Multimodal Interaction for Distributed Interactive Simulation

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 Artificial Intelligence research to distributed interactive simulations, with the goal of reducing exercise generation time and effort, yet maximizing training effectiveness. 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 NRaD's[1] LeatherNet system, a distributed interactive training simulator built for the US Marine Corps. The paper describes our novel multimodal integration strategy offering mutual compensation among modalities, as well as QuickSet's agent-based infrastructure, and provides an example of multimodal simulation setup. Finally, we discuss our applications experience and lessons learned.

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

1. INTRODUCTION

In order to train personnel more effectively, the US Government is developing large-scale military simulation capabilities. Begun as SIMNET in the 1980's [Thorpe, 1987], these distributed, interactive environments attempt to provide a high degree of fidelity in simulating combat equipment, movement, atmospheric effects, etc. Numerous workstation-based simulators that share a replicated and distributed database are networked together worldwide to provide a computational substrate. A rather ambitious goal for 1997 is to be able to create and simulate a large scale exercise, in which there may be on the order of 50,000 entities (e.g., a vehicle or a person). A major goal of the Government, as well as of the present research, is to develop technologies that can aid in substantially reducing the time and effort needed to create scenarios.

The Simulation Process

There are four general phases of user interaction with these simulations: creating entities, supplying their initial behavior, interacting with the entities during a running simulation, and reviewing the results. In the first phase, a user "lays down" or places forces on the terrain that need to be positioned in realistic ways, given the terrain, mission, available equipment, etc. In addition to positioning these entities at the start of a simulation, the user needs to supply them with behavior, which may involve complex maneuvering, communication, etc. While the simulation is being run, human controllers may observe ongoing events, and change the behavior of those entities to react appropriately to those events. Finally, after a simulation is run, a user will often want to review and query the resulting simulation history. This paper discusses the first two of these phases, while our current and future research addresses the remaining two.

2. THE SIMULATION INTERFACE

Our contribution to the distributed interactive simulation (DIS) effort is to rethink the nature of the user interaction. As with most modern simulators, DISs are controlled via graphical user interfaces (GUIs). However, for a number of reasons, GUI-based interaction is rapidly losing its benefits, especially when large numbers of entities need to be created and controlled. 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.

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

3. QUICKSET

To address these simulation interface problems, we have developed QuickSet (see Figure 2) a collaborative, handheld, multimodal system for configuring military simulations based on LeatherNet [Clarkson and Yi, 1996], a system used 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. In addition to LeatherNet, QuickSet is being used in a second effort called ExInit (Exercise Initialization), that will enable users to create brigade-level exercises.

A major design goal for QuickSet was to provide the same user interface for handheld, desktop, and wall-sized terminal hardware. We believe that only gesture and speech-based interaction comfortably span this range. However, rather than provide just one of these modalities, QuickSet offers both 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].

QuickSet offers speech and pen-based gesture input on multiple 3-lb hand-held PCs (Fujitsu Stylistic 1000), which communicate via wireless LAN through the Open Agent Architecture (OAA) [Cohen et al., 1994], to ModSAF, and to CommandVu. With this highly portable device, a user can create entities, establish "control measures" (e.g., objectives, checkpoints, etc.), draw and label various lines and areas, (e.g., landing zones) and give the entities behavior; see Figure 1, where the user has said "M1A1 platoon follow this route <draws curved line >."

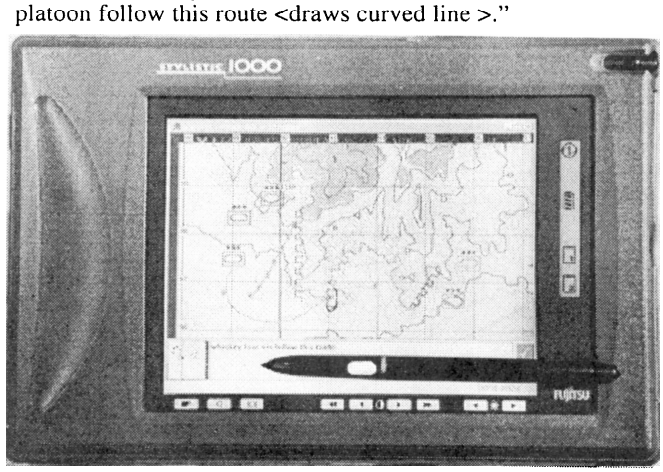

Figure 1: QuickSet running on a wireless handheld PC. The user has created numerous units, fortifications and objectives.

