QuickSet: Multimodal Interaction for Distributed Applications

Philip R. Cohen, Michael Johnston, David McGee, Sharon Oviatt,
Jay Pittman, Ira Smith, Liang Chen and Josh Clow
Center for Human Computer Communication
Oregon Graduate Institute of Science and Technology
P.O.Box 91000
Portland, OR 97291-1000 USA
Tel: 1-503-690-1326
E-mail: pcohen@cse.ogi.edu
http://www.cse.ogi.edu/CHCC

ABSTRACT

This paper presents an emerging application of multimodal interface research to distributed applications. We have developed the QuickSet prototype, a pen/voice system running on a hand-held PC, communicating via wireless LAN through an agent architecture to a number of systems, including NRaD's[1] LeatherNet system, a distributed interactive training simulator built for the US Marine Corps. The paper describes the overall system architecture, a novel multimodal integration strategy offering mutual compensation among modalities, and provides examples of multimodal simulation setup. Finally, we discuss our applications experience and evaluation.

KEYWORDS: multimodal interfaces, agent architecture, gesture recognition, speech recognition, natural language processing, distributed interactive simulation.

1. INTRODUCTION

A new generation of multimodal systems is emerging in which the user will be able to employ natural communication modalities, including voice, hand and pen-based gesture, eye-tracking, body-movement, etc. [Koons et al., 1993; Oviatt, 1992, 1996; Waibel et al., 1995] in addition to the usual graphical user interface technologies. In order to make progress on building such systems, a principled method of modality integration, and a general architecture to support it is needed. Such a framework should provide sufficient flexibility to enable rapid experimentation with different modality integration architectures and applications. This experimentation will allow researchers to discover how each communication modality can best contribute its strengths yet compensate for the weaknesses of the others.

Fortunately, a new generation of distributed system frameworks is now becoming standardized, including the CORBA and DCOM frameworks for distributed object systems. At a higher level, multiagent architectures are being developed that allow integration and interoperation of semi-autonomous knowledge-based components or "agents". The advantages of these architectural frameworks are modularity, distribution, and asynchrony — a subsystem can request that a certain functionality be provided without knowing who will provide it, where it resides, how to invoke it, or how long to wait for it. In virtue of these qualities, these frameworks provide a convenient platform for experimenting with new architectures and applications.

In this paper, we describe QuickSet, a collaborative, multimodal system that employs such a distributed, multiagent architecture to integrate not only the various user interface components, but also a collection of distributed applications. QuickSet provides a new *unification-based* mechanism for fusing partial meaning representation fragments derived from the input modalities. In so doing, it selects the best *joint* interpretation among the alternatives presented by the underlying spoken language and gestural modalities. Unification also supports multimodal discourse. The system is scaleable from handheld to wall-sized interfaces, and interoperates across a number of platforms (PC's to UNIX workstations). Finally, QuickSet has been applied to a collaborative military training system, in which it is used to control a simulator and a 3-D virtual terrain visualization system.

This paper describes the "look and feel" of the multimodal interaction with a variety of back-end applications, and discusses the unification-based architecture that makes this new class of interface possible. Finally, the paper discusses the application of the technology for the Department of Defense.

[1] NRaD = US Navy Command and Control Ocean Systems Center Research Development Test and Evaluation (San Diego).

2. QUICKSET

QuickSet is a collaborative, handheld, multimodal system for interacting with distributed applications. In virtue of its modular, agent-based design, QuickSet has been applied to a number of applications in a relatively short period of time, including:

- *Simulation Set-up and Control* — Quickset is used to control LeatherNet [Clarkson and Yi, 1996], a system employed in training platoon leaders and company commanders at the USMC base at Twentynine Palms, California. LeatherNet simulations are created using the ModSAF simulator [Courtmanche and Ceranowicz, 1995] and can be visualized in a wall-sized virtual reality CAVE environment [Cruz-Neira et al., 1993; Zyda et al., 1992] called CommandVu. A QuickSet user can create entities, give them missions, and control the virtual reality environment from the handheld PC. QuickSet communicates over a wireless LAN via the Open Agent Architecture (OAA) [Cohen et al., 1994] to ModSAF, and to CommandVu, each of which have been made into agents in the architecture.
- *Force Laydown* — QuickSet is being used in a second effort called ExInit (Exercise Initialization), that enables users to create large-scale (division- and brigade- sized) exercises. Here, QuickSet interoperates via the agent architecture with a collection of CORBA servers.
- *Medical Informatics* — A version of QuickSet is used in selecting healthcare in Portland, Oregon. In this application, QuickSet retrieves data from a database of 2000 records about doctors, specialties, and clinics.

Next, we turn to the primary application of QuickSet technology.

3. NEW INTERFACES FOR DISTRIBUTED SIMULATION

Begun as SIMNET in the 1980's [Thorpe, 1987], distributed, interactive simulation (DIS) training environments attempt to provide a high degree of fidelity in simulating combat equipment, movement, atmospheric effects, etc. One of the U.S. Government's goals, which has partially motivated the present research, is to develop technologies that can aid in substantially reducing the time and effort needed to create large-scale scenarios. A recently achieved milestone is the ability to create and simulate a large-scale exercise, in which there may be on the order of 60,000 entities (e.g., a vehicle or a person).

QuickSet addresses two phases of user interaction with these simulations: creating and positioning the entities, and supplying their initial behavior. In the first phase, a user "lays down" or places forces on the terrain, which need to be positioned in realistic ways, given the terrain, mission, available equipment, etc. In addition to force laydown the user needs to supply them with behavior, which may involve complex maneuvering, communication, etc.

Our contribution to this overall effort is to rethink the nature of the user interaction. As with most modern simulators, DISs are controlled via graphical user interfaces (GUIs). However, GUI-based interaction is rapidly losing its benefits, especially when large numbers of entities need to be created and controlled, often resulting in enormous menu trees. At the same time, for reasons of mobility and affordability, there is a strong user desire to be able to create simulations on small devices (e.g., PDA's). This impending collision of trends for smaller screen size and for more entities requires a different paradigm for human-computer interaction with simulators.

