within DVEs.

The Synthetic BattleBridge (SBB) project is the application at the core of our work on developing tools for analysis and collaboration [1, 2, 3, 4]. The SBB is a DVE [5] application that allows its user to monitor, assess, and analyze the activities within a virtual battlespace. The SBB’s immersive user-interface, the Pod, provides controls for the user’s view of the world, supports display of information about the entities within the virtual environment, and provides a framework to extend application functionality. Previous work on the SBB has inserted capabilities for movement to any location in a virtual battlespace and observation of activity without interfering with the activity occurring in the virtual battlespace. Additionally, the SBB provides, as a commander’s decision-aid, the capability for remotely monitoring activity at specific locations within the DVE using drop cameras, video missiles, trails, locators, and Sentinels [1]. These capabilities help the individual SBB user to locate important activity within a large, complicated battlespace. However, these tools do not help the individual to work cooperatively and interact with other DVE participants. We extended the functionality of the SBB by integrating tools for Computer Supported Cooperative Work (CSCW) into its architecture. The tools we implemented were selected based upon their ability to allow the individual to communicate using techniques that are analogous to their current modes of work or that support anticipated future modes of work.

The current implementation allows the user to interact with participants in other SBBs using public bulletins, private e-mail, real-time chat sessions, shared viewpoints, live or recorded video, and shared annotations (notes) posted within the virtual environment. Communication between application host computers uses experimental Distributed Interactive Simulation (DIS) compliant protocol data units as defined in IEEE 1278-1993 and non-DIS data streams.
The SBB’s collaborative workspace gives the user a wide range of communication options within a DVE application. Below, we describe each of the tools for CSCW that we developed, their purpose and capabilities, their design, the techniques we use for display of information, and the user interface to the tools. Finally, based upon our experience in the design and implementation of these tools, we will discuss requirements for additional collaborative tools for use within DVEs.

The next section presents a brief review of background information that supports our work. Section Three presents the requirements and design for the CSCW enhanced SBB. Section Four summarizes our implementation of the CSCW capabilities. Section Five contains our results and suggestions for future work.

2. BACKGROUND

A distributed virtual environment (DVE) is a large-scale, networked, computer-based, virtual world wherein several thousand human and computer controlled entities can interact [5]. Entities move within the virtual environment using aerodynamics, celestial mechanics, vehicle dynamics, and soil mechanics models to compute their motion. The environment requires models of the terrain and other features such as roads, forests, rivers, bridges and buildings.

Table 1: Cooperative Systems Classification Scheme

<table>
<thead>
<tr>
<th></th>
<th>Same Time</th>
<th>Different Times</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Same Place</strong></td>
<td>face to face</td>
<td>asynchronous interaction (project scheduling, work product coordination)</td>
</tr>
<tr>
<td><strong>Different Places</strong></td>
<td>synchronous distributed communication (shared editors, video)</td>
<td>asynchronous distributed communication (e-mail, videoconferences, web, voice mail)</td>
</tr>
</tbody>
</table>

To enable cooperation, the user must be provided with facilities that support the location and mode of work to be performed. Table 1 [6] presents a classification system for cooperative systems. The table highlights the need for tools that operate asynchronously and synchronously as well as for tools that permit users to work together from separate locations. The design and implementation of collaborative and cooperative work capabilities for a person within a DVE must consider the types of work to be accomplished and the manner in which work is performed within the real world. The first clear design consideration is that users must be able to cooperate synchronously and asynchronously because within an organization information is communicated both ways. For example, synchronous communication occurs when one organization member communicates with another using phone or videoteleconference capabilities or a supervisor is briefed in person by a subordinate. Asynchronous communication occurs when papers, notes, or e-mail are exchanged. A further consideration is that network bandwidth constrains the types of communication currently practical within a DVE. As a result of bandwidth limitations and the need for both asynchronous and synchronous communication, we decided to concentrate our efforts on the development of capabilities for synchronous distributed and non-video asynchronous distributed communication.

Previous work in distributed CSCW has focused on the need to present users with a virtual office-like environment wherein work can be performed. An example of this approach is the DIVE system [7]. DIVE was undertaken to develop multi-actor distributed virtual environments for distributed cooperative work. In these environments, information about each entity is maintained within a homogeneous, shared, distributed database. Communication between participants is controlled by movement within a shared virtual space, which allows participants to enter and leave conversations, or worlds, by moving within the virtual space. The information describing a world is common the participants in the world. Only a host that has joined the multicast group for a specific world is aware of the activity in the world. DIVE supports text, audio, and video information exchanges and allows a wide variety of host systems to interact by supporting information interchange between hosts with a variety of interface capabilities, such as only text, text and voice, and text and video capabilities. Because of the challenges involved, we limited our efforts to supporting systems with equivalent interface capabilities.