In the remainder of the paper, we describe the system components, the multimodal interface architecture, and the agent infrastructure that integrates both the components of QuickSet, as well as the military applications. We discuss its application, and finally lessons we have learned, particularly about the agent architecture.

4. SYSTEM ARCHITECTURE

Architecturally, QuickSet uses distributed agent technologies based on the Open Agent Architecture[2] for interoperation, information brokering and distribution. An agent-based architecture was chosen to support this application because it offers easy connection to legacy applications, and the 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 workstations and PCs. Additionally, the architecture supports mobility in that lighter weight agents can run on the handheld, while more computationally-intensive processing can be migrated 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 the Interagent Communication Language (see below). The configuration of agents used in the Quickset system is illustrated in Figure 2. The architecture is described in detail in Section 6. A brief description of each agent, relevant to the user interface portions of the QuickSet system follows.

QuickSet interface: On the handheld PC is a geo-referenced map of the region 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 agent provides the usual pan and zoom capabilities, multiple overlays, icons, etc. The user can draw directly on the map, in order to create points, lines, and areas. The user can create entities, give them behavior, and watch the simulation unfold from the hand-held. When the pen is placed on the screen, the speech recognizer is activated, thereby allowing users to speak and gesture simultaneously.

Speech recognition agent: The speech recognition agent used in QuickSet is built on IBM's VoiceType Application Factory. The recognizer uses an HMM-based continuous speaker-independent speech recognition technology for PCs under Windows 95/NT and OS/2. Currently, it produces a single most likely interpretation of an utterance. One copy of this agent runs on every PC in the QuickSet system.

Gesture recognition agent: OGI's gesture recognition agent collects and processes all pen input from the PC screen or tablet. The recognizer sends an n-best list of possible interpretations to the facilitator, who forwards the recognition results to the multimodal integration agent. A gesture recognition agent runs on each PC in the Quickset architecture. A more detailed description of the gesture recognition agent is in Section 6.

[2] Open Agent Architecture is a trademark of SRI International

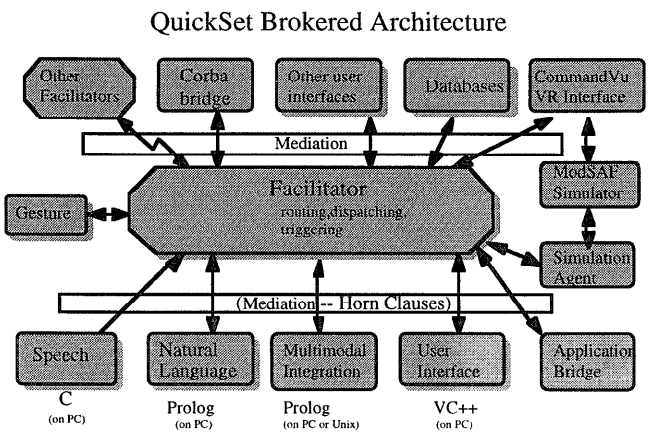

Figure 2: The blackboard serves as a facilitator, channeling queries to agents who claim they can solve them.

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 this task, the language consists of noun phrases that label entities, as well as a variety of imperative constructs for supplying behavior.

Multimodal integration agent: The multimodal interpretation agent accepts typed feature structure meaning representations from the language and gesture recognition agents. It unifies those feature structures together, producing a multimodal interpretation. A more detailed description of multimodal interpretation is in Section 5.

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 using a suitable browser.

Other user interfaces: When another user interface connected to the facilitator, subcribes to and produces the same set of events as others, it immediately becomes part of a collaboration. One can view this as human-human collaboration mediated by the agent architecture, or as agent-agent collaboration.