A major design goal for QuickSet is to provide the same user input capabilities for handheld, desktop, and wall-sized terminal hardware. We believe that only voice and gesture-based interaction comfortably span this range. QuickSet provides *both* of these modalities because it has been demonstrated that there exist substantive language, task performance, and user preference advantages for multimodal interaction over speech-only and gesture-only interaction with map-based tasks [Oviatt, 1996; Oviatt, in press].[2] Specifically, for these tasks, multimodal input results in 36% fewer task performance errors, 35% fewer spoken disfluencies, 10% faster task performance, and 23% fewer words, as compared to a speech-only interaction. Multimodal pen/voice interaction is known to be advantageous for small devices, for mobile users who may encounter different circumstances, for error avoidance and correction, and for robustness [Oviatt, 1992; Oviatt 1995].

In summary, a multimodal voice/gesture interface complements, but also promises to address the limitations of, current GUI technologies for controlling simulators. In addition, it has been shown to have numerous advantages over voice-only interaction for map-based tasks. These findings had a direct bearing on the interface design and architecture of QuickSet.

4. SYSTEM ARCHITECTURE

In order to build QuickSet, distributed agent technologies based on the Open Agent Architecture[3] were employed because of its flexible asynchronous capabilities, its ability to run the same set of software components in a variety of hardware configurations, ranging from standalone on the handheld PC to distributed operation across numerous computers, and its easy connection to legacy applications. Additionally, the architecture supports user mobility in that less computationally-intensive agents (e.g., the map interface) can run on the handheld PC, while more computationally-intensive processes (e.g., natural language processing) can operate elsewhere on the network. The agents may be written in any programming language (here, Quintus Prolog, Visual C++, Visual Basic, and Java), as long as they communicate via an interagent communication language. The configuration of agents used in the QuickSet system is illustrated in Figure 1. A brief description of each agent follows.

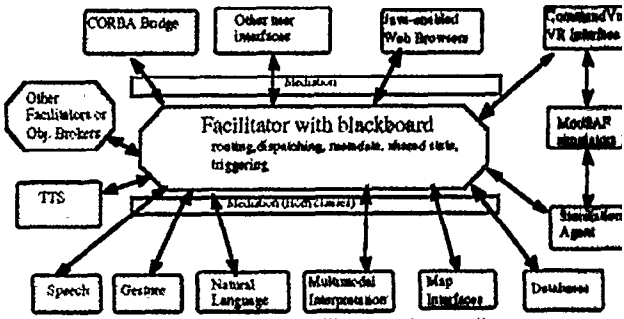

Figure 1: The facilitator, channeling queries to capable agents.

[2] Our prior research [Cohen et al., 1989; Cohen, 1992] has demonstrated the advantages of a multimodal interface offering natural language and direct manipulation for controlling simulators and reviewing their results.

[3] Open Agent Architecture is a trademark of SRI International.

5. EXAMPLES

5.1 Leathernet

Holding QuickSet, the user views a map from the ModSAF simulation. With speech and pen, she then adds entities into the ModSAF simulation. For example, to create a unit in QuickSet, the user would hold the pen at the desired location and utter: "red T72 platoon" resulting in a new platoon of the specified type being created. The user then adds a barbed-wire fence to the simulation by drawing a line at the desired location while uttering "barbed wire." A fortified line can be added multimodally, by drawing a simple line and speaking its label, or unimodally, by drawing its military symbology. A minefield of an amorphous shape is drawn and is labeled verbally. Finally an M1A1 platoon is created as above. Then the user can assign a task to the platoon by saying "M1A1 platoon follow this route" while drawing the route with the pen.

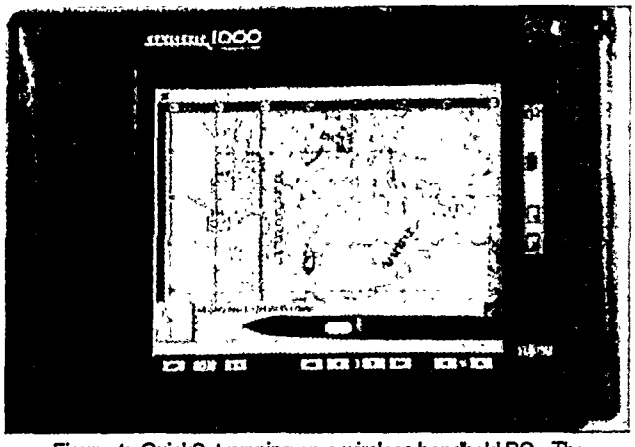

Figure 4: QuickSet running on a wireless handheld PC. The user has created numerous units, fortifications and objectives.

The results of these commands are visible on the QuickSet screen, as seen in Figure 4, as well as on the ModSAF simulation, which has been executing the user's QuickSet commands in the virtual world (Figure 5).

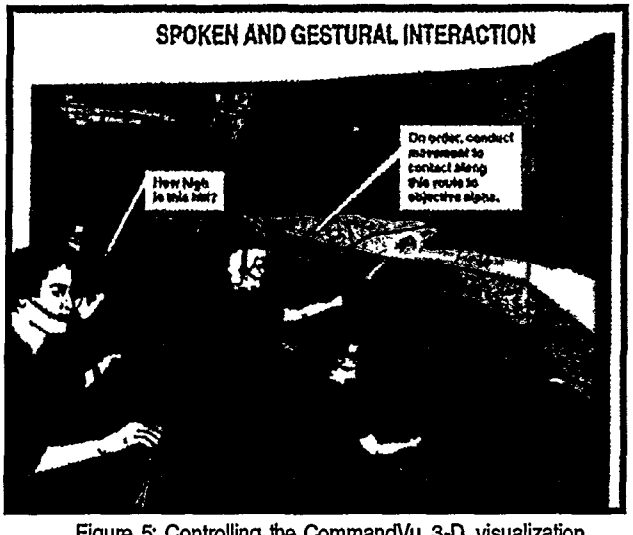

Figure 5: Controlling the CommandVu 3-D visualization via QuickSet interaction. QuickSet tablets are on the desks.

Two specific aspects of QuickSet to be discussed below are its usage as a collaborative system, and its ability to control a virtual reality environment.

