Social Interaction in Collaborative Engineering Environments

by

Joon Suk Hor

S.B. Environmental Engineering Science, Massachusetts Institute of Technology, 1998

Submitted to the Department of Civil and Environmental Engineering
in partial fulfillment of the requirements for the degree of

Master of Engineering in Civil and Environmental Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1999

Author _____
Department of Civil and Environmental Engineering
May 18, 1999

Certified by _____
Feniosky Peña-Mora
Assistant Professor of Civil and Environmental Engineering
Thesis Advisor

Accepted by _____
Andrew J. Whittle
Chairman, Department Committee on Graduate Studies

Social Interaction in Collaborative Engineering Environments

by

Joon Suk Hor

Submitted to the Department of Civil and Environmental Engineering
on May 18, 1999, in partial fulfillment of the
requirements for the degree of

Master of Engineering in Civil and Environmental Engineering

Abstract

Communication between humans consists of more than just the verbal component that we all associate with common human interaction. Nonverbal elements are essential parts of full and effective form of communication and conveyance of idea from one individual to another. Nonverbal factors include such things as facial expressions, gestures, body movements, stance, and gaze. These factors are unconsciously and unintentionally present in the communication ritual of humans, and full conveyance of a person's ideas or emotions cannot be realized and transferred to another in absence of these factors. It is only when these nonverbal aspects of communication are represented that human social interaction can occur in its complete form. This holds true for face-to-face interactions as well as when these interactions take place over a virtual collaborative environment via a networked computer system.

This thesis explores human social interactions within the context of virtual collaborative work environments. Specifically, this thesis will examine the role of gestures in personal expression in human social interaction as it pertains to collaborative engineering environment where users or the involved parties are interested in collaborating in a geographically distributed setting to achieve a certain goal in their work process. The significance of social interaction, composed of personal expression and social feedback elements, in a collaborative interaction setting is discussed. Gestures, a specific element of nonverbal behavior that plays an integral role in communication that takes place in human interactions, is explored in particular. With the aim of enabling a more complete social interaction capability for a user of a collaborative system, a gesture expression prototype is designed. The proposed design will allow certain gestures to be made at will by the user with the effect of increasing the level of personal expression and social feedback in the virtual collaborative environment.

Thesis Supervisor: Feniosky Peña-Mora
Title: Assistant Professor of Civil and Environmental Engineering

Acknowledgements

I would like to thank Professor Feniosky Peña-Mora and the entire DiSEL 1998-1999 team who made this great experience possible: Kiran Choudary, Gregorio Cruz, Caglan Kuyumcu, Gregoire Landel, Christian Manasseh, Sanjeev Vadhavkar, Jaime Solari, and Padmanabha Vedam. To the DiSEL '98 team, with whom I have gone through many enriching and fun times with this year, I wish you guys the best in everything.

I would also like to share my gratitude and love for all my friends who have made life fun, interesting, and worthwhile. I thank you all for everything throughout the years. Lastly, I would like to thank my parents and my sister, Ji Won, who have given me love, inspiration, and support throughout every moment in my life.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Communication between humans consists of more than the verbal component that we all associate with common human interaction. Human communication goes far beyond just verbal speech, for nonverbal component of communication are just as and, in some cases, plays even a more significant role in conveying ideas and emotions. Nonverbal elements are essential parts of full and effective form of communication and conveyance of idea from one individual to another. Nonverbal factors include such things as facial expressions, gestures, body movements, stance, and gaze. These factors are unconsciously and unintentionally present in the communication ritual of human beings, and full conveyance of a person's ideas or emotions cannot be realized and transferred to another in absence of these factors.

Human interaction and communication can occur in a number of ways through various media. Among the various forms, human interaction include face to face (where all forms of communication, verbal and nonverbal, is utilized for most effective social interaction), telephone (where only verbal speech is utilized), video conferencing (where both verbal and nonverbal elements can be conveyed), text chat (where essentially verbal form of communication is channeled via typing), and more recently three-dimensional virtual worlds (where avatars represent the involved users in a virtual world). Furthermore, human interaction occurs in any scenario, whether it be in pure social or entertainment purposes or in collaboration in a work-oriented setting. No matter in which scenario and method of interaction and collaboration is taking place, the need for

effective communication is universal. Without effective communication, social interaction capabilities are significantly diminished because of the weakness in the participants' abilities to convey, share, and perceive ideas and feelings of one another. As it will be discussed in the next chapter, such ability to express as well as to perceive ideas and emotions is crucial in social interaction and, consequently, in collaborative settings.

This thesis will emphasize the latter of the two scenarios in which human interactions occur – within the context of collaborative work environment. Specifically, this thesis will examine the role of gestures in personal expression in human social interaction as it pertains to collaborative engineering environment where users or the involved parties are interested in collaborating in a geographically distributed setting to achieve a certain goal in their work or learning process.

1.1 DiSEL and CAIRO

The motivation for this research stems from the development and the research that extends from the Collaborative Agent Interaction and synchROnization (CAIRO) software that has been originally developed by Karim Hussein, Sc.D. (Hussein, 1998) and worked on by the Distributed Software Engineering Laboratory (DiSEL) from 1997-1999 (DiSEL 1997, DiSEL 1998). The DiSEL team consists of members from both MIT and CICESE in Mexico. The members of the 1997-1998 DiSEL team are: Bob Yang, Felix Loera, Humberto Chavez, Simoneta Rodriguez, Lidia Gomez, Gregorio Cruz, Rene Navarro, Christine Su, Tim Wuu, Sergio Infante, Juan Contreras, Ruben Martinez, Charles Njendu, Diana Ruiz, Kareem Benjamin, Juan Garcilazo, and Marcela Rodriguez. The DiSEL 1998-1999 team consists of: Christian Manesseh, Gregorio Cruz, Gregoire Landel, Sanjeev Vadhavkar, Cagaln Kuyumcu, Joon Hor, Kiran Choudary, Padmanabha Vedam, Jaime Solari, Octavio Garcia, Alberto Garcia, Juan Contreras, Ricardo Acosta, and Rafael Llamas. Working as software engineering teams, the DiSEL groups worked to add features to CAIRO to enhance social interaction and casual contact.

CAIRO is a platform-independent collaborative software framework developed in Java that enable geographically and temporally distributed designers and teams to collaborate in real-time over the Internet and private computer networks. Distributed collaboration is achieved through the unique combination of features that creates a virtual environment in which social feedback and casual contact mechanisms simulate a real meeting and work setting.

CAIRO serves as the common base framework that allows the users to create their specified meeting environment by giving the users the connectivity tools directly needed in a meeting environment. The modular tools that can be accessed from the CAIRO platform are called "drivers". They include, but are not restricted to, social feedback mechanism, casual contact ability, text-based chat function, shared whiteboard for sketching, schedule/agenda monitoring tool, as well as 3D interface and affective computing abilities. In addition, plug-ins to the system allow for customization to fit virtually any type of collaborative work among geographically dispersed members. The CAIRO platform, as well as social interaction feature that it supports will be discussed more in detail in later sections.

In the context of such an architecture that supports collaboration among distributed members, the question of how to improve the effectiveness and "realness" of the collaboration process arises. Undoubtedly, one major component of collaboration is solidly founded on human social interaction that makes communication among people possible and drives the dynamics of the group setting. This research is mainly focused on this aspect of collaboration - social interaction. More specifically, I will look at the role of gestures in social interaction and how this can be incorporated into and represented in a social interaction-supporting system such as CAIRO so that a more natural and more effective collaboration process can take place in the virtual environment.

1.2 3D Virtual Environments

Virtual communities allow people to meet in a fictitious setting where they can socialize, work, and build worlds that are totally independent from our "reality" domain that we live in. Until recently, most of these networked virtual communities, such as MUD's (multi-user domains), have been text-based so that any form of communication that occurs among the users in this virtual environment was through a text dialog interface where the users were limited to a very one-faceted method of communication and interaction with each other. However, these text-based environments are now going graphical, displaying models of colorful locales and the people that inhibit them.

These graphical virtual environments are usually three-dimensional, where people and the objects are represented as 3D models. Thus, this 3D virtual world allows the user to delve into a completely different realm where they have the ability to meet and interact with other users in the world. When users connect to such a system, they choose a character that will become their graphical representation in the virtual 3D world, called an *avatar*, a word from Sanskrit meaning "incarnation" (Gesture and Narrative Language Group MIT Media Lab, 1999). Once inside, the users can explore the environment, often from a first person perspective, by controlling the movement of their avatar. The avatar of all the other users that are concurrently logged onto the system can be seen, and they can be approached to initiate conversation.

Such a 3D virtual world are currently used largely for entertainment and purely social purposes. However, a virtual environment can be equally effective in a collaborative work-oriented setting, where each participant or a team member can log on and, in effect, conduct a productive meeting. Such a 3D interface for the CAIRO system was proposed and developed in DiSEL '98-'99 (DiSEL 1998). It is in this context of collaborative engineering environment that social interaction and effective communication is discussed.

Like any form of human interaction in virtual environments, effective communication among the users is crucial and indispensable. As mentioned earlier, effective communication involves many factors that include the verbal as well as nonverbal aspects. In regard to the nonverbal factors associated with communication and social interaction among people, gestures undoubtedly convey and express much subtle as well as obvious meanings. With the 3D environment for CAIRO, natural extensions of social interaction and communication issues arise. The main focus of this research is in that area of gestures in effective communication in the context of social interaction, and how this may be represented in a 3D environment.

1.3 Overview

This chapter will serve as an introduction to the concept of social interaction and virtual environments where collaboration can take place. The idea of three-dimensional virtual environment for computer systems is presented and DiSEL as well as CAIRO is introduced. The remainder of the thesis is divided into five chapters, each providing information to further understand and to define the significance of gestures as part of social interaction, and how such an element can be incorporated into a expression tool within a three-dimensional virtual environment.

Chapter two describes the background and motivation for the research. The CAIRO system is described and the concepts and elements of social interaction among humans are explained in detail.

Chapter three takes an element of social interaction that was detailed in Chapter 3 and goes in depth with it. Specifically, the chapter explores the theories of gestures in communication and human interaction, and how it may be relevant to the representation of personal expression and social feedback in a collaborative setting.