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. For example, the user can ask "CommandVu, fly me to this platoon <gesture on the map>."

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.

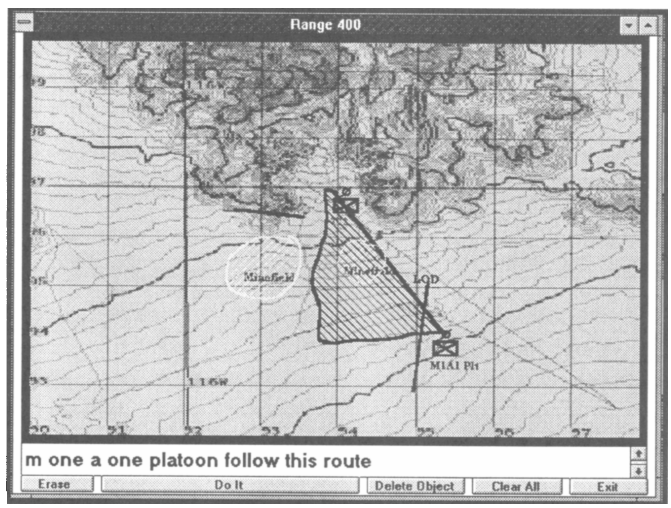

Figure 3: The QuickSet interface as the user establishes two platoons, a barbed-wire fence, a breached minefield, and issues a command to the platoon in the southeast to follow a specified route.

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

4. EXAMPLE

Holding QuickSet in hand, the user views a map from the ModSAF simulation, and with spoken language coupled with pen gestures, issues commands to ModSAF. 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." Similarly a fortified line is added. A minefield of an amorphous shape is drawn and is labeled verbally, and finally an M1A1 platoon is created as above. Then the user can assign a task to the new platoon by saying "M1A1 platoon follow this route" while drawing the route with the pen. The results of these commands are visible on the QuickSet screen, as seen in Figure 3, as well as on the ModSAF simulation, which has been executing the user's QuickSet commands (Figure 4). They are also visible in the CommandVu 3D rendering of the scene.

Since any recognizer will make mistakes, the output of the gesture recognizer is not accepted as a simple decision. Instead the recognizer produces a set of probabilities, one for

each possible interpretation of the gesture. The identities and types of objects, units, points, lines, and areas that have been drawn, tapped on, or encircled by the pen, as well as their recognition probabilities, are sent to the blackboard. Through triggers, the multimodal interpretation agent is sent these candidate interpretations.

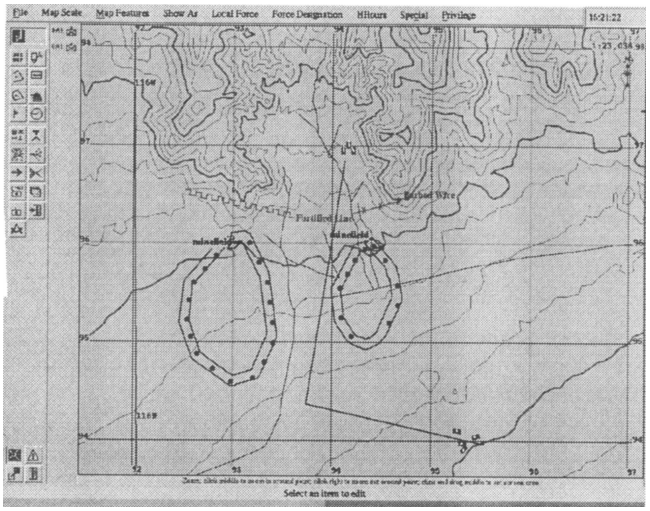

Figure 4: The ModSAF interface showing the forces and obstacles created from QuickSet.

In combining the meanings of the gestural and spoken interpretations, we attempt to satisfy an important design consideration, namely that the communicative modalities should compensate for each other's weaknesses [Cohen, 1992; Oviatt, 1992]. Here, we deliberately show an early version of QuickSet because it illustrates graphically that the unification-based multimodal integration process discussed in Section 7 has ruled out the higher-scoring area interpretation of the gesture (the shaded region in Figure 3) in favor of the unifying interpretation of the gesture as a linear route ("dog-legged" line in Figure 4), which was sent to ModSAF.[3] In response to this multimodal command, the automated platoon follows the route, sidesteps the minefields, breaches the fortifications, and engages in combat at the destination.