5.1.1 Collaboration.

In virtue of the facilitated agent architecture, when two or more user interfaces connected to the same network of facilitators subscribe to and/or produce common messages, they (and their users) become part of a collaboration. The agent architecture offers a framework for heterogeneous collaboration, in that users can have very different interfaces, operating on different types of hardware platforms, and yet be part of a collaboration. For instance, by subscribing to the entity-location database messages, multiple QuickSet user interfaces can be notified of changes in the locations of entities, and can then render them in whatever form is suitable, including 2-D map-based, web-based, and 3-D virtual reality displays. Likewise, users can interact with different interfaces (e.g., placing entities on the 2-D map or 3-D VR) and thereby affect the views seen by other users. To allow for tighter synchronicity, the current implementation also allows users to decide to couple their interface to those of the other users connected to a given network of facilitators. Then, when one interface pans and zooms, the other coupled ones do as well. Furthermore, coupled interfaces subscribe to the "ink" messages, meaning one user's ink appears on the others' screens, immediately providing a shared drawing system. On the other hand, collaborative systems also require facilities to prevent users from interfering with one another. QuickSet incorporates authentication of messages in order that one user's speech is not accidentally integrated with another's gesture.

In the future, we will provide a subgrouping mechanism for users, such that there can be multiple collaborating groups using the same facilitator, thereby allowing users to be able to choose to join collaborations of specific subgroups. Also to be developed is a method for handling conflicting actions during a collaboration.

5.1.2 Multimodal Control of Virtual Travel

Most terrain visualization systems allow only for flight control, either through a joystick (or equivalent), via keyboard commands, or via mouse movement. Unfortunately, to make effective use of such interfaces, people need to be pilots, or at least know where they are going. Believing this to be unnecessarily restrictive, our virtual reality set-up follows the approach recommended by Baker and Wickens [unpublished ms]., Brooks [1996], and Stoakley et al., [1995] in offering two "linked" displays — a 2-D "birds-eye" map-based display (QuickSet), and the 3-D CommandVu visualization. In addition to the existing 3-D controls, the user can issue spoken or multimodal commands via the handheld PC to be executed by CommandVu. Sample commands are:

"CommandVu, heads up display on,"

"take me to objective alpha"

"fly me to this platoon <gesture on QuickSet map>" (see Figure 4).

"fly me along this route <draws route on QuickSet map> at fifty meters"

Spoken interaction with virtual worlds offers distinct advantages over direct manipulation, in that users are able to describe entities and locations that are not in view, can be teleported to those out-of-view locations and entities, and can

QuickSet interface: On the handheld PC is a geo-referenced map of some region,[4] such that entities displayed on the map are registered to their positions on the actual terrain, and thereby to their positions on each of the various user interfaces connected to the simulation. The map interface provides the usual pan and zoom capabilities, multiple overlays, icons, etc. Two levels of map are shown at once, with a small rectangle shown on a miniature version of the larger scale map indicating the portion of it shown on the main map interface.

Employing pen, speech, or more frequently, multimodal input, the user can annotate the map, creating points, lines, and areas of various types. The user can also create entities, give them behavior, and watch the simulation unfold from the handheld. When the pen is placed on the screen, the speech recognizer is activated, thereby allowing users to speak and gesture simultaneously. The interface offers controls for various parameters of speech recognition, for loading different maps, for entering into collaborations with other users, for connecting to different facilitators, and for discovering other agents who are connected to the facilitator. The QuickSet system also offers a novel map-labeling algorithm that attempts to minimize the overlap of map labels as the user creates more complex scenarios, and as the entities move (cf. [Christensen et al., 1996]).

Speech recognition agent: The speech recognition agent used in QuickSet is built on IBM's VoiceType Application Factory and VoiceType 3.0, recognizers, as well as Microsoft Whisper speech recognizer.

Gesture recognition agent: QuickSet's pen-based gesture recognizer consists of both a neural network [Pittman, 1991, Manke et al., 1994] and a set of hidden Markov models. The digital ink is size-normalized, centered in a 2D image, and fed into the neural network as pixels. The ink is also smoothed, resampled, converted to deltas, and given as input to the HMM recognizer. The system currently recognizes 68 pen-gestures, including various military map symbols (platoon, mortar, fortified line, etc.), editing gestures (deletion, grouping), route indications, area indications, taps, etc. The probability estimates from the two recognizers are combined to yield probabilities for each of the possible interpretations. The inclusion of route and area indications creates a special problem for the recognizers, since route and area indications may have a variety of shapes. This problem is further compounded by the fact that the recognizer needs to be robust in the face of sloppy writing. More typically, sloppy forms of various map symbols, such as those illustrated in Figure 3, will often take the same shape as some route and area indications. A solution for this problem can be found by combining the outputs from the gesture recognizer with the outputs from the speech recognizer, as is described in the following section.

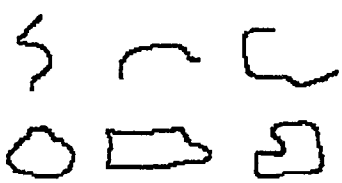

Figure 2: Pen drawings of routes and areas. Routes and areas do not have signature shapes that can be used to identify them.

[4] QuickSet can employ either UTM or Latitude/Longitude coordinate systems.

Figure 3: Typical pen input from real users. The recognizer must be robust in the face of sloppy input.

Natural language agent: The natural language agent currently employs a definite clause grammar and produces typed feature structures as a representation of the utterance meaning. Currently, for the force laydown and mission assignment tasks, the language consists of noun phrases that label entities, as well as a variety of imperative constructs for supplying behavior.

Text-to-Speech agent: Microsoft's text-to-speech system has been incorporated as an agent, residing on each individual PC.

Multimodal integration agent: The task of the integrator agent is to field incoming typed feature structures representing individual interpretations of speech and of gesture, and identify the best potential unified interpretation, multimodal or unimodal. In order for speech and gesture to be incorporated into a multimodal interpretation, they need to be both semantically and temporally compatible. The output of this agent is a typed feature structure representing the preferred interpretation, which is ultimately routed to the bridge agent for execution. A more detailed description of multimodal interpretation is in Section 6.