Chapter four explains the context of this research in terms of the DiSEL 1998-1999 group and its project goals. It explains the nature of DiSEL and the specific work in the

development of a 3D interface done in the 1998-1999 academic year as it pertains to this research. It contains relevant sections of the group's requirements analysis, conducted primarily by the analyst, Gregoire Landel.

Chapter five then covers the feasibility of gesture expression in a virtual system. Some methods of gesture expression in computer systems is discusses, followed by an examination of what types of gestures are appropriately expressed in the case considered in this thesis.

Chapter six proceeds with the design of such a gesture expression prototype that can be implemented to represent animating gesture movement by the user. The requirements of such a tool is presented along with the technical design issues that are to be considered in such a prototype. VRML avatar standards, VRML execution model, animation techniques, and the Java-VRML interface is examined in detail.

Chapter seven serves as a conclusion and evaluation of representation of gestures as a means to enhance social interaction. The evaluation of the proposed design in terms of effectiveness, feasibility, and potential shortfalls is presented, along with a conclusion of gestures and social interaction in collaborative systems.

Chapter 2

Background and Motivation

As mentioned in Chapter 1, the motivation behind this research of gestures and its role in social interaction in collaborative virtual environments lies with CAIRO and its social interaction mechanism available to its users. CAIRO platform allows such interaction within the distributed collaborative environment, and the study of gestures to gain a more effective communication and social interaction in such environments is what is driving this research.

In this chapter, CAIRO is described in more detail and the concepts of social interaction in a collaborative environment is clarified and developed fully. Moreover, the significance and relevance of gestures in this framework will be outlined to give a sense of where it falls under in the broad picture.

2.1 Collaborative Agent Interaction and synchROnization (CAIRO)

As mentioned earlier, CAIRO is a system that allows effective collaboration among geographically distributed users by incorporating tools of communication and meeting process control. A unique aspect of CAIRO that most distance communication methods don't allow is social interaction in casual contact and planned sessions of the users.

2.1.1 Background

CAIRO is a platform-independent collaborative software framework that enable geographically and temporally distributed teams to collaborate in real-time over the Internet and private computer networks (Hussein, 1998). Distributed collaboration is achieved through the unique combination of features that creates a virtual collaboration environment in which social feedback and casual contact mechanisms simulate a real meeting and work setting.

Information exchanges in a shared work environment comprise of a variety of media. Typical exchanges between members of a group involve verbal and nonverbal forms of communication and social interaction, shared documents, and sketches or visual aids. Such interactions have occurred traditionally in a face-to-face, real time situations, as in a meeting, or asynchronously, in the form of memos/e-mail.

CAIRO was created and improved upon with this aim of recreating an effective real-time collaborative environment where human interactions that have traditionally occurred in co-located settings are achieved in a virtual, networked computer system. (Hussein, 1998) Specifically, CAIRO system provides an environment that allows geographically distributed individuals/ teams to collaborate over the Internet by simulating virtual meetings. It provides a natural, powerful, and responsive environment to the multi-modal communication needs of users across the Internet in real-time. The CAIRO collaboration system will enable users to function in an information immersion environment, able to retrieve, forward, and re-use information regardless of the temporal nature of its source or destination.

CAIRO provides an environment for structured information exchange across the Internet in real-time. Critical design elements of the system are synchronous and asynchronous multimedia communication channel, coordinated social interaction support, system modularity and adaptability with a variety of media and tools, robust and reliable means of communication, and finally, a multi-user interface for collaboration.

Since collaborative work may include sections that are real-time, as well those that are asynchronous, CAIRO supports capture and replay of real-time discussions. In this way, group members unable to participate in portions of an ongoing collaboration will not be deprived of information. This is further supplemented by automated indexing of captured material, incorporation of asynchronously generated materials into the collaboration stream, and object annotation support for the workgroup.

2.1.2 CAIRO Design

Real time interaction is inherently taxing on both system and communication resources. Furthermore, the multimedia nature of human interaction necessitates a synchronization mechanism between the media channels to preserve the time dependence of the initial user input. The CAIRO system is based on the Internet and its underlying TCP/IP protocols. CAIRO assumes that the underlying network is non-deterministic and methods have been developed to accommodate this inadequacy, which are based on real time scheduling techniques. This basically means that CAIRO assumes that the packet delay times are not pre-specified and may vary.

In order to satisfy the various requirements and needs of a real time interaction system via the Internet as mentioned above, a model of the distributed negotiation meeting process has been devised. This model is composed of four critical components (Hussein, 1998):

1. *Co-location*: a physical meeting room in which the participant can meet.
2. *Cooperation*: a common language and shared understanding of materials to be presented in the meeting
3. *Coordination*: an agenda and an individual or set of individuals that ensures the agenda is maintained and the group is focused on resolving the issues outlined in the agenda.
4. *Documentation*: group memory which is comprised of each individual's memory and notes as well as the formally defined group memory incorporated in the minutes of the meeting.

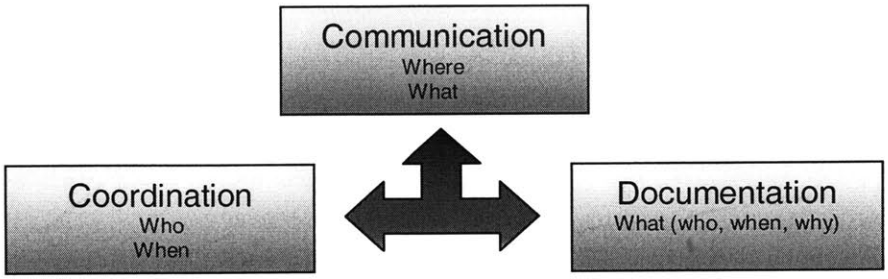


Figure 2-1: Collaboration Mechanism of CAIRO

The collaboration mechanism in CAIRO can be illustrated in Figure 2-1. This model maps the necessary physical meeting elements into a general requirement list. The following provides a description of the layers of a computer-based system required for an effective collaborative environment:

Co-location involves dealing with the network infrastructure to provide seamless communication among distributed clients in a conference. This layer should provide naming services to identify client locations as well as interaction with the network protocols to transmit data across the network between the clients.

Co-operation involves the sharing of information among clients in a team. Due to differences in software and capabilities of the various clients, translations need to be performed in order to provide a coherent view of the data among the clients.

Co-ordination involves control of the work flow and communication process. This allows for efficient control mechanisms to coordinate group effort. The coordination layer acts as a "virtual manager" of the conferring clients.

Documentation involves the capture and storage of conversation elements exchanged during a meeting. The documentation process provides a mechanism for the retention of group memory.

2.1.3 Features of CAIRO

CAIRO is the common base framework that allows the users to create their specified meeting environment by connecting the working tools directly into the meeting environment. The modular tools that can be accessed from the CAIRO platform are called "drivers" which include, but are not restricted to, social feedback mechanism, casual contact ability, text-based chat function, shared whiteboard for sketching, and schedule/agenda monitoring tool.

As it stands today, CAIRO exists as a collaboration enabling platform by which the above mentioned drivers, i.e., basic text chat, shared whiteboard, agenda manager, and personal expression tool, work together to supports meeting and collaboration work on the Internet. Multiple meetings can co-exist and the different styles of meetings, such as lecture, conference, round table, are supported and are readily changed. DiSEL 1997-1998 has added a casual contact ability and a personal expression component to CAIRO by the addition of the facial expression capability, whereby a user can express his mood by changing the facial expression that is displayed in the meeting window in CAIRO.

In addition to the already existing features that CAIRO provides, there are further work that is proposed to be done. Parts of this future work was conducted by DiSEL '98-'99, as detailed in Chapter 4. The additional work that would enable a next generation of CAIRO would include the following elements:

Multi User Collaboration Interface:

- CAIRO would provide an environment that enables geographically distributed individuals/ teams to collaborate over the Internet, by simulating a virtual meeting.

- A centralized collaboration manager incorporates the CAIRO client interfaces and maintains a list of forums, meeting rooms, participants. It also provides security by restricting access to forums/meeting rooms.
- The CAIRO collaboration manager would provide multiple meeting environments namely, free style, chaired meeting, lecture room meeting each with its own underlying sets of collaboration protocols. The meeting room environments would have both 2-D and 3-D interfaces.
- The collaboration interfaces in CAIRO are based on metaphors derived from extensive research in the field of collaboration among distributed teams, focusing on issues like meeting entry and exit, floor state, member state and addressability, focus of attention, degree of engagement.

Media Drivers:

CAIRO would provide the following media drivers to enable effective collaboration

- A text driver with a language translator that will allow exchange of short messages among participants. The built-in language translator allows people with different cultural backgrounds to collaborate effectively.
- An audio and video driver enables users to exchange communication using speech and/or video in a synchronous manner.
- ***A social interaction (awareness driver)*** driver allows meeting participants to get feedback on the emotional responses of their distributed counterparts.
- ***3D Environment*** that allows the user to interact and collaborate in a virtual environment where social interaction is enhanced through a greater control and freedom of personal expression and social feedback.

These are some of the visions that future works would focus on. As denoted by the bold-face, DiSEL '98-'99 has worked to improve on the Awareness driver and the 3D environment interface for the user. These two components are directly related to the

enhancement of social interaction in the CAIRO environment and will be further examined in Chapter 4.

2.2 Social Interaction

The following sections outlines the basic concepts and elements of social interaction as defined in such a collaborative system as CAIRO. This discussion is taken from DiSEL '97-'98 requirements analysis done by Simoneta Rodriguez and Humberto Chavez (Rodriguez, 1998). While DiSEL '97-'98 concentrated on a distributed collaborative learning and I am looking at distributed collaborative work environments, the basic ideas of collaboration and social interaction are quite similar.

2.2.1 Distributed Collaboration

Distributed project collaboration is a social activity conducted by those who happen to be geographically dispersed. The effectiveness of such social activity depends upon a computer network, communication technology and tools such as CAIRO, that enable human social interaction across the distances separating the participants in a way that resembles face-to-face interactions as successfully as possible. Components of the virtual work environment created by such technologies usually consist of elements such as conference rooms and project spaces that simulate real-life collaborative settings where social interactions occur. Within these settings, the social tasks that are involved require teams of people to perform tasks with some level of common purpose, varying but persistent leadership roles, and consistent expectations for required collaboration.