More detail on the gesture agent, multimodal integration, and agent architecture are provide below, since these artificial intelligence technologies are novel contributions of our effort.

6. GESTURE RECOGNITION

In order increase accuracy, QuickSet's pen-based gesture recognizer consists of both a neural network [Pittman, 1991] 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 smoothed, resampled, converted to deltas, and given as input to the HMM recognizer. The system currently recognizes 68 pen-gestures (including units, lines of various types, objectives, etc.), and new ones are being added continually.

Both recognizers provide the same coverage (they recognize the same set of gestures). These gestures, some of which are illustrated in Figure 5, include 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. Both recognizers recognize shape (although they see the shape in different data formats).

But as Figures 5 and 6 show, route and area indications may have a variety of shapes. This problem is further compounded by the fact that we want the recognizer to be robust in the face of sloppy writing. More typically, sloppy forms of various map symbols, such as those illustrated in Figure 6, 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.

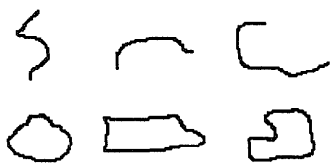

Figure 5: Pen drawings of routes and areas. Routes and areas do not have signature shapes that can be used to identify them.

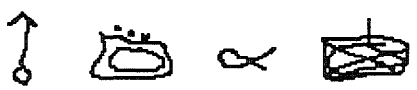

Figure 6: Typical pen input from real users. The recognizer must be robust in the face of sloppy input.

7. A 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. The major contribution of the present work is an architecture in which speech and gesture can compensate for errors in the other modality.

To model this integration, we utilize a unification operation over typed feature structures [Carpenter 1990, 1992; Calder 1987]. Unification is an operation that determines the consistency of two pieces of partial information, and if they are consistent combines them into a single result. As such, it is ideally suited to the task at hand, in which we want to determine whether a given piece of gestural input is

[3] In the present version of the system, the ultimate interpretation is reflected on the QuickSet screen

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.

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. 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*.

The spoken phrase 'barbed wire' is assigned the feature structure in Figure 7.

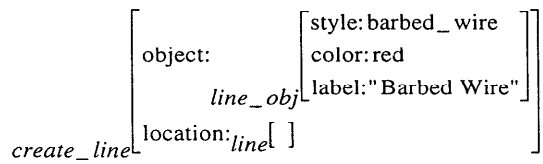

Figure 7: Feature Structure for 'barbed wire'

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 8 and 9). Interpretations of gestures as location features are assigned the more general *command* type which unifies with all of commands taken by the system.

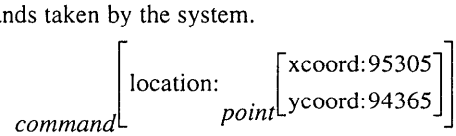

Figure 8: Point Interpretation of Gesture

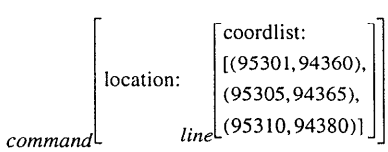

Figure 9: Line Interpretation of Gesture

The task of the integrator agent is to field incoming typed feature structures representing interpretations of speech and of gesture, identify the best potential interpretation, multimodal or unimodal, and issue a typed feature structure representing the preferred interpretation to the bridge agent, which will execute the command.

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 succeed and be passed on as a valid multimodal interpretation (Figure 10).

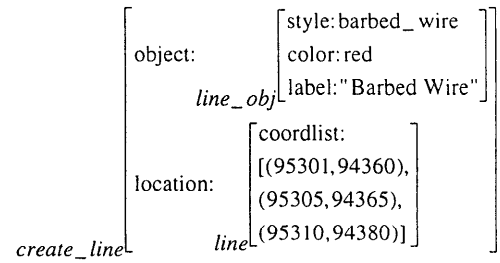

Figure 10: Feature Structure for Multimodal Line Creation

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 8.