Simulation agent: The simulation agent, developed primarily by SRI International [Moore et al., 1997], but modified by us for multimodal interaction, serves as the communication channel between the OAA-brokered agents and the ModSAF simulation system. This agent offers an API for ModSAF that other agents can use.

Web display agent: The Web display agent can be used to create entities, points, lines, and areas, and posts queries for updates to the state of the simulation via Java code that interacts with the blackboard and facilitator. The queries are routed to the running ModSAF simulation, and the available entities can be viewed over a WWW connection.

CommandVu agent: Since the CommandVu virtual reality system is an agent, the same multimodal interface on the handheld PC can be used to create entities and to fly the user through the 3-D terrain.

Application bridge agent: The bridge agent generalizes the underlying applications' API to typed feature structures, thereby providing an interface to the various applications such as ModSAF, CommandVu, and Exinit. This allows for a domain-independent integration architecture in which constraints on multimodal interpretation are stated in terms of higher-level constructs such as typed feature structures, greatly facilitating reuse.

CORBA bridge agent: This agent converts OAA messages to CORBA IDL (Interface Definition Language) for the Exercise Initialization project.

To see how QuickSet is used, we present the following examples.

ask questions about entities in the scene. We are currently engaged in research to allow the user to gesture directly into the 3-D scene while speaking, a capability that will make these more sophisticated interactions possible.

5.2 Exercise Initialization: ExInit

QuickSet has been incorporated into the DoD's new Exercise Initialization tool, whose job is to create the force laydown and initial mission assignments for very large-scale simulated scenarios. Whereas previous manual methods for initializing scenarios resulted in a large number of people spending more than a year in order to create a division-sized scenario, a 60,000+ entity scenario recently took a single ExInit user 63 hours, most of which was computation.

ExInit is distinctive in its use of CORBA technologies as the interoperation framework, and its use of inexpensive off-the-shelf personal computers. ExInit's CORBA servers (written or integrated by MRJ Corp. and Ascent Technologies) include a relational database (Microsoft Access or Oracle), a geographical information system (CARIS), a "deployment" server that knows how to decompose a high-level unit into smaller ones and position them in realistic ways with respect to the terrain, a graphical user interface, and QuickSet for voice/gesture interaction.

In order for the QuickSet interface to work as part of the larger ExInit system, a CORBA bridge agent was written for the OAA, which communicated via IDL to the CORBA side, and via the interagent communication language to the OAA agents. Thus, to the CORBA servers, QuickSet is viewed as a Voice/Gesture server, whereas to the QuickSet agents, ExInit is simply another application agent. Users can interact with the QuickSet map interface (which offers a fluid multimodal interface), and view ExInit as a "back-end" application similar to ModSAF. A diagram of the QuickSet-ExInit architecture can be found in Figure 6. Shown there as well is a connection to DARPA's Advanced Logistics Program demonstration system for which QuickSet is the user interface.

To illustrate the use of QuickSet for ExInit, consider the example of Figure 7, in which, a user has said: "Multiple boundaries," followed in rapid succession by a series of multimodal utterances such as "Battalion <draws line>," "Company <draws line>," etc. The first utterance tells ExInit that subsequent input is to be interpreted as a boundary line, if possible. When the user then names an echelon and draws a line, the multimodal input is interpreted as a boundary of the appropriate echelon.

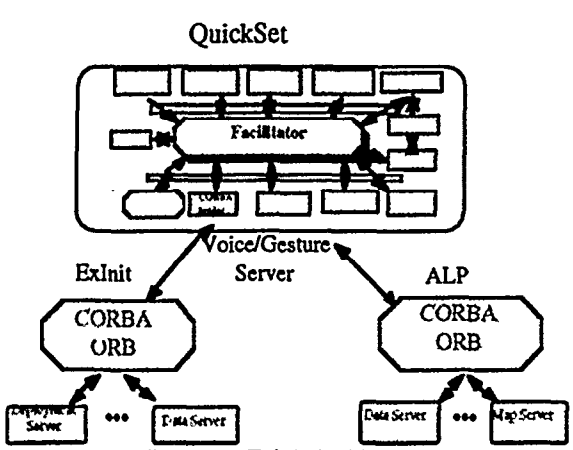

Figure 6: ExInit Architecture

Numerous features describing engineering works, such as a fortified line, a berm, minefields, etc. have also been added to the map using speech and gesture. Then the user creates a number of armored companies facing 45 degrees in defensive posture; he is now beginning to add armored companies facing 225 degrees, etc. Once the user is finished positioning the entities, he can ask for them to be deployed to a lower-level (e.g., platoon).

An informal user test was recently run in which an experienced ExInit user (who had created the 60,000 entity scenario) designed his own test scenario involving the creation of 8 units and 15 control measures (e.g., the lines and areas shown in Figure 7). The user first entered the scenario via the ExInit graphical user interface, a standard Microsoft Windows mouse-menu-based GUI. Then, after a relatively short training session with QuickSet, he created the same scenario using speech and gesture. Interaction via QuickSet resulted in a two-fold to seven-fold speedup, depending on the size of the units involved (companies or battalions). Although a more comprehensive user test remains to be conducted, this early data point indicates the productivity gains that can potentially be derived from using multimodal interaction.

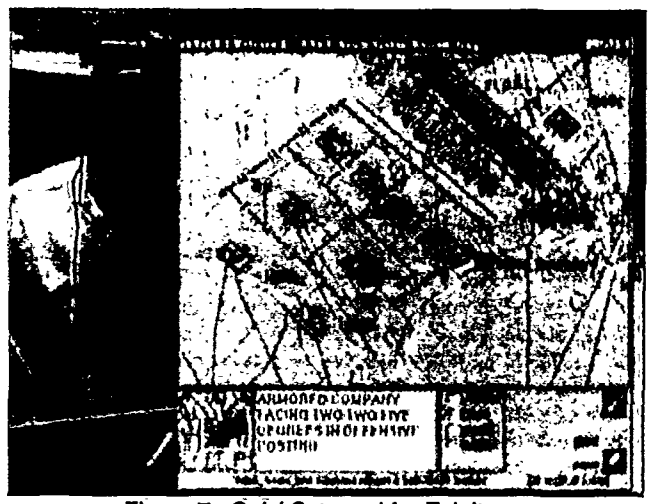
Figure 7: QuickSet used for ExInit — large-scale exercise initialization

5.3 Multimodal Interaction with Medical Information: MIMI

The last example a QuickSet-based application is MIMI, which allows users to find appropriate health care in Portland, Oregon. Working with the Oregon Health Sciences University, a prototype was developed that allows users to inquire using speech and gesture about available health care providers. For example, a user might say "show me all psychiatrists in this neighborhood <circling gesture on map>". The system translates the multimodal input into a query to a database of doctor records. The query results in a series of icons being displayed on the map. Each of these icons contains one or more health care providers meeting the appropriate criterion. Figure 8 show the map-based interaction supported by MIMI.