The SBB’s architecture is based upon the Common Object DataBase (CODB) [8]. The Common Object DataBase is primarily a data-handling architecture that uses structured classes, data containers, and a central runtime data repository to transmit data between application objects. This architecture reduces the coupling in a simulation by reducing the amount of information that a class must maintain about other classes. As a result, the simulation engineer can abstract away from the classes and methods that produce data. To acquire the data the simulation engineer must only access the container in the CODB where the information resides. Figure 1 depicts the CODB architecture used within the Synthetic BattleBridge.
In the SBB, the CODB provides an inheritable, reusable interface to the computer hardware, input and output devices, and Silicon Graphics' Performer software. When the SBB begins execution, the CODB loads the terrain description for the distributed virtual environment and the description of the static portions of the distributed virtual environment. The CODB also loads the 3D models for all the actors in the DVE. Finally, the CODB provides an interface to reusable and inheritable classes for a variety of input devices. The CODB allows its serviced processes to access data within it at any time, unless the data is locked by a semaphore.

The SBB’s DIS interface is our Lab’s World State Manager 3.0 (WSM). The WSM provides both a protocol data unit send and receive capability. The World State Manager provides containers for broadcasting aircraft and weapons entity state PDUs and for receiving entity state PDUs.

The SBB’s user interface is based upon the Information Pod, or Pod, metaphor. The Pod is a user “capsule” within the virtual world that supports immersed user operation of a system within a virtual environment. The Pod is composed of five large control panels which are available to the user in each of the forward-facing cardinal directions: center, left, right, top, bottom. The Pod replaces a desktop metaphor for the interface with one of a control console or the bridge of a ship, which is readily understood. The Pod virtually encases the user with controls and with displays mounted upon 2D panels in 3D space, with each panel devoted to a specific set of tasks. Tasks are assigned to panels and subpanels by factoring the tasks to be performed by the affinity of the tasks. The controls and displays change contents depending upon the user’s focus of attention and the task to be accomplished. The Pod allows us to provide controls that support precise user movement, observation, decision making, and environment contents.
The next design task was to determine where the CSCW tool controls should be placed and to make the best use of the Pod’s five panels without creating additional panels. The top and bottom panels were filled with radar information that could not be removed. The left panel contained detailed DIS entity information and a control for attaching the user to an entity or a predefined camera position. This meant that the center panel (used for navigation) and the right panel (primarily used for displaying Heads-Up Displays, or HUDs) would be the most likely choices for adding collaborative workspace features. We decided to place the CSCW controls on the center panel for the sake of user convenience and because this placement seemed more appropriate than a placement on the right panel.

The center panel contains six subpanels, five of which are used for navigational controls. However, the Pod’s design allows a panel to be temporarily hidden from the user and replaced with an entirely different panel. This functionality provides increased console space by allowing the user to flip through various control panels until the desired controls are displayed. Therefore, we chose to use a single button on the center navigational panel that clears the console and provides access to three full-sized CSCW panels “hidden underneath.” These three panels control the CSCW capabilities for email, bulletins, chat, video, shared viewpoint, and annotations. We discuss the requirements for these five components next.

Private email is simply a text message directed from a sender to a specific recipient. The recipient must be able to display the message and identity of the sender so that a reply can be formed if necessary. Three other features that we require are message deletion, saving messages to a file, and message reply. Public bulletins are broadcasts sent to all users on the network. Public bulletins have the same requirements of a private email system with the exception that the targeted audience includes all members on the network. Due to their commonalities, public bulletins and private emails are accessed through the same user interface. Chat is a restricted form of public bulletins and can be built upon the facilities that public bulletins require.

The SBB user should be able to capture a sequence of images that occur within the HUD and then share these files with other users who can display the sequence as an animation. The SBB’s animation controls must be integrated together as a single unit within the HUD interface module to facilitate quick capture and transfer of video data.