Similarly, if the spoken command described an area, for example an 'anti tank minefield', 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. 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. The architecture also supports selection among the n-best speech recognition results on the basis of the preferred gesture recognition. In the future, n-best recognition results will be available from the recognizer, and we will further examine the potential for gesture to help select among speech recognition alternatives.

COMPARISON WITH OTHER MULTIMODAL INTEGRATION EFFORTS

Systems capable of integration of speech and gesture have existed since the early 80's. One of the first such systems was the "Put-That-There" system [Bolt 1980]. However, in the sixteen years since then, research on multimodal integration has not yielded a reusable scalable architecture for the construction of multimodal systems that integrate gesture and voice. There are four major limiting factors in previous approaches to multimodal integration:

(i) The majority of approaches limit the bandwidth of the gestural mode to simple deictic pointing gestures made with a mouse [Brison and Vigouroux (ms.); Cohen 1992; Neal

and Shapiro 1991; Wauchope 1994] or with the hand [Koons et al 1993].

(ii) 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 systems, 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. 'this one', 'the red cube').

(iii) None of the existing approaches provide a well-understood generally applicable common meaning representation for the different modes.

(iv) None of the existing approaches provide a general and formally-well defined mechanism for multimodal integration.

Our approach to multimodal integration overcomes these limiting factors in that 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, and multimodal integration is accomplished through unification. Vo and Wood [1996] 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. 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.

8. AGENT INFRASTRUCTURE

Major considerations in designing QuickSet have been: interoperation with legacy systems running in a heterogeneous computing environment, modularity, network transparency and distributed operation, scalability, and collaboration. A requirement of the infrastructure was that it be able to respond fast enough to provide near real-time performance from a multimodal user interface running on a small, handheld PC. The entire architecture had to run standalone on a 486/100Mhz PC, but be able to be reconfigured rapidly when more computing resources were available on a LAN.

The Open Agent Architecture satisfies most of these requirements. The architecture is based on FLiPSiDE [Schwartz, 1993], an enhanced distributed Prolog-based blackboard architecture. In the traditional blackboard model, individual knowledge sources (agents) communicate by posting and reading messages on a common blackboard. An agent will periodically poll the board to see if there are any posted goals (from other agents) it can solve; when an agent needs help, it can post a goal to be solved, then retrieve the answer when it appears on the board. The OAA model enhances this with a facilitator agent resident on the blackboard. This facilitator stores the blackboard data, identifies agents that can solve particular posted goals and routes requests to the appropriate agents.

All communication among the agents takes place through the facilitator agent. In addition to the standard blackboard operations of posting and reading, agents in an OAA can send queries to the facilitator agent and they can request the facilitator to set triggers on itself, or to route triggers to agents who can satisfy the trigger's conditions. These requests are expressions in an Interagent Communication Language (ICL) consisting of Horn clauses.

When an agent registers with the facilitator agent, it supplies a list of goals it can solve as an argument. The facilitator will add the agent to its list of available knowledge sources, recording the goals in the supplied parameter list. The current semantics of this registration are those of an assertion followed by a conditional offer. That is, not only is the registering agent asserting that it can solve the goals in the argument list, it is also offering to do so whenever requested by the facilitator.

Whenever a goal to be solved is posted to the blackboard, the facilitator routes the goal to those registered agents that have claimed to be able (and willing) to solve it. These solution-requests use synchronous communication — the agent posting the solve will block until a solution (or failure) is reported by the facilitator. The OAA also provides an asynchronous query facility —allowing the agent who posts the request to continue local computation while waiting for a response. Through the use of triggers, the agent can request that the facilitator notify it when an event has occurred, where an event can be any phenomenon in the distributed system describable in the ICL. In particular, the event in question could be the appearance of a solution to an asynchronous request posted earlier. Most of the QuickSet integration is based on asynchronous agent interaction, thereby allowing multiple agents to engage in interactions with centralized agents (e.g., the simulation agent) without blocking or waiting for others. This is particularly important for user interfaces, which must respond to speech and gesture input within a very short time window, even while the system's responses to prior inputs are being computed.