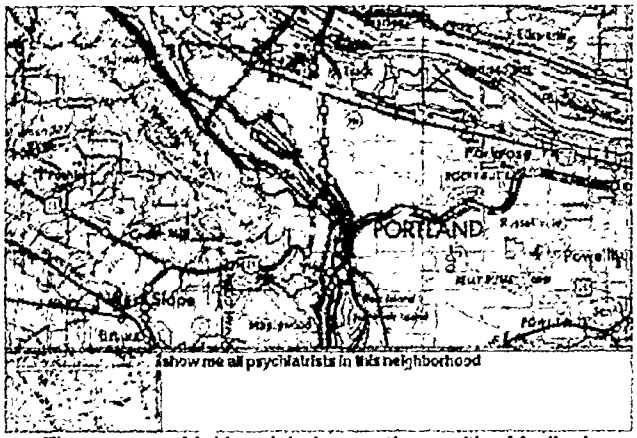

Figure 8: Multimodal Interaction with Medical Information

Users can ask to see details of the providers and clinics, ask follow-up questions, and inquire about transportation to those sites.

In summary, QuickSet provides a multimodal interface to a number of distributed applications, including simulation, force laydown, virtual reality, and medical informatics. The heart of the system is its ability to integrate continuous spoken language and continuous gesture. Section 6 discusses the unification-based architecture that supports this multimodal integration.

6. MULTIMODAL INTEGRATION

Given the advantages of multimodal interaction, the problem of integrating multiple communication modalities is key to future human-computer interfaces. However, in the sixteen years since the "Put-That-There" system [Bolt 1980], research on multimodal integration has yet to yield a reusable scaleable architecture for the construction of multimodal systems that integrate gesture and voice. As we reported in Johnston et al. [1997], we see four major limiting factors in previous approaches to multimodal integration:

- The majority of approaches only consider simple deictic pointing gestures made with a mouse [Brison and Vigouroux (ms.); Cohen 1992; Neal and Shapiro 1991; Wauchope 1994] or with the hand [Bolt, 1980; Koons et al 1993].
- Most previous approaches have been primarily language-driven, treating gesture as a secondary dependent mode [Neal and Shapiro 1991, Cohen 1992; Brison and Vigouroux (ms.), Koons et al 1993, Wauchope 1994]. In these approaches, integration of gesture is triggered by the appearance of expressions in the speech stream whose reference needs to be resolved, such as definite and deictic noun phrases (e.g. 'the platoon facing east,' 'this one', etc.).
- None of the existing approaches provide a well-understood and generally applicable common meaning representation for the different modes.
- None of the existing approaches provide a general and formally-well defined mechanism for multimodal integration.

6.1 Multimodal Architecture Requirements

In order to create such a mechanism we need:

- Parallel recognizers and "understanders" that produce a set of time-stamped meaning fragments for each continuous input stream
- A common framework within which to represent those meaning fragments
- A time-sensitive grouping process that decides *which* meaning fragments from each modality stream should be combined. For example, should the gesture in a sequence of <speech, gesture, speech> be interpreted with the preceding speech, the following speech, or by itself?
- Meaning "fusion" operations that combine semantically compatible meaning fragments. The modality combination operation needs to allow any meaningful part to be expressed in any of the available modalities
- A process that chooses the best *joint* interpretation of the multimodal input. Such a process will support mutual compensation of modes — allowing, for example, speech to compensate for errors in gesture recognition, and vice-versa.
- A flexible asynchronous architecture that allows multiprocessing and can keep pace with human input.

6.2 Overview Of Quickset's Approach To Multimodal Integration

Using a distributed agent architecture, we have developed a multimodal integration process for QuickSet that meets these goals.

- The system employs continuous speech and continuous gesture recognizers running in parallel. A wide range of continuous gestural input is supported, and integration may be driven by either mode.
- Typed feature structures are used to provide a clearly defined and well understood common meaning representation for the modes.
- Multimodal integration is accomplished through unification.
- The integration is sensitive to the temporal characteristics of the input in each mode.
- The unification-based integration method allows spoken language and gesture to compensate for recognition errors in the other modality.
- The agent architecture offers a flexible asynchronous framework within which to build multimodal systems.

In the remainder of this section, we briefly present the multimodal integration method. Further information can be found in [Johnston et al., 1997].

6.3 A Temporally-Sensitive Unification-Based Architecture for Multimodal Integration

One the most significant challenges facing the development of effective multimodal interfaces concerns the integration of input from different modes. In QuickSet, inputs from each mode need to be both temporally and semantically compatible before they will be fused into an integrated meaning.

6.3.1 Temporal compatibility

In recent empirical work [Oviatt et al. 1997], it was discovered that when users speak and gesture in a sequential manner, they

gesture first, then speak within a relatively short time window; speech rarely precedes gesture. As a consequence, our multimodal intepreter prefers to integrate gesture with speech that follows within a short time interval, than with preceding speech. If speech arrives after that interval, the gesture will be interpreted unimodally. This temporally-sensitive architecture requires that there at least be time stamps for the beginning and end of each input stream. However, this strategy may be difficult to implement for a distributed environment in which speech recognition and gesture recognition might be performed by different machines on a network, requiring a synchronization of clocks. For this reason, it is preferable to have speech and gestural processing performed on the same machine.