Remote viewing using a heads-up display (HUD) had already been implemented in the SBB [4]. Remote cameras, video missiles, and radar displays proved to be valuable aids in situational awareness. We required a shared viewpoint capability so that a user can quickly “step into the shoes” of another SBB user and quickly return back to the original location. The shared viewpoint design mimics the previous implementation of the HUD, remote cameras, video missiles, and radar displays with two important additions: the ability to teleport to and from the shared viewpoint’s position (with the correct orientation) and the ability to capture, transmit, and playback a sequence of HUD images. Shared viewpoint teleportation transports the SBB’s Pod to the exact position and orientation of the remote SBB.

The goal for the annotations facility was to create a three-dimensional whiteboard attached to a particular location in the virtual world. However, technical limitations did not permit us to pursue this solution. Instead, we developed an annotation capability that allows a SBB user to attach notes to objects within the DVE that other SBB users can then read at their own convenience. The final annotation interface design used the email and chat interfaces and combined them to control annotations. As a result, annotations could be displayed on the Pod’s console either by cycling through the annotations using the standard forward/back button controls or by clicking on the physical model of the annotation and requesting that its contents be displayed. To aid in locating annotations at a distance, screen-space labels were placed above them. Annotations can also be attached to a DIS entity.

Taking the discussed requirements and individual tool design considerations into account, we developed a CSCW design for the SBB that utilizes the CODB architecture, as shown in Figure 2. The design consists of a Common Object DataBase at the center of the application, this is the primary mechanism for sharing data between simulation classes. The SBB_Renderer is responsible for setting up the Performer environment, managing the top-level Performer tree, managing entity models (including locators, trails, and range cones), and manipulation of the current view. The SBB’s World State Manager is responsible for communicating with the other entities in DIS-based DVE. The Pod_Player class, which takes up the right side of the figure, is responsible for maintaining all of the Pod’s panels and their respective functions. Each panel is shown with an aggregated listing of its particular components.
4. IMPLEMENTATION

Our implementation of CSCW capabilities required development of tools for the following capabilities: identifying other SBBs, Messages, Chat, Annotations, Shared Viewpoints, and animation controls. As a first step toward implementing SBB communication with other SBBs, we drafted a Stealth Entity-State PDU format. This new PDU format contains position and orientation information about a player that can be used to correctly display its representation within the virtual environment. This information can also be used to view the DVE from another SBB’s perspective, allowing us to achieve the shared viewpoint capability.

Our next task was to devise a unique identity for each SBB so that its activities can be followed and its state maintained by other SBBs. Therefore, we created a composite ID code for each SBB consisting of site, application, and entity IDs. SBB users are identified by name by asking the user to input a name upon SBB initialization. This name is used to identify all transmissions from the user including positional information, messages, and annotations. To ensure that each SBB’s state is properly maintained by the other SBBs, each SBB broadcasts stealth entity state PDUs with 12 seconds of simulation start. Thereafter, as the SBB moves within the DVE, the SBB’s coordinates are broadcast to all of the other stealth players.

After enabling a SBB’s state to be reliably maintained by other SBBs, the most basic form of direct communication was implemented, the text message. While it was relatively simple to send and receive messages using an IRIX console window, the challenge was to manage sending and receiving messages using the Pod. Two issues must be resolved for workspace messages, organization of the message structure and display of the messages. The nature of messages, their time sequential ordering, the potentially unlimited number of messages, and the need to delete any message at will, led us to structure the messages within a doubly-linked list. The displaying of messages was a more difficult problem than anticipated. The challenge was to find a way to display long 3D text strings in the smallest possible area. This was solved by creating a pseudo-panel that appeared between the center panel and the top panel. The display panel is completely opaque with a black background and yellow text for maximum contrast. The type of message, the message number, and the sender’s name form the message header, the message body is displayed below the header. We developed a word-wrap function so that each message can be fully specified in three dimensions before being drawn. Messages are broadcast one time only and are not stored outside of the SBB; therefore, if a message is missed a SBB has no means of acquiring it.

Using the Message control panel as a guide, Chat development was a straightforward task. The Chat control panel includes the ability to send preloaded chat messages, display the current chat session, kill the current chat log, or save the chat log to a file. These functions mirror those found in the Message control panel. At the first line of a chat message, the sender’s name, cropped to three characters, is displayed as a prefix to the text string. Like Messages, Chat sessions are not saved.