Although this architecture is derived from FliPSiDe, it omits Schwartz' considerable effort devoted to locking of the blackboard. (Similar considerations were vital in the early blackboard systems [Fennel and Lesser, 1977].) The architecture is similar to that of KQML/KIF [Finin et al., 1994; Genesereth and Ketchpel, 1994] in its use of a brokered architecture, an agent communications language and a logically-based propositional content language. We employ many fewer speech act types (and have argued that in fact, KQML should only have a small number of communication acts as well [Cohen and Levesque, 1995; Smith and Cohen, 1996]) and use a subset of first-order logic, where that effort uses KIF. The OAA has provided an effective framework for integrating legacy applications. By linking in an agent library written in the application's native language, the application can become a full-fledged agent, potentially capable of participating in multiagent systems. To date, the architecture has been sufficient for our

initial experiments, but needs to be revised significantly for data and interaction intensive applications.

8. CONCLUDING REMARKS

Below we characterize how QuickSet has been applied and what we have learned from this research, development and technology transition experience.

Applications

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 accomodate these needs.

Finally, QuickSet is now being extended for use in the ExInit simulation initialization system for DARPA's STOW-97 (Synthetic Theater of War) Advanced Concept Demonstration. In this version, QuickSet runs either on a Pentium Pro or on the handheld unit, with the agent architecture interoperating with a collection of CORBA servers (written by MRJ Corp. and Ascent Technologies) that decompose high-level units into their constituents, and that perform terrain reasoning, and other "back-end" functions. The multimodal integration here is used to create and position the high level units, and to mark up boundaries, objectives, and obstacles on a map.

Lessons learned

Much was learned from the effort to date about the scalability of agent architectures and about multimodal user interfaces. Although a conceptually reasonable first step, the implementation of the OAA agent architecture lacked necessary features for supporting robust interaction and collaboration. First, it had no features for authentication, or for locking, as in FliPSiDe. Thus, one user's agents could interfere with another's. This was addressed through better identification of the user behind each agent. In an environment where there is one multimodal integration agent on the network, this approach prevents one user's speech from being combined with another user's gesture. Second, the implementations in Prolog and C were not multi-threaded. Thus, when multiple users were rapidly creating units, the simulator agent would lose data and only create the last of the units received. Finally both control and data were routed through the blackboard to the various agents. This works well when the amount of data is relatively small, but will not scale for multimedia applications. Rather, data and control paths should be separated. Research is ongoing at OGI to redesign an agent architecture that overcomes the above limitations.

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 that 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, 1995], 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 artificial intelligence 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, interacting with the end users of the system in the various services, and will begin to collect field usage data during future exercises.

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

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.

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

Cheyer, A., and L. Julia. 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.

Clarkson, J. D., and J. Yi. 1996. 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, 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., 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.

Cohen, P. R. and Levesque, H. J. Communicative actions for artificial agents. In *Proceedings of the International Conference on Multiagent Systems,* AAAI Press, Menlo Park, California, 1995.

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.

Fennell, R. D., and Lesser, V. R., Parallelism in artificial intelligence problem solving: A case study of HEARSAY-II, *IEEE Transactions on Computers C-26(2),* 1977, 98-111.

Finin, T. and Fritzson, R. and McKay, D. and McEntire, R. KQML as an agent communication language, *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94),* ACM Press, New , November, 1994.

Genesereth, M. and Ketchpel, S., Software Agents, *Communications of the ACM, 37(7),* July, 1994, 100-105.

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.

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.

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

Schwartz, D.G. Cooperating heterogeneous systems: A blackboard-based meta approach. Technical report 93-112, Center for Automation and Intelligent Systems Research, Case Western Reserve University, Cleveland, Ohio, April 1993. Ph.D. thesis.

Smith, I. A. and Cohen, P. R. Toward a Semantics for an Agent Communications Language Based on Speech-Acts *Proceedings of the Thirteenth National Conference in Artificial Intelligence* (AAAI 96), AAAI Press, 24-31.

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

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.