6.3.2 Semantic compatibility through unification of typed feature structures

Semantic compatibility is captured via unification over typed feature structures [Carpenter 1990, 1992; Calder 1987]. Unification is an operation that determines the consistency of two representational structures, and if they are consistent combines them into a single result. Feature structure unification is a generalization of term unification in logic programming languages, such as Prolog (and is often implemented using term unification). Feature structure unification differs from term unification in logic programming where the features are positionally encoded in a term, in that they are explicitly labeled and unordered in a feature structure.

A feature structure consists of a collection of feature-value pairs. The value of a feature may be an atom, a variable, or another feature structure. When two features structures are unified, a composite structure containing all of the feature specifications from each component structure is formed. Any feature common to both feature structures must not clash in its value. If the values of a common feature are atoms they must be identical. If one is a variable, it becomes bound to the value of the corresponding feature in the other feature structure. If both are variables, they become bound together, constraining them to always receive the same value (if unified with another appropriate feature structure). If the values are themselves feature structures, the unification operation is applied recursively. Importantly, feature structure unification can result in a directed acyclic graph structure when more than one value in the collection of feature/values pairs makes use of the same variable. Whatever value is ultimately unified with that variable thus will fill the value slot of all the corresponding features, resulting in a DAG.

Typed feature structures are an extension of the representation whereby feature structures and atoms are assigned to hierarchically ordered types. Typed feature structure unification requires pairs of feature structures or pairs of atoms which are being unified to be compatible in type. To be compatible in type, one must be in the transitive closure of the subtype relation with respect to the other. The result of a typed unification is the more specific feature structure or atom in the type hierarchy.

Typed feature structure unification is ideally suited to the task of multimodal integration because we want to determine whether a given piece of gestural input is compatible with a given piece of spoken input, and if they are compatible, to combine the two inputs into a single result that can be interpreted by the system. Unification is appropriate for multimodal integration because it can combine complementary or redundant input from both modes [5]but rules out contradictory inputs.

6.3.3 Advantages of typed feature structure unification

We identify four advantages of using typed feature structure unification to support multimodal integration — partiality, mutual compensation, structure sharing, and multimodal discourse. These are discussed below.

Partial meaning representations. The use of feature structures as a semantic representation framework facilitates the specification of partial meanings. Spoken or gestural input which partially specifies a command can be represented as an underspecified feature structure in which certain features are not instantiated, but are given a certain type based on the semantics of the input. For example, if a given speech input can be integrated with a line gesture, it can be assigned a feature structure with an underspecified location feature whose value is required to be of type *line*, as in Figure 9 where the spoken phrase 'barbed wire' is assigned the feature structure shown.

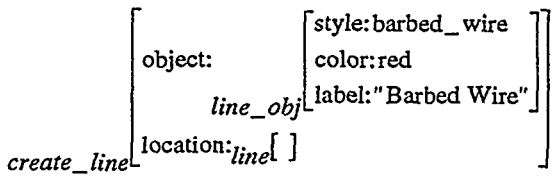

Figure 9: Feature Structure for 'barbed wire'

Since QuickSet is a task-based system directed toward setting up a scenario for simulation, this phrase is interpreted as a partially specified creation command. Before it can be executed, it needs a location feature indicating where to create the line, which is provided by the user's drawing on the screen. The user's ink is likely to be assigned a number of interpretations, for example, both a point interpretation and a line interpretation, which are represented as typed feature structures (see Figures 10 and 11). Interpretations of gestures as location features are assigned the more general *command* type which unifies with all of the commands supported by the system, one of which is *create_line* (see Figure 9).

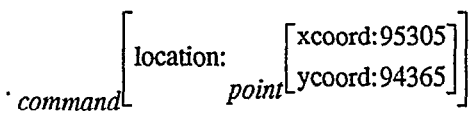

Figure 10: Point Interpretation of Gesture

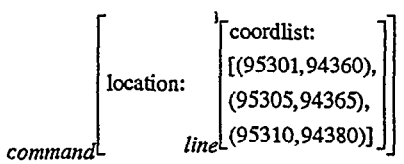

Figure 11: Line Interpretation of Gesture

Multimodal Compensation. In the example case above, both speech and gesture have only partial interpretations, one for speech, and two for gesture. Since the speech interpretation (Figure 7) requires its location feature to be of type *line*, only unification with the line interpretation of the gesture will

[5] Redundant multimodal input occurs infrequently in map-based tasks [Oviatt and Olsen, 1994; Oviatt et al. 1977].

succeed and be passed on as a valid multimodal interpretation (Figure 12).

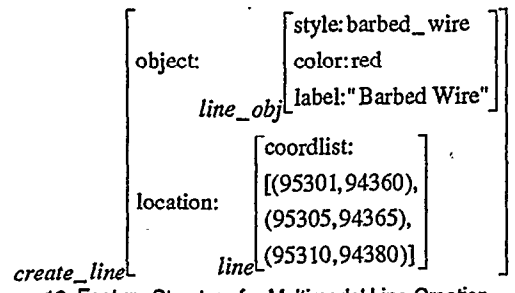

Figure 12: Feature Structure for Multimodal Line Creation

The ambiguity of interpretation of the gesture was resolved by integration with speech, which in this case required a location feature of type *line*. If the spoken command had instead been 'M1A1 Platoon', intending to create an entity at the indicated location, it would have selected the point interpretation of the gesture in Figure 10. Similarly, if the spoken command described an area, for example a swamp, it would only unify with an interpretation of gesture as an area designation. In each case the unification-based integration strategy compensates for errors in gesture recognition through type constraints on the values of features.

Gesture also compensates for errors in speech recognition. As a simple example, in the open microphone mode, spurious speech recognition errors are more common than with click-to-speak, but are frequently rejected by the system because of the absence of a compatible gesture for integration. For example, if the system recognizes 'M1A1 platoon', but there is no overlapping or immediately preceding gesture to provide the location, the speech will be ignored. More generally, the architecture also supports selection among the n-best speech recognition results on the basis of the preferred gesture recognition. We obtain the best joint interpretation using the maximum of the sum of the log probabilities of the spoken and gestural interpretations among the semantically and temporally compatible joint interpretations. We are currently engaged in quantifying the benefits observed by this mutually compensatory recognition process.