The Annotations implementation is a hybrid between Workspace Messages and DIS entities because they contain text and have state within the DVE. The SBB allows Annotations to be broadcast using the annotation control panel. The annotation control panel allows the user to dispatch preloaded messages in a similar manner to Message and Chat. The user chooses the appropriate message during execution and sends it rather than composing a message in real time. The annotation is broadcast to all SBBs in the DVE. Each annotation is represented in the virtual world with a pennant icon. Each pennant is color-coded to a single SBB player, so that an annotation’s owner can be immediately determined. Pennant colors are also used to indicate newly created or recently changed annotations, selected annotations, and old annotations. Viewing annotation messages involves selecting the annotation and then displaying its contents. The display of annotation messages is identical in nature to the Chat. Due to the limitations of screen space, only the most recent messages are shown on the text display screen. The remaining annotation management controls include deleting an annotation, clearing an annotation, appending a message to an annotation, moving an annotation, and attaching an annotation to an entity. Since Annotations are permanent fixtures of the DVE for every SBB, the status of each annotation must be retained by its host SBB and broadcast whenever an annotation is updated. To conserve bandwidth, Annotations do not emit a heartbeat to keep them in the simulation. Instead, annotation information is received via annotation updates. Upon receiving an
annotation update, each SBB is able to retrieve a complete record of the annotation’s state.

Workspace shared viewpoints were implemented in the same fashion as the Video Missile and Drop Camera HUDs. Using a style similar to the Drop Camera HUD controls, the SBB user is able to cycle through the viewpoints of stealth players based upon their current position and orientation.

The final component of the SBB’s CSCW capability is the video controls. The video controls consist of three buttons: Record, Playback, and Send. When the Record button is depressed, the top right-hand corner of the screen (where HUD images are displayed) is captured every other frame. This maintains acceptable SBB performance (i.e., 10 frames per second) while capturing enough images to produce a smooth animation of over two minutes of captured video. Playback of this video occurs through the use of a movie player. The animation is displayed at the same resolution of the captured images.

5. RESULTS AND FUTURE WORK

The PDU formats that we developed satisfied all of our needs for communication. The Stealth Entity State and Comment PDUs were used to broadcast stealth entity state changes and messages between SBBs. The Comment PDU is used for all text message traffic, including private email, public bulletins, and chat sessions. The Annotation Update PDU was created as the starting point for a new family of PDUs since none of the existing family of PDUs were appropriate for annotations created by stealth applications.

Within the DVE, every SBB can now operate like normal entities, broadcasting movement changes that exceed dead-reckoning limits and “heartbeats” to keep the player “alive” in the exercise.

The Workspace Message panel, shown in Figure 3, allows private and public messages to be displayed, saved to a file, deleted, or sent via the touch of a button. The Pod interface did limit the user to sending only preloaded messages, this shortcoming should be remedied in future SBB development.

Figure 3: Workspace Message Status Panel on the Center Panel Within the SBB.

Figure 4 shows how a message is displayed above the front panel within the SBB.

Figure 4: Message Display Panel Within the SBB. Annotations, shown in Figure 5, were successfully implemented in the SBB’s CSCW capabilities, and are the basis for a new class of DIS objects. While the current annotation implementation contains only text messages and performs basic annotation operations, the infrastructure is expandable to a wide variety of media types and actions.

Figure 5: A SBB’s Annotation Pennant Within the Distributed Virtual Environment.

Recommendations for continued research in the field of CSCW for virtual environments include implementation of a virtual keyboard, support for real-time audio communication, creation of an ATM client/server mechanism for transmission of audio and video messages, and improvement of manipulation techniques for annotations. Additionally, the implementation of the White Board PDU and creation of the Annotation Data PDU are necessary. The White Board PDU should replace the Comment PDU for basic message transfers and be integrated into the Annotations Update PDU. The Annotation family of PDUs should be expanded to permit late arrivals to the DIS environment to be informed of all annotations within the battlespace.

A few items of additional research and implementation are required to improve the CSCW capabilities in the SBB. Audio is clearly needed. One tool that would assist users in cooperative work would be to implement a whiteboard with drawing capability. A virtual meeting capability, as in DIVE, would also enhance the capabilities for CSCW in the
SBB. As part of this capability, facilities for identifying meeting topics and participants would be needed. Support for cooperation between heterogeneous collections of host systems is also needed. The current system does not support archiving of audio, video, chat or messages; this capability should be implemented so that latecomers to the DVE can review previous activity and so that it can be used for after action analysis.

REFERENCES