As technology enables that world to shrink by facilitating connectivity among people, collaborative work and learning performed by groups of people who are distributed in space and time face certain problems with regard to social interaction. It is impossible to disregard the importance of certain elements of social interaction among distributed teams. Moreover, such elements of effective communication and interaction that a co-

located team may take for granted are rather very difficult to provide in distributed collaborative environments.

While current techniques and technologies for conducting distributed project collaboration readily permit basic forms of human social interaction such as speech and writing, they have significant deficiencies that include severe restrictions on possible forms and the range of social interaction. These restrictions inevitably limit the effectiveness, applicability, and usefulness of current distance collaboration techniques.

2.2.2 Implication of Current Limitations and Problems

There are many significant deficiencies in conducting distributed project collaboration in terms of communication and social interaction. These limitations are mainly caused by issues of technology and techniques that are implemented which disregard certain elements of social interaction. Specifically, these include the following:

- Limitations upon the forms and the range of ***social interaction*** possible between participants. "Complete" social interaction is necessary within collaborating groups because it allows:

  - "lasting bonds" among the participants. Lasting bonds seem to be essential for effective learning and collaborative environments.
  - participants to "interpret the thinking" of other participants so that complete and effective communication of ideas and emotions is possible. Groups appear to work significantly better when group participants can predict some of the future behaviors of other participants.

- Removal of the nonverbal immediate-feedback aspects of co-located human interaction. That is, ***social feedback*** while speaking, lecturing, or attempting to interact is severely limited or non-existent without some mechanism by which these nonverbal factors are transferred and acknowledged by the other parties involved.

Social feedback, as a component of social interaction, is essential for appropriate delivery of content, as well as for assurance that intended communications have been established. Social feedback includes:

- Facial expressions
- Hand movements
- Body movements (body language)
- Orientation of the head
- Focal point of the eyes – gaze
- Stance

Note that hand movements, body movements, and the orientation of the head are inclusive of human gestures, or gesticulation, which this research will focus on.

- Non-existent opportunities for unplanned encounters or ***casual contact*** in informal settings unrelated to the structured sessions of collaboration. Casual contacts among participants are significant contributors to effective collaborative environments because they allow observation of unrehearsed behaviors. This facilitates learning about the other participants that cannot occur without such unplanned encounters. It is clear that social feedback mechanisms are crucial in this regard as well.

Social interaction is a technical term used in the field of environmental psychology, referring distinctly to the interaction of one or more human beings with other human beings in a specific setting. The term is used here in this technical sense, to distinguish from other kinds of interaction, such as those between human beings and machines.

Social feedback as used in this context refers to those components of social interaction that permit a speaker to determine the emotional state of the listener. This allows a distinction between feedback information perceived by humans about/from other humans, and feedback information perceived by humans about/from non-humans (e.g.

machine feedback mechanisms). In other words, machines can provide feedback, which may or may not provide humans with social feedback about other humans.

Casual interaction is used here to refer to unplanned, spontaneous social interaction occurring outside of planned sessions, and meetings.

2.2.3 Social Interaction

Social interaction is comprised of two elements: 1) Personal expression, and 2) social feedback. A third phenomenon, casual contact, is intimately and inherently related to these components. It is evident that personal expression and social feedback are directly related to each other, because portions of one person's expressions, such as speech and behavior, provide the signals interpreted as feedback by another. This social interaction can be accomplished in two scenarios of human interaction: during planned encounters among the participants and during unplanned encounters. Examples of planned encounters include scheduled meetings, and work sessions. Examples of unplanned encounters, called casual contact, are chance meetings among participants on a hallway. Both forms are composed of the two tightly coupled elements of social feedback and personal expression, as shown in Figure 2-2.

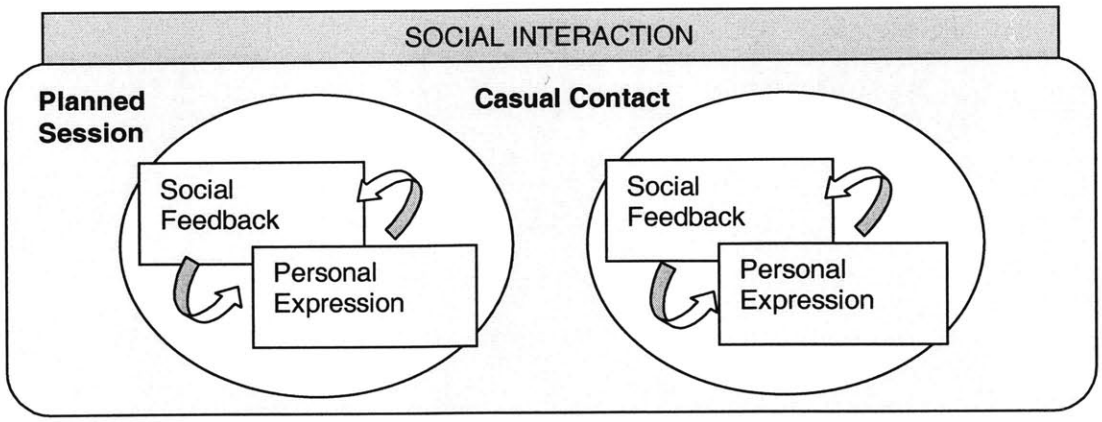


Figure 2-2: Social Interaction (social feedback, personal expression, and casual contact)

It is important to note the close relationship between social feedback and personal expression in both planned and casual forms of social interaction. They are mutually dependent on each other, as one would lose meaning without the existence of the other. Social feedback depends on personal expressions of the participants in order for its mechanism of communication and interpretation to have meaning. Conversely, personal expression is meaningless without the social feedback mechanism by which the expression displayed by the participant is transferred to and interpreted by the other.

Human beings express themselves both verbally and non-verbally, through activity and inactivity, voluntarily and involuntarily. Some portions of the total range of a person's expressive behaviors provide the cues that others use to interpret the person's ideas and their state of emotion. This is termed social feedback. In a collaborative virtual environment, at least some representation of personal expression must be transmitted among the participants in order to achieve this social feedback, and thus social interaction.

2.2.4 Personal Expression

To analyze the concept of personal expression, a representation called a "black-box human" is introduced. Each participant or user in the collaborative environment is represented as a black-box human that provides only output signals as far as the observer is concerned. These output signals that are given by such a black-box model in a given human social setting include these six readily identified signals:

- Movement or lack of movement and focal point of the eyes
- Presence or absence of verbal expression, i.e. speech
- Movement or lack of movement of facial muscles – facial expression
- Position, orientation, and movement of the head
- Position, orientation, and movement of the body – body-language
- Position, orientation, and movement of the hands

It is important to note that this list of six signals emitted by the black-box human is not exhaustive. Additional signals can be identified easily, and when found, they should be added to the list accordingly. Figure 2-3 illustrates how the six signals transmit information from a black-box human to someone perceiving these signals. As explained below, the signals from the eyes, speech, face, head, body, and hands get transmitted to another person to relay information about the presence, behavior, and attention state of the transmitting human, who in the Figure 2-3 is the black-box human. The information that gets relayed and expressed through the transmission of the signals (presence, behavior, and attention state) is termed here as the three requirements of transmission.

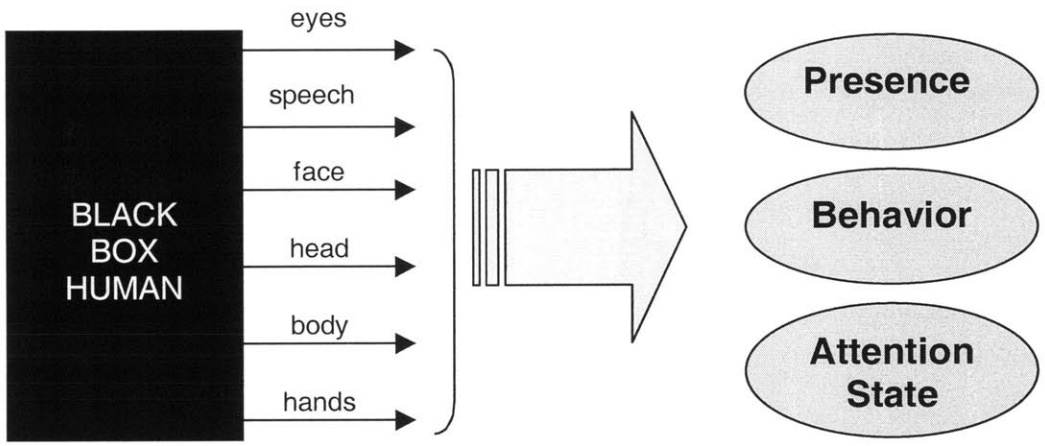


Figure 2-3: Personal Expression Components

The personal expression construct outlined above can be simplified as follows:

- The signals that must be transmitted and translated by the collaborative system are body, head, face, hand, and the resulting combinations specified below. Speech signal is readily represented via text chat, or microphone.

- The *body signal* is essential for transmission of all three requirements. Body signals that must be conveyed in the collaborative environment include:
  1. Each participant's bodily presence during planned sessions
  2. Intermittent, or non-continuous, bodily presence outside of planned sessions, i.e. casual contact
  3. Sitting
  4. Standing
  5. Shoulders up
  6. Shoulders down
  7. Walking away

- The *head signal* is also important in conveying of all three requirements. Head signals that must be conveyed include:
  1. Orientation (which way it's facing)
  2. Upright
  3. Tilted
  4. Nod
  5. Shake

- The *hand signal* is used in the transmission of the requirements mainly in use wholly or as a part of a gesture. Hand signals, two for each participant, modulated individually, that must be conveyed include:
  1. Raised
  2. Point
  3. Fist
  4. Count
  5. Held up to head
  6. Palm up
  7. Palm down
  8. Wave

- The *face signal* is an integral element for transmission and perception of the three requirements. Face signals that must be conveyed include:
  1. Smile
  2. Laugh
  3. Frown
  4. Serious
  5. Yawn

- Combination signals that is may be used to convey information about the person's emotional or affective state include:
  Excuse me signal: one hand raised to indicate a wish to speak
  Puzzled signal: shoulders up, head tilted, both hands palm up
  Demand signal: face serious, hand fist
  Bored signal: head tilted, face yawn, hand held up to head
  Leaving now signal: body walking away, hand wave
  Negative attention signal: no voluntary signal during specified period of time: head tilted, face away

Taking a cue from the signal labeled by "body", commonly called "body language", a little reflection leads to the conclusion that each of these signals is a language used by human beings to communicate in social settings. Thus, the six identified signals are transmitted in six "native languages". These six signals, emitted by the black-box human in their six respective native languages, transmit the components of social feedback identified as presence, behaviors, and attention state.