Structure Sharing. Another advantage of typed feature structure unification is the use of shared variables among elements of the feature structure. For example, if the user says "M1A1 platoon facing this way <draws arrow>", in the resulting feature structure, the orientation feature of the command is structured-shared with the angle of its location feature. When it is unified with an arrow gesture feature structure, the orientation feature is automatically instantiated with the angle at which the arrow was drawn.

Multimodal Discourse. The user can explicitly enter into a "mode" in which s/he is creating a specific type of entity, for example, M1A1 platoons, by simply saying "multiple M1A1 platoons." This results in a more specific feature structure that will subsequently be unified with future input (Figure 13).

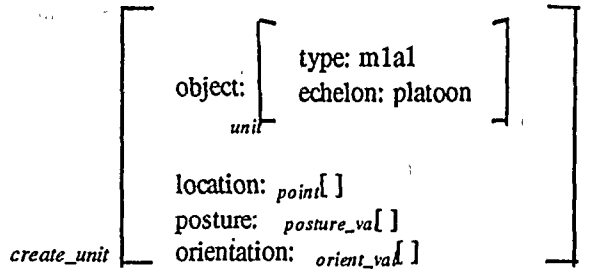

Figure 13: Feature structure for the "mode" of creating M1A1 platoons"

For example, the user could then place the pen at a desired location and say "whiskey four six," intending to create an M1A1 platoon named "W46" at that location. Any phrase resulting in a structure that unifies with the type of entity that is being created will result in the creation of that more specific type of entity. For instance, the subsequent utterances "whiskey four seven facing southeast," "whiskey four eight oriented one hundred and thirty five degrees," (see Figure 7), result in the creation of units with those names and orientations. When there is no interpretation that unifies with the one initially specified, the "mode" is ended.

In summary, we have identified four main advantages to using unification of typed feature structures as the core of a multimodal integration process: partiality, mutual compensation, structure sharing, and multimodal discourse. In virtue of these capabilities, the QuickSet system is now a usable testbed for experimenting with multimodal architectures, and for developing next-generation multimodal systems.

Vo and Wood [1996] and Waibel et al., [1995] present an approach to multimodal integration similar in spirit to that presented here in that it accepts a variety of gestures and is not solely speech-driven. However, we believe that unification of typed feature structures provides a more general, formally well-understood, and reusable mechanism for multimodal integration than the frame merging strategy that they describe. In particular, the unification approach allows for DAG interpretations and supports multimodal discourse in an elegant way. Cheyer and Julia [1995] sketch a system based on Oviatt's [1996] results and the Open Agent Architecture [Cohen et al., 1994], but describe neither the integration strategy nor multimodal compensation.

7. CONCLUDING REMARKS

QuickSet has been delivered to the US Navy and US Marine Corps. for use at Twentynine Palms, California, where it is primarily used to set up training scenarios and to control the virtual environment. The system was also used by the US Army's 82 Airborne Corps. at Ft. Bragg during the Royal Dragon Exercise. There, QuickSet was deployed in a tent, where it was subjected to noise from explosions, low-flying jet aircraft, generators, etc. Not surprisingly, it readily became apparent that spoken interaction with QuickSet would not be feasible. To support usage in such a harsh environment, a complete overlap in functionality between speech, gesture, and direct manipulation was desired. The system has been revised to accommodate these needs. As part of ExInit, QuickSet is being delivered to STRICOM, the US Army's Simulation and Training Command for use in DARPA's STOW-97 Advanced Concept Demonstration.

Regarding the multimodal interface itself, QuickSet has undergone a "proactive" interface evaluation in that high-

fidelity "wizard-of-Oz" studies were performed in advance of building the system, which predicted the utility of multimodal over unimodal speech as an input to map-based systems [Oviatt, 1996; Oviatt et al., 1997]. For example, it was discovered there that multimodal interaction would lead to simpler language than unimodal speech. Such observations have been confirmed when examining how users would create linear features with CommandTalk [Moore et al., 1997], a unimodal spoken system that also controls LeatherNet. Whereas to create a "phase line" between two three-digit <x,y> grid coordinates, a user would have to say: "create a line from nine four three nine six one to nine five seven nine six eight and call it phase line green," a QuickSet user would say "phase line green" while drawing a line. Given that numerous difficult-to-process linguistic phenomena (such as utterance disfluencies) are known to be elevated in lengthy utterances and also to be elevated when people speak locative constituents [Oviatt, 1996; Oviatt in press], multimodal interaction that permits pen input to specify locations offers the possibility of more robust recognition.

In summary, we have developed a handheld system that integrates numerous advanced technologies, including speech recognition, gesture recognition, natural language processing, multimodal integration, distributed agent technologies, and reasoning. The multimodal integration strategy allows speech and gesture to compensate for each other, yielding a more robust system. We are currently engaged in evaluation experiments to quantify the benefits of this approach. The system interoperates with existing military simulators and virtual reality environments through a distributed agent architecture. QuickSet has been deployed for the US Navy, US Marine Corps, and the US Army, and is being integrated into the DARPA STOW-97 ACTD. We are currently evaluating its performance in the field.

ACKNOWLEDGMENTS

This work is supported in part by the Information Technology and Information Systems offices of DARPA under contract number DABT63-95-C-007, in part by ONR grant number N00014-95-1-1164, and has been done in collaboration with the US Navy's NCCOSC RDT&E Division (NRaD), Ascent Technologies, MRJ Corp. and SRI International.

REFERENCES

Bolt, R. A. 1980. "Put-That-There":Voice and gesture at the graphics interface. *Computer Graphics*. 14.3, pp. 262-270.

Baker, M. P., and Wickens, C. D. Human factors in virtual environments for the visual analysis of scientific data. Unpublished ms., University of Illinois.

Brooks, Frederic. 3-D user interfaces: When results matter, Invited presentation (unpublished), UIST'96, Seattle, 1996.

Brison, E. and Vigouroux, N. (unpublished ms.). Multimodal references: A generic fusion process. URIT-URA CNRS. Université Paul Sabatier, Toulouse, France.