Further analysis of personal expression calls for the following:

- Identify key forms or instances of each signal in its native language.
- Of these signal forms or instances, and combinations of them, identify which are essential for transmitting presence, behaviors, and attention state.

- Identify how additional or alternative languages, suitable for transmission through computer systems, will be developed for some or all of the native black-box human signal languages. It is crucial to note that new languages for representing these signals, must be tested for their ability to be interpreted by the intended market of system participants, in the social feedback stage.
- Identify how to translate the essential signals or combination of signals into the new languages.

To this aim of analyzing personal expression as it relates to social feedback in the social interaction framework, gestures will be studied in detail. Gestures comprise of elements of body signals, head signals, and hand signals that have been discussed above. Further exploration of gestures in personal expression in virtual environments will be discussed in more depth in the following chapters.

2.2.5 Social Feedback

With personal expression represented as black-box humans emitting signals in native signal languages, social feedback may be represented as an interested participant's receipt and interpretation of these signals. While each person in a social setting continuously emits a variety of signals, a participant received some of these signals at various times. These reception times are more less within the participant's control, especially in a group setting. Such phenomenon can occur in two situations, on an individual level basis, and in a group level basis. The transmission and perception of signals in a one-on-one situation is shown in Figure 2-4, where there are only two involved parties in this mechanism.

However, personal expression and social feedback can also occur in a group level, where an individual is interacting with more that just one other person. Therefore, in a group setting, the participant must interpret the same signals at two (minimum) levels, individual level interpretation and group level interpretation. This is illustrated in Figure 2-5.

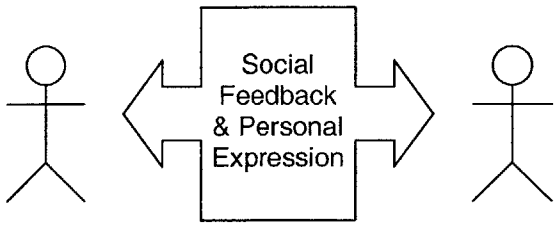


Figure 2-4: Individual Level Interpretation

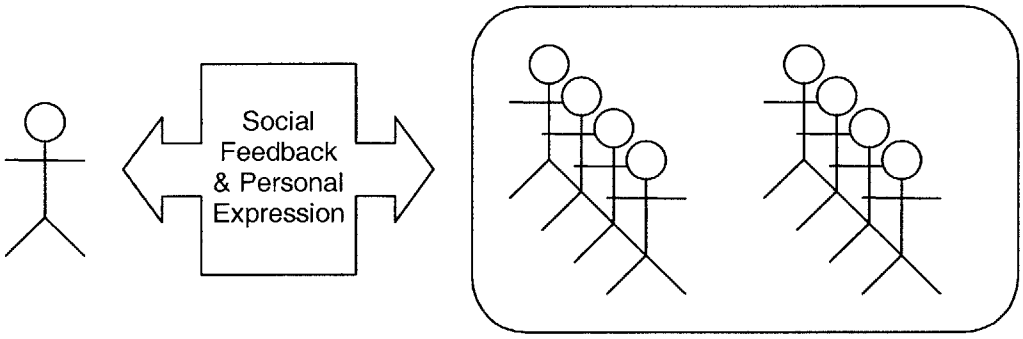


Figure 2-5: Group Level Interpretation

The social feedback construct can be summarized as follows:

- The previously defined signals of the body, head, hand, face, and their combinations, along with speech, can be assumed to be sufficient for an involved participant in the collaborative virtual environment to perceive and interpret presence, behaviors, and attention states.
- Emitted signals in the framework of the virtual environment are assumed to be the result of voluntary actions by participants.
- Presence should be interpreted when the body signal operates.
- Behaviors should be interpreted through the operation of any voluntary signal.
- Attention state for each participant should be interpreted as follows:
  - Positive Attention State: voluntary signals occurring.
  - Negative Attention State: no voluntary signal within a specified period of time.

In certain cases, the personal signals that are transmitted to the other involved party/parties are not the only factors that play a role in the communication channel. Other circumstantial information, such as the context of the environment (e.g., in a formal meeting room) and other possible external information (e.g., dress), can also play a role in determining the presence, behavior, and attention state of the involved parties of social feedback in an social interaction setting. For example, someone dressed in a formal business suit sitting at a meeting table in an office gives off a more formal, serious, and "down-to-business" state of mood, as compared to someone dressed in jeans sitting at a couch with a soda in his hand. Such situation is illustrated in Figure 2-6 below.

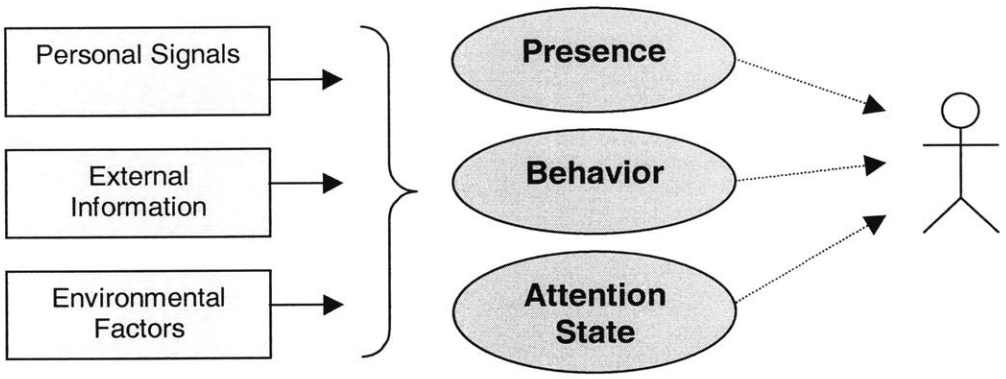


Figure 2-6: Social feedback Components

2.2.6 Casual Contact

With the preceding representations of personal expression and social feedback, casual contact may be represented as a "when" issue rather than a "what" issue. Since the components of social interaction are personal expression and social feedback, the fundamental criterion for the existence of casual contact is determined by when social interaction takes place: during planned sessions, or during unplanned events or encounters. Various forms of black-box human signals can and will be emitted during

both planned and unplanned events, and must be interpreted by a participant to obtain social feedback.

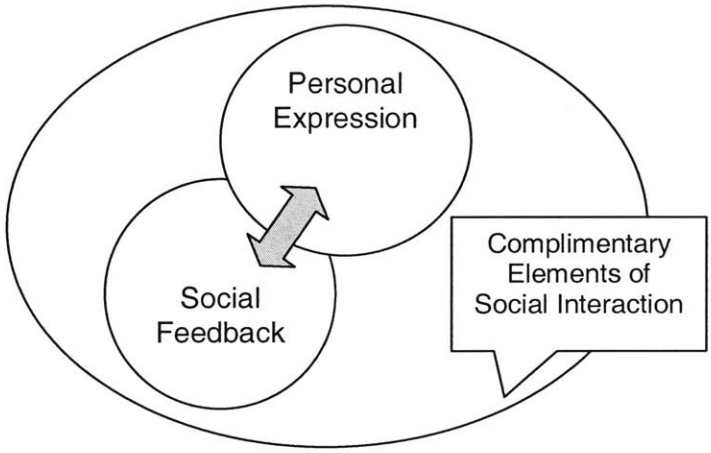


Figure 2-7: Components of Social Interaction

To move beyond the fundamental criterion of casual contact, occurrence of casual contact in "real life" must be examined. The "real life" scenario is when and where the participants would be in physical co-location with each other and would exchange forms of social interaction and communication in a real time basis. This will be termed "real time contiguous space" (RTCS). Some of the features of RTCS that are notable and are crucial aspects of effective social interaction include:

- RTCS casual contact events occur essentially randomly. This follows from the fact that they are unplanned. Patterns in RTCS casual contact do appear, due to factors such as personal schedules, physical proximity of living space or office quarters, preferences for times or places to eat, and so on. Individuals find that they "run into" certain other individuals more often than others, in certain places and/or at certain times. Yet these patterns do not guarantee contact, for there is a large element of chance that is involved. This random quality of RTCS casual contact is a crucial

component to preserve in a computerized social interaction environment that provides opportunities for casual contact.

- While RTCS casual contact events occur randomly, individuals are able to exercise various levels of control over them. For example, an individual who knows that an encounter with Professor Peña-Mora is more likely to occur on the stretch of hallway near his office, may choose to either a) avoid that hallway, or b) to use it more frequently, with the choice based on desire for casual contact with the professor. Professor Peña-Mora, on the other hand, like all human beings, routinely uses a variety of personal expression signals to indicate his willingness to engage in social interaction at any given moment. He may, for example, state "See me later, I have a meeting right now." Or he may walk very fast with head down, which most people would interpret as "Do not disturb". On the other hand, if one encountered the professor in a relaxed pose, apparently lounging in the hall outside of his office, most people would assume that he is open to casual contact at that moment.
- RTCS causal contact events may be viewed as primarily related to spaces, activities, and broadcasts. While all RTCS events occur in the same space (by definition), some are more directly related to spatial and locational issues, for example, the hallway contact mentioned above. While all RTCS events occur at the same time by definition, some are more directly related to activities rather than spaces, for example, running into someone when faxing, at any of the several fax machines in the vicinity. Another example of activity-related causal contact is running into someone when eating, at any of several eating locations that one frequents. Broadcast casual contact may be viewed as the result of broadcasting one's availability for casual contact. This is tied to expressing the level of desire for casual contact, as demonstrated in the hallway example above. Broadcasting one's availability for casual contact welcomes interaction by transmitting signals that attract others to you. For example, if one wishes to broadcast his availability to interact, he/she may look around for any familiar face, put on a rather "friendly" smile, and address a very relaxed casual stance to let others know that he is available to socialize and interact.
- Issues involving definitions about what is private and what is public must be considered for casual contact provided as part of a computerized social interaction

environment. The distinction between public and private space must be made when considering social interactions and collaboration-based work. For example, when someone is in his/her own room, he/she is in his/her private space and thus casual contact ability from random individuals who don't have access to his private space is significantly reduced. On the other hand, if that person were to be in a hallway, he/she is now in a public space where casual contact is more likely and is more welcomed. Translating this concept to a computer environment, when someone is working from a computer hard drive, the user is effectively in a private realm which most people won't be able to access. Thus, this activity will be similar to being in a private office with limited outside interactions. Conversely, the user can also choose to put him/herself in public space by working in the public network domain. As he/she ventures onto the network space, he/she is putting him/herself in a public realm where he/she can be identified as online, and is available or accessible for contact by others online. This activity mimics the public space where interactions are not limited. However, in this case, the availability of interaction will depend on the availability of that person in the public space or online. If that person returns to work on the hard drive without any activity on the network, that person returns to his/her private space where interactions are once again limited. In this way, public and private space metaphor is utilized to closely mimic the face-to-face human interactions that occur everyday. As exemplified here, public and private issues are involved in the RTCS case, and it is noted that some portion of the legal system in most countries revolves around this issue. The ability to exercise personal control over casual contact may be viewed as essential to avoiding problems in this area.

2.2.7 Social Interaction: Further Representation

After the preceding analyses and representation, examining social interaction in yet more detail at this point is instructive. Several conclusions can be drawn, and debated.

- The components of social interaction are personal expression and social feedback.

- The languages of emitted signals must be shared between black-box humans and interested parties, for social feedback to occur.
- As a result, these two components are tightly integrated – they do not occur in isolation from one another.
- For a particular integrated system that builds a collaborative environment for a number of black-box humans, the shared signal language used by the participants and thus the system's ability to support such social interaction holds across space and time.
- Culture, whether shared or differing between participants, will affect both personal expression and social feedback, and thus social interaction systems.

2.3 Summary

Collaborative work effort and such an environment that supports group collaboration is generally considered by many to be feasible exclusively for co-located teams who can have face-to-face contact on a regular basis. However, recent advancements in technology has increased the feasibility of collaborative work possible for distributed teams. Many would consider options such as video conferencing to be a successful approach to distributed collaboration. However, if one were to examine the concepts and requirements of collaboration and the social interaction elements that are prerequisites, using methods such as video conferencing as the only channel of communication and interaction is greatly limited and flawed.

Collaboration support architecture that the CAIRO system provides is significantly eliminating such limitations in social interaction and successful collaboration efforts, such as one presented by an engineering design project involving team members from distributed locations. In particular, this thesis concentrates on the social interaction aspects of collaboration in this distributed environment through networked computer system. We have seen that social interaction is composed of personal expression and social feedback which are intricately related. It is clear that these two components are crucial for establishing solid lines of communication between the involved parties, which

in return form a lasting bond between them that is important in any human interaction. Through personal expression in forms of facial expressions, gestures, speech tone, ideas and feelings are transmitted, and through the mechanism of social feedback, these signals are perceived and interpreted resulting in a dynamic social interaction.

Having established the concepts of social interaction and its components, it is now possible to examine a particular element of social interaction that is used in face-to-face interactions and should be included in a virtual representation of such environment. To this aim, the following chapter will delve into depth of one of the aspects of personal expression that is commonly used in social interaction: gestures.

Chapter 3

Theories of Gestures in Communication and Social Interaction

The interest that linguists have been showing in how language is used in interaction has led to a realization that, from a functional point of view, spoken utterances often only work because they are embedded in contexts of other forms of behavior, including gesture. Psychology has also put higher mental processes in greater light, and gestures, as a form of symbolic expression, becomes a critical element in communication, and consequently in social interaction (Kendon, 1983).

In a typical social interaction between people, communication is accomplished through various channels that are both verbal and non-verbal. Exploring the non-verbal aspects of communication in social interactions, we find many elements that encompass a person's repertoire during the interaction. Elements such as facial expressions, eye gaze, and gestures are used to express emotion, to convey meaning, to help regulate the flow of conversation, to signal the search for feedback during the interaction, to look for information, or to influence another person's behavior.

In observing gestures specifically, it is easy to notice that people spontaneously use forms of gesture, such as hand gesture, while they speak. Such gestures support and expand on information conveyed by words. Furthermore, other types of gestures that are not used concurrently with speech are used in place of certain verbal speech to convey meaning to another person.

In this chapter, the psychological and linguistics theories on communication and the role and use of nonverbal features such as gesturing in social interaction and communication are explored. I will define gestures and consequently outline the various forms and roles of gestures, its significance in communication, and its relevance to social interaction between people in a collaborative environment. This section will build a strong theoretical foundation on the definition, relevance and importance of gesturing.

3.1 Defining Gesture

Before considering the roles and significance of gestures, the term itself must be defined. A common definition of the word "gesture" today, as given in the Oxford English Dictionary for example, is that it is a movement of the body, or any part of it, that is considered as expressive of thought or feeling. This, however, is a broad definition. At first sight, this definition seems to include practically everything that a person might do. However, a brief consideration of how the word is commonly used shows that the word gesture refers only to certain kinds of bodily movements that are considered expressive of thought or feeling.

As commonly understood, gesture refers to such actions as waving goodbye, giving the thumbs up sign, or shrugging the shoulders. It includes pointing and pantomimes that people sometimes engage in when they are too far away from one another to talk or in situations where talking would interfere. It includes the head waggings and arm wavings of vigorous talk, as well as the movements a person may improvise to convey meaning of something for which words seem inadequate. However, there are other kinds of action which, though expressive, seem less appropriately called "gesture". For example, one would not say that a person who was weeping is engaged in gesturing, or if one did, this would probably imply that the weeping was "put on" as a show, and that it was not wholly genuine as an expression of emotion (Kendon, 1986). Furthermore, the term gesture is not usually applied to the movements that people make when they are nervous, such as self-groomings, clothing adjustments, and the repetitive manipulations of rings or necklaces or other personal adornments. In ordinary interaction, such movements are

normally disregarded, or they are treated as habitual or involuntary and, although they are often revealing, and they may read by others as indications of a person's mood or feelings, they cannot be considered as gestures as a rule.

It can be also said that practical actions are not normally considered gestures even when such actions play a part in social interaction (Kendon, 1986). For example, when people have conversations they may also engage in such activities as smoking, drinking, or eating. The actions required for such activities may sometimes be used as devices to regulate the social interaction (Jarvella, 1982). People who meet for talk over coffee and a cigarette may vary the rate at which they drink up their coffee or smoke their cigarette as a way of regulating the amount of time to be spent in conversation. Lighting a cigarette or re-lighting a pope can often be elaborated as a way of "buying time", as when a person needs to think a little before he replies. Yet, despite the communicative significance such activity undoubtedly may have, it s not typically treated as intended to communicate anything. To spend time getting one's cigarette to light is to take time out of a conversation, and not to engage in a conversational moves or turns of which the conversation is composed.

3.2 Types of Gestures

Most researchers in the field of nonverbal communication, such as Kendon (1993), McNeill (1992), Efron (1941), Eckman & Friesen (1969), and Rime & Schiaratura (1991), have offered classifications, suggesting various types of gestures. Without surprise, there is much variation in terminology and techniques used to categorize certain types of gestures. However, a review of these classifications reveals that in general, there is a fundamental agreement among the various classification theories put forth by various researchers in this field. The main fundamental agreement is that all divide gestures into two broad categories of those that are performed independently of speech and those that occur in conjunction with speech (Kendon, 1986). Thus, most draw a distinction between speech-associated gesturing that somehow provides a direct representation of some aspect of the content of what is being said, and gesturing that appears to have a more abstract

sort of relationship. For example, Efron (1972) distinguishes those speech-related gestures that present a sort of picture of some aspect of the content as "physiographic". Moreover, he terms "ideographic" those speech-related gestures which he says are "logical" in their meaning and which portray not so much the content of the talk as the course of the ideational process itself. More recently, Freedman (1972) distinguishes "representational gestures" from non-representational or "speech primacy gestures", and McNeil ad Levy (1982) distinguish "iconic" gestures, "metaphoric" gestures and gestures which seem to be related only to the rhythmic structure of the speech, which he terms "beats". Gestures of this sort have also been recognized by Efron (1972) and by Ekman and Friesen (1969/1972) under the term "batons". I will discuss these classification schemes more in detail in the following sections.

The pioneering work in this field was done by Efron (1941) with his observations of the conversational behaviors of Jewish and Italian immigrants in New York City. The most important feature that Efron brought to the categorization process was the orientation toward the referent of the gesture. This distinction that he introduced has been retained in much of the following work by other researchers after him. It assumed that although some gestures clearly had relation to some external referent (object or event around the speaker), others have their referent within the speaking person's ideational process itself.

3.2.1 Gestures Referring to Ideational Processes

Gestures that refer to the ideational process closely follow the natural flow and contour of speech by marking the speaker's logical pauses, stresses, and voice intonations. According to Efron (1941/1972), there are two major subclasses in this category: speech-marking hand movements and ideographs.

3.2.1.1 Speech-Marking Hand Movements

Efron denotes these types of gestures as *batonlike*: their referents are in the speech, and they either (1) stress some element of the speech, (2) introduce some new element into the talk, or (3) divide the spoken sentence into chunks according to underlying reasoning. All the other existing classifications mention categories of gestures representing some variant of this, Freeman's (1972) *punctuating movements*, Ekman and Friesen's (1969,1972) *batons*, McNeill's (1987) *beats*, and McNeill and Levy's (1985) *batonic movements* all similarly fall under this category. Taken together, these various items represent the general class of speech-marking movements. In this type of gesture, the hand moves with the rhythmic pulsation of speech. The typical baton or beat is a simple flick of the hand or fingers up and down, or back and forth. The movement is short and quick and the space may be the periphery of the gesture space (e.g., the lap and an armrest of a chair). The critical things that distinguishes the beat from other types of gesture is that it has just two movement phases: in/out, or up/down (McNeill, 1992). These types of gestures mark the rhythm of the speech and can be seen by most as trivial, random movements of the hand while one is speaking.