Calder, J. 1987. Typed unification for natural language processing. In E. Klein and J. van Benthem (Eds.), Categories, Polymorphisms, and Unification. Centre for Cognitive Science, University of Edinburgh, Edinburgh, pp. 65-72.

Cheyer, A., and Julia L. 1995. Multimodal maps: An agent-based approach. *International Conference on Cooperative Multimodal Communication (CMC/95)*, May 1995. Eindhoven, The Netherlands. pp. 24-26.

Carpenter, R. 1990. Typed feature structures: Inheritance, (In)equality, and Extensionality. In W. Daelemans and G. Gazdar (Eds.), *Proceedings of the ITK Workshop: Inheritance in Natural Language Processing*, Tilburg University, pp. 9-18.

Carpenter, R. 1992. *The logic of typed feature structures*. Cambridge University Press, Cambridge.

Christensen, J., Marks, J.., and Shieber, S. Placing text labels on maps and diagrams, *Graphics Gems IV*, Heckbert, P. (ed.), Academic Press, Cambridge, Mass., 1994, 497-504.

Clarkson, J. D., and Yi, J. LeatherNet: A synthetic forces tactical training system for the USMC commander. *Proceedings of the Sixth Conference on Computer Generated Forces and Behavioral Representation*. Orlando, Florida, 1996 275-281.

Cohen, P. R. The Role of Natural Language in a Multimodal Interface. *Proceedings of UIST'92*, ACM Press, NY, 1992, 143-149.

Cohen, P. R., Dalrymple, M. , Moran, D. B., Pereira, F. C. N., Sullivan, J. W., Gargan, R. A., Schlossberg, J. L., Tyler, S. W., Synergistics use of direct manipulation and natural language, *Proceedings of Human Factors in Computing Systems* (CHI'89), ACM Press, New York, 1989, 227-234.

Cohen, P. R., Cheyer, A., Wang, M., and Baeg, S.C. An Open Agent Architecture. *Proceedings of the AAAI Spring Symposium Series on Software Agents* (March 21-22, Stanford), Stanford Univ., CA, 1994, 1-8.

Courtemanche, A.J. and Ceranowicz, A. ModSAF Development Status. *Proceedings of the Fifth Conference on Computer Generated Forces and Behavioral Representation*, Univ. Central Florida, Orlando, 1995, 3-13.

Cruz-Neira, C. D. J. Sandin, T. A. DeFanti, "Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE," *Computer Graphics*, ACM SIGGRAPH, August 1993, pp. 135-142.

Johnston, M., Cohen, P. R., McGee, D., Pittman, J., Oviatt, S. L., and Smith, I. Unification-based multimodal integration, *Proceedings of the Annual Meeting of the Association for Computational Linguistics* (ACL-97/EACL-97) Conference, Madrid, Spain, July, 1997.

Koons, D.B., C .J. Sparrell and K. R. Thorisson. 1993. Integrating simultaneous input from speech, gaze, and hand gestures. In Mark T. Maybury (ed.) *Intelligent Multimedia Interfaces*. AAAI Press/ MIT Press, Cambridge, MA, pp. 257-276.

Manke, S., Finke, M., and Waibel, A., The use of dynamic writing information in a connectionist on-line cursive handwriting recognition system, *Advances in Neural Information processing Systems 7* (NIPS), 1994.

Moore, R. C., Dowding, J, Bratt, H., Gawron, M., and Cheyer, A. CommandTalk: A spoken-language interface for battlefield simulations, *Proc. of the 3rd Applied Natural Language Conference*, Wash. DC, 1997.

Neal, J.G. and Shapiro, S.C. Intelligent multi-media interface technology. In J.W. Sullivan and S.W. Tyler, editors, *Intelligent User Interfaces*, chapter 3, pages 45–68. ACM Press Frontier Series, Addison Wesley Publishing Co., New York, New York, 1991.

Oviatt, S. L. Pen/Voice: Complementary multimodal communication, *Proceedings of SpeechTech'92*, New York, February, 1992, 238-241.

Oviatt, S.L. Multimodal interfaces for dynamic interactive maps. *Proceedings of CHI'96 Human Factors in Computing Systems* ACM Press, NY, 1996, 95-102.

Oviatt, S.L. Multimodal interactive maps: Designing for human performance, *Human Computer Interaction*, in press.

Oviatt, S. L, A. DeAngeli, and K. Kuhn. Integration and synchronization of input modes during multimodal human-computer interaction. *Proceedings of the Conference on Human Factors in Computing Systems (CHI '97)*, ACM Press, NY, 1997, 415-422.

Oviatt, S. L., and Olsen, E., Integration themes in multimodal human-computer interaction, *Proceedings of the International Conference on Spoken Language Processing*, Acoustical Society of Japan, Yokohama, Japan, 1994, 551-554.

Pittman, J. A. Recognizing handwritten text *Human Factors in Computing Systems (CHI'91)*, 1991, 271-275.

Stoakley, R., Conway, M., and Pausch, R. Virtual reality on a WIM: Interactive worlds in miniature, *Proceedings. of Human Factors in Computing Systems (CHI'95)*, ACM Press, New York, 1995, 265-272.

ACM Multimedia *97 Seattle Washington USA*

Thorpe, J. A. The new technology of large scale simulator networking: Implications for mastering the art of warfighting. *9th Interservice Training Systems Conference*, 1987, 492-501.

Vo, M. T. and Wood, C. Building an application framework for speech and pen input integration in multimodal learning interfaces. *International Conference on Acoustics, Speech, and Signal Processing*, Atlanta, GA. 1996.

Waibel, A., Vo, M. T., Duchnowski, P., and Manke, S. Multimodal interfaces, *Artificial Intelligence Review*, 1995.

Wauchope, K. 1994. Eucalyptus: Integrating natural language input with a graphical user interface. Naval Research Laboratory, Report NRL/FR/5510--94-9711.

Zyda, M. J., Pratt, D. R., Monahan, J. G., and Wilson, K. P., NPSNET: Constructing a 3-D virtual world, *Proceedings of the 1992 Symposium on Interactive 3-D Graphics*, March, 1992.