3.2.1.2 Ideographs

A second class of gestures that Efron put in the category of those referring to ideational processes is termed *ideographs*. These gestures trace the path of the speaker's thought and thus, their referents are in the speaker's ideational processes, such as an abstract idea as opposed to a concrete object or event. Although obviously present in everyday life situations, ideographs have less often been isolated than speech markers by later classifications. Nevertheless, Ekman and Friesen (1969,1972) distinctly mention ideographs under this label and define them as movements sketching the path or direction of thought. Similarly, McNeill and Levy (1982) recognizes the existence of ideographs and includes them in what is termed metaphoric gestures, which depict some abstract meaning occurring in the speech. For example, when someone is speaking about the logical relation of reciprocity, they may represent this by a hand movement from left to right and from right to left. Another example is when a speaker may say "It was a

Sylvester and Tweety cartoon." with his hands rising up and offering the listener an "object". A particular cartoon event is concrete, but the speaker here is not referring to a particular event, but rather he is referring to the genre of the cartoon. This concept is abstract, yet he makes it concrete in the form of an image of a bounded object supported in the hands and presented to the listener (McNeill, 1992). McNeill categorizes this type as a metaphoric because of the metaphor that a concept of a genre of a certain kind is presented as a bounded, supportable, spatially localizable physical object. Thus with ideographs, gestures can have some sort of depictive function that can extend further beyond the speaker's ideational process to the actual objects in his speech.

3.2.2 Gestures Referring to the Object of the Speech: Depictive Type

Another gesture type that can be distinguished from the ones that refer to the ideational process is categorized as depictive gestures, meaning that they bear a close relationship to the semantic contents of speech. As the name suggests, these gestures act to depict something, whether it be object or event. Efron (1941) categorizes two types of gestures in this class: *physiographic and pantomimic.*

3.2.2.1 Physiographic Gestures

Physiographic gestures are those that parallel speech and present some kind of "figural representation" of the object being spoken about. The referents of these gestures are in the content of the speech. Gestures of this kind is present in every kind of classification scheme, represented by Freedman's (1972) *motor primacy representational movements*, McNeill and Levy's (1982) *iconic gestures*, and Ekman and Friesen's (1969,1972) *illustrators*, which is a broad class consisting of eight different gesture types.

Basically, three types of physiographic gestures seem to be distinguished with respect to the aspect of the referent (Rime & Schiaratura, 1991). First, the gesture may represent the shape of the referential object, for example, the upward spiraling movement of the fingers may represent a spiral staircase. Efron called this type of gesture an *iconograph*

while Ekman and Friesen called it a *pictograph*. Second, the gesture may represent some kind of spatial relationship with the referent. For example, two open hands placed palm to palm may be used in referring to the restaurant located between the bank and the department store. Gestures of this kind, called *spatiographic*, is distinct from ideographs in which the depicted relationship is always abstract. Third, the gesture may describe some action of the object, as in ascending movement that accompanies the verbal expression "growing up" or as in the descending motion that would depict some notion of "falling down". These action-depictive gestures were labeled as *kinetographs* by Efron and by Ekman and Friesen (1972).

The three types of gestures that fall under this category would be summarized as *pictographic*, *spatiographic*, and *kinetographic*. McNeill and Levy (1982) introduces the general label of *iconographic* or *iconic gestures*, to categorize this class. This term can be used in place of Efron's *physiographs*, which portrays a less encompassing class.

3.2.2.2 Pantomimic Gestures

These gestures also parallel speech like physiographic or iconic gestures, but serve, in addition, to illustrate the function or manipulation of some object. The referents of these gestures are in the objects which are spoken about. These referents are sometimes indirect, as when one speaks about "cutting" and pantomimes the movement of a hand holding a knife.

Iconic and pantomimic gestures were distinguished by NcNeill and Levy (1982) using a criterion of the level of differentiation between a gesture and the referred object. At the lower level of this differentiation, the hands of the gesturing person are playing their own role, illustrating their function of manipulating objects. In this case, the referred object of the speech is some acting person, and the described actions are imitated by the speaker's hands. For example, the speaker may shape an imaginary box with his hands to illustrate the phrase, "he grasped the box". This type of hand gesture is distinguished from those in which the hands play a more abstract role in the representational process, for example, in representing "he swallowed it" with the

movement of the left hand "swallowing" the right fist. Here, there is a marked differentiation between the referent and the symbol. Although this second case gives a clear example of a spatiographic gesture, which is a subclass of iconic gestures, the first example with a lower symbol-referent differentiation introduces *pantomimic gestures* which consist of true mimicking actions. Although pantomimes may be restricted to the activity of the hands, they are often likely to involve the entire body so that the speaker becomes an actor. In their strongest form, pantomimes do not need to be accompanied by speech and thus become autonomous, where this is the lowest level of symbol-referent differentiation (Rime & Schiaratura, 1991).

3.2.3 Gestures Referring to the Object of the Speech: Evocative Type

The last two types of gestures that Efron classified, *deictic* and *symbolic* or *emblematic*, may be grouped in a category referring to their evocative effect. In this class, gestures no longer depict the referent, as iconic and pantomimic gestures did, but rather simply evoke this referent by some action likely to elicit its presence in the common mental space created by the speaker and the listener.

3.2.3.1 Deictic Gestures

Deictic gestures are pointing movements, which are typically performed with the index finger, although any extensible object or body part can be used. These gestures are directed toward some visually or symbolically present object, which may be a place or an event, that is simultaneously referred to in the speech.

In addition to having the obvious function of indicating objects and places around the speaker, deictic gestures, or points, also play a part in narration and other discourse situations where there is nothing objectively to point at (McNeill & Levy, 1993). In storytelling, for example, such abstract pointing, deictics, establish characters in space and mark the occurrence of narrative boundaries. An example is a speaker who said, "the artist and the girl are walking by," pointing fist to his right and then making an iconic

gesture for walking by. In this case, the pointing gesture set up the characters in the scene and the iconic gesture depicted their action.

3.2.3.2 Symbolic Gestures

These gestures, also called *emblems* by Efron (1941) and Ekman and Friesen (1969, 1972), are representations that have one-to-one correspondence with meanings. They are devoid of any morphological relationship with the object that is being represented. A couple of common examples of an emblem may be a circled thumb and forefinger, used to convey the work "OK", and hand waving as a greeting gesture to mean "hello". Symbolic gestures are included in most of the different classifications, and furthermore, have very strictly defied characteristics in nonverbal communication.

According to Ekman and Friesen (1969, 1972), emblems are those nonverbal acts (1) which have a direct verbal translation usually consisting of a word or two, or a phrase, (2) for which a precise meaning is known by members of a group, class, subculture, or culture, (3) which are most often deliberately used with the conscious intent to send a particular message to the other person(s), (4) for which the person(s) who sees the emblem usually not only knows the emblem's message but also knows that it was deliberately sent to him, and (5) for which the sender usually takes responsibility for having made that communication. A further characteristic of an emblem is whether it can be replaced by a word or two, its message verbalized, without substantially modifying the conversation.

People are usually aware of their use of emblems just as they are aware of the words they speak, which may not be the case when using some other types of gestures that may occur unconsciously during speech. They would be able to repeat the gesture if asked to do so. Similarly, the use of an emblem is usually an intentional, deliberate effort to communicate. Moreover, emblems most often occur when verbal discourse is prevented by some external circumstance (e.g., between pilot and landing crew), by distance (e.g., between hunters spread out in the field), by agreement (e.g., while playing charades), or by organic impairment (e.g., deaf mute). Emblems also occur during conversation in

repeating a verbalized message, replacing a word or two, adding a separate message not necessarily part of the verbal discourse, or contradicting the verbalization. As defined as such, emblems are communicative and interactive acts. Emblems can involve actions in any part of the body, although usually they involve the hands, head orientation, facial expression, and posture.

Table 3-1 shows a summary of the gesture classifications used by the major researchers in this field. As the table shows, all the psychologically-based researchers owe their classification systems to Efron's initial work (Wexelblat, 1994).

Table 3-1: Summary of Gesture Classification Systems

Efron (1941)	**Ekman & Friesen** (1969)	**McNeill & Levy** (1982)	**Rime & Schiaratura** (1991)	**Kendon** (1993)	**Identifying Characteristics**
Kinetographic	Pictographs	Iconic	Physiographics	Physiographic	Picture the content of speech
Ideographic	Ideographs	Metaphoric	Iconic	Ideographic	Portray the speaker's ideas, but not directly the speech content
Baton	Baton	Beats/ Butterworths	Speech-making	Gesticulation	Marking the rhythm of speech
Symbolic/ Emblematic	Emblems	Symbolic	Symbolic	Automomous gestures	Standarized gestures, complete within themselves, without speech
None	Deictic Movements	Deictic	Deictic	None	Pointing at thing/area; space around body used

3.3 Gestures and Speech

Having defined the various types of gestures that are possible, we must now see how gesture may be related to speech. Several researchers have constructed theories in this regard to explain gestures in their relationships to the speech process. Among them are

views put forth by Freedman (1972), Kendon (1994), and McNeill (1985). All three theories rest on the assumption that gestures originate somewhere in the process through which unarticulated mnemonic elements are translated into the articulated speech formats. The three views are different from one another in the respect to the reason for the emergence of gestures.

3.3.1 Freedman: Gestures as Instruments to Speech

Freedman views body movement and speech as two components of a common symbolic process. Among body movements, those of the hands accompanying either the content or the rhythm of speech play a critical role in verbal coding. He believes that they define the structures underlying speech. With regard to this, two types of speech-accompanying gesture that ranged along a continuum have been distinguished (Freedman, 1972). At one extreme, *speech-primacy* gestures (or speech-markers in Efron system (Efron, 1941) (Efron,1971)), which closely parallel the formal and rhythmic properties of speech, exist as the most integrated gestural element that occur in parallel with verbal speech. At the other end of the spectrum, motor-primacy gestures, which consist of movements that are representational in themselves, exist as the least integrated into the verbal content as they represent some object, idea, or event.

When speech-primacy gestures (speech-markers) predominate in the speaker's expression, speech is usually the primary vehicle of the message. In this case, the syntactical structure and language forms tend to be very large and complex, and thus had gestures play a minimal role in such context, being limited to a self-monitoring and clarifying function. On the other hand, motor-primacy (iconic) gestures play a bigger role, representing a visible expression of the speaker's thought. Freedman believes that the motor-primacy responses indicate a failure to verbalize and to transform images into words (Nespoulous, 1986).

Freedman's views of gesture and speech are summarized in Figure 3-1. First, the mental image that the speaker is trying to articulate is translated into gestures. When this

is done successfully without any trouble, no motor-primacy or iconic gestures are expected. Second, an inverse relationship is predicted between the level of speech and sophistication and the patterning of gesture. The less articulated the speech is, the more the gesture will be patterned, and vice versa. Third, the gesture always fulfills some role with respect to the speech process, either through the self-monitoring function of speech-primacy gestures or through the reactivation of images an their link to words in the case of motor-primacy gestures.

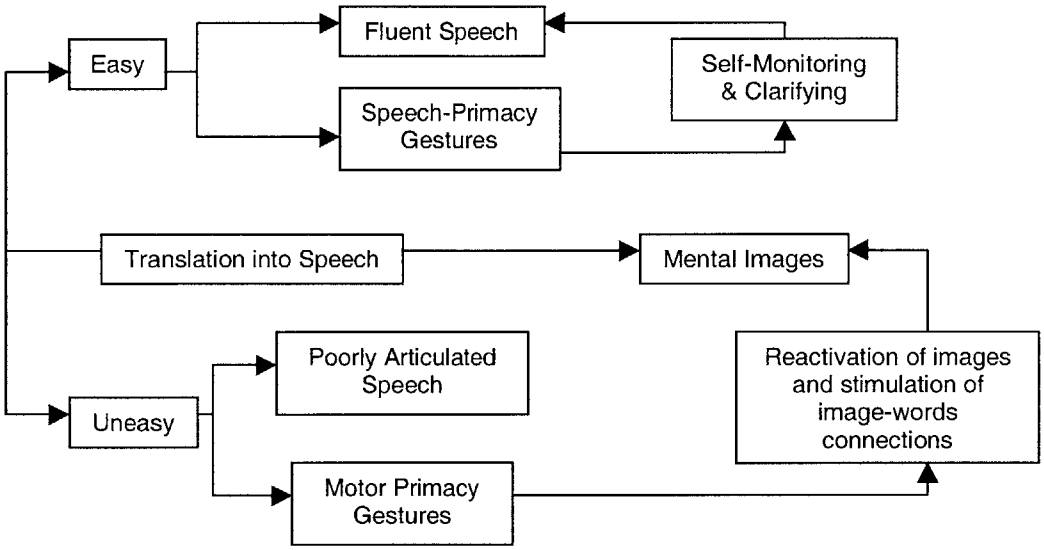


Figure 3-1: Freedman's theory on relationship between speech and gesture (Adapted from Rime & Schiaratura, 1991)

3.3.2 Kendon: Gestures as Substitutes for Speech

Speech-accompanying gestures, termed *gesticulation*, are characterized by Kendon as an integral part of an individual's communicative effort (Nespoulous, 1986). He believes that gestures do not represent redundant material that is also represented as words. Rather, gestures allow for representing aspects of the communicative experience that can be represented in words at best only indirectly and, in some respects, not at all. For

example, it is impossible to display the occurrence of an action except through some form of action, or to represent spatial arrangements without moving the hands and the body.

The view that gesticulation is a part of a person's communicative effort holds for all types of gesticulation, even for those that comprise of only rhythmic spatial patterning without informational content or any apparent communicative function. Kendon believes that such hand gestures actually constitute part of a person's actions used to represent meaning to another. In this sense, gestures are employed not to meet the transmission conditions of the interactional event, but rather to meet the requirements of representational activity. Therefore, when gesticulation takes this form, it functions as a visual analogue of the phonological "chunking" carried out by stress intonation and pause. With iconic gestures, the speaker appears to be visually representing aspects of content that are not referred to in the verbal component of the utterance. When gesture is that of emblems or symbols, the speaker uses gestures as an alternate to speech, letting them do the work of the speech component itself. Thus, according to Kendon, gestures are just another resource by which the speaker gets his meaning across.

According to Kendon, these are the conditions under which a person may use gestural expressive mode (Rime & Schiaratura, 1991):

1. To supplement speech when certain environmental factors, such as distance or ambient noise, make it difficult for speech to be received.
2. As a substitute for speech when the speech channel is already occupied by another speaker.
3. As a device for completing a sentence that, if spoken, might prove embarrassing to the speaker.
4. To clarify some potentially ambiguous word.
5. As a means of telescoping what one wants to say, when the available turn space is smaller than one would like.
6. As an additional component of the utterance, when the spoken account appears unlikely to approach the suitable representation of what the speaker intends.

Kendon's views on gesture and speech can be summarized in Figure 3-2. Inherent limitations the verbal channel of expression as a conveyor of meaning, combined with any potential constraints exerted on this channel by certain conditions, contribute to the speaker's using the more flexible gestural expressive mode as an alternative for conveying meaning. Therefore, speech-accompanying gestures usually represent components of the utterance content that are not represented in spoken words.

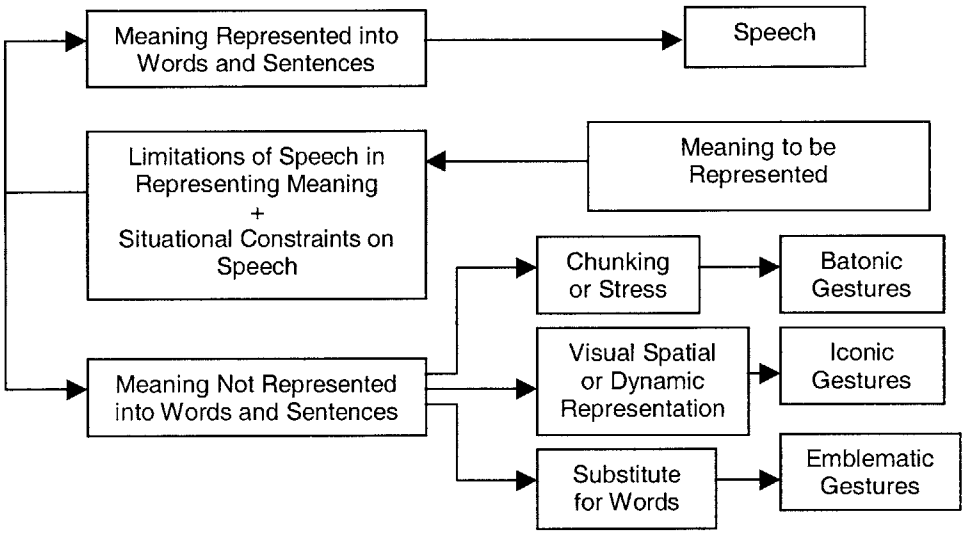


Figure 3-2: Kendon's theory on relationship between speech and gesture (Adapted from Rime & Schiaratura, 1991)

3.3.3 McNeill: Gesture and Speech as Parts of the Same Psychological Structure

McNeill (1985) holds the view that gestures and speech share a computational stage and that they are, accordingly, parts of the same psychological structure. This notion of a shared computational stage represents the processing aspects of speech, and further states that sentences and gestures develop internally together as psychological performances. The notion of a common psychological structure captures the idea that speech and gesture respond to the same forces at the same time.

The occurrence of gestures as a person speaks reveals that he or she is engaging in two types of thinking simultaneously: imagistic (i.e., thinking that is global and synthetic) and syntactic (i.e., thinking that is linear and segmented). The linguistic act creates a synthesis of both types of elements. Images are shaped by the system of linguistic values, and sentences are shaped by the images they unpack. A critical concept in McNeill's view is that the sentence is, at all stages, both imagistic and syntactic. There is no break between input and output in this situation, but only different developmental stages of a single process. Influencing one another, gesture and speech refer interactively to a single memory system in which complex configurational structures are stored.

Because there are such close connections between gestures and overt speech, gestures act as a second channel of observation of the psychological activities that take place during speech production, where the fist channel would be verbal speech itself. The channels of gesture and speech are close but different. McNeill believes that combining a spoken sentence and its concurrent gesture into a single observation gives two simultaneous views of the same process.

McNeill's view of the relationship between speech and gesture evolved from his conception of how knowledge is stored mentally. He views meanings as being stored in complex configurational structures independent of language format. In the speech process, meanings are transformed directly into either linguistic or gestural form, which is chosen by the speaker.

3.3.4 Summary

From the theories of these researchers as presented above, it is not difficult to see the inherent connection between gestures and speech. Gestures support and expand on information conveyed by words. Not only are the meanings of words and of gestures intimately linked in a discourse, but so are their functions in accomplishing conversational work. It is clear that, like facial expression, gesture is not a kinesic act independent of speech, or simply a translation of speech. Rather, gesture and speech are

so intimately connected that one cannot say which one is dependent on the other, and as McNeill has proposed, both can be claimed to arise from a single internal encoding process (McNeill, 1985,1992).

3.4 The Role of Gestures in Nonverbal Communication

With a definition, classification, and the relationship of gestures to speech and communication set forth, it is possible to see that gestures, as a part of the nonverbal repertoire of human beings, is very relevant and significant to any kind of interaction among people where communication is paramount. Along with facial expression, gaze, and stance, gestures are part of that nonverbal behavior that plays a central role in every day social interaction. Our ability to understand others, and the responses that we make to them, is based in large part on our ability to use effectively the nonverbal behavior that is displayed in any interpersonal interaction.

Consider, for instance, the example of two students to whom a professor has just returned their corrected midterm exams with the grade at the top of the paper. The classmates, seated next to each other, may have very different nonverbal reactions. One smiles broadly, pumps his fist, and leaves the paper on his desk. The other frowns, sighs, lets his hands and arms hang and then quickly puts the paper away. The first student, seeing his classmate's reaction, then tried to make eye contact with the second and adopts a look of what he hopes is sympathy, despite the fact that she can barely contain his own feelings of delight at doing well on the exam.

No matter how simple this fairly simple social interaction may be, it serves as a good example of what mechanisms and factors are in play. There are also opportunities for miscommunication in this interaction that used no spoken words. The success of this interaction in terms of the participants' conveying what they wanted to communicate to one another, relies on the abilities of the participants to encode and decode nonverbal behaviors appropriately. It is clear that effective nonverbal communication is critical to the success of particular social interactions. As shown in this simple example, roles of

the nonverbal elements, in which gestures are included, to express and convey emotion and meaning are critical in the success of a particular social interaction.

Current approaches to nonverbal communication try to assess the extent to which bodily signals can translate in visible form unobservable mental states like subjective emotion, interpersonal attitudes, or personality traits, and the extent to which people can, from perceived body signals, infer psychological states that may or may not correspond to the true conditions of elicitation (Feyereisen & de Lannoy, 1991). Thus, in exploring such roles and extent of gestures, the studies may be distinguished first, according to the nature of the conveyed information (i.e., emotions, attitudes, etc.) and, second, in each of these cases, accordingly to the perspective chosen in describing communication (i.e., the emission or the reception of body signals).

3.4.1 Communication of Emotion and Behavioral Attitudes by Gestures

In emotional expression, the quality and intensity of the affect can be distinguished and differentiated (Ekman & Friesen, 1969). The hypothesis has been proposed that the nature of the emotion is mainly expressed by facial movements, while the corresponding intensity cues of the emotion is displayed through body signals. Thus, gestures and postures could be interpreted as signs of emotional arousal, but little information about the nature of the emotion could be conveyed through this channel. Such a hypothesis seems counterintuitive, however, when one considers gestures that express states like happiness, anger, or fear that may be expressed by manifestations of triumph by an athlete after a goal, blows or kicks as a sign of anger, or face covering as expression of fear. These movements have been described as emblematic expressions of emotion (Sogon & Masutani, 1989 as cited in Feyereisen & de Lannoy, 1991).

3.4.1.1 Sending Cues of Emotion and Expressing Interpersonal Attitude by Gestures

Gesture expression can be affected by certain conditions that control the level of emotional arousal. For example, talking about a sad, boring, impersonal topic and talking

about a funny, interesting, personal occurrence will produce different levels of gesturing. In this manner, there is little doubt that emotional arousal may influence gestural activity, but several questions still remain open. First, different movements such as illustrative, as opposed to self-touching gestures, might relate to qualitatively distinct states. According to Ekman and Friesen (1969, 1972), automanipulative gestures called "adaptors" allow one to cope with such emotional states as depression and embarrassment. Nevertheless, it can be hypothesized that other emotional experiences, such as anger or disgust, might not be managed in similar ways. Similarly, within a single category of gestures, the relationship between arousal and motor activity might vary according to the nature of the emotion.

In interpersonal relationships, body movements may reveal whether interaction is desired and may influence the reactions of the partner who perceives them. Together with smiling and eye contact, orienting the torso toward the partner, leaning forward, nodding the head, and gesturing have been studied as expressions of involvement or affiliation, whereas self-touching, leaning backward, and moving the legs are signals of more negative feelings.

Gestures and postures may also be influenced by other variables and thus transmit other information relative to status, competence, assertiveness, or sincerity. For instance, self-confidence or dominance may be expressed by gesturing, touching the partner, or being relaxed. Inversely, shyness, anxiety, low self-esteem, or submissiveness may be manifested by head lowering, increased self-touching, or postural changes.

3.4.1.2 Perceiving Cues of Emotion and Attitudes from Gestures

There have not been much research to investigate whether facial and bodily indications play separate roles in conveying information about emotional state. Little information is available on the attribution of emotional states like joy, sadness, fear, or anger from body movements isolate from facial expression. However, gestural cues have been used in rating the target person on dimensions like calm-agitated, relaxed-tense, or unexpressive. Embarrassment may be recognized from body movements such as hand

gestures, whereas amusement is not effectively read by gestures but rather by facial expressions.

Judgments of the attitudes of others are influenced by perceived gestural behavior, even if these cues are sometimes of minor importance in comparison with vocal and facial ones. For instance, bodily signals become more decisive in friendliness ratings when a neutral facial expression is displayed. In Western societies, "closed" postures with folded arms and tight knees, give an impression of passivity and coldness. A listener may be judged more or less involved in a conversation or to agree or disagree by means of head orientation and arm and leg positions. Posture may also modify the impression formed from other nonverbal cues like gaze or interpersonal distance. Level of appreciation is also influenced by nonverbal signals such as standing up or sitting down, erectness of posture, gaze orientation, and facial expression. In regard to gestures, brief self-touching of the face and palm presentation movements give higher impressions of warmth and naturalness than immobility.

3.4.1.3 Social Perception and Social Impact of Hand Gestures

Specifically talking about gestures that are performed in conjunction to verbal speech, Rime and Schiaratura (1991) contend that hand gestures carry no significant role in complementing or supplementing the decoder's (the person perceiving the gesture) ability to understand the conveyed message that is given verbally along with the accompanying gestures. Three main arguments are made, the strongest being that the decoder is unable to guess the speech content to which gestures relate, which support the view that hand gestures (excluding *emblems*) do not provide independent access to a meaning expressed in words. This view is a contradictory to ones held by Kendon and McNeill as discussed before. Despite the contradictory views held by different researchers, the fact that gestures do accompany speech and is inherently incorporated into the human communication behavior suggests that gestures play a role in natural human discourse. The absence of gestures during discourse would suggest a certain level of awkwardness and unnaturalness, since humans perform gestures unconsciously whether or not they complement or supplement speech. To this extent, the view taken in this thesis is that,

gestures do exist in face-to-face human interaction and communication, and thus is an important part of natural human behavior in communication efforts.

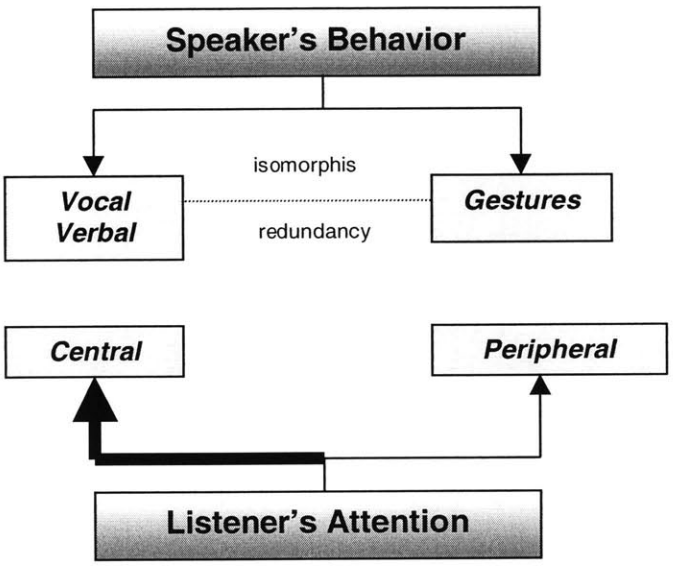


Figure 3-3: Figure-Ground Model of the listemer's attention to the speaker's verbal and nonverbal behavior (Adapted from Rime & Schiaratura, 1991)

During social interactions, the context in which nonverbal signals have to be understood is mainly constituted by the behavior of the partner. In this paradigm, the listener's (decoder) attention during an interaction is given mainly to the verbal aspect of communication, as shown in Figure 3-3 by the bold line indicating the listener's main channel of communication. Rime and Schiaratura (1991) develop this figure-ground model, suggesting that gestures and nonverbal behavior is only at the periphery of the listener's attention. Nevertheless, the nonverbal data, or the "elements of the ground", are always present as potential attention grabbers. The figure-ground model also implies that the relation may slip and reverse temporarily, so that nonverbal elements take the role of the figure, thereby occupying the center of the perceiver's attention. Gestures and nonverbal communication become important when certain conditions such as rise in intensity in the nonverbal channel and a fall in intensity in the verbal channel occur (e.g.,

verbal signal weakens, noise occurs in the communication, or the intelligibility of the speech content itself degrades) as shown in Figure 3-4.

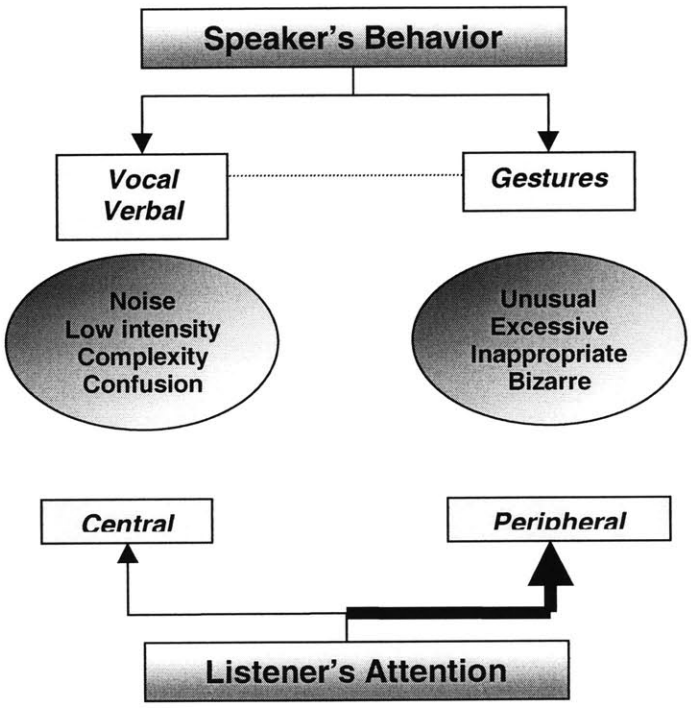


Figure 3-4: Figure-Ground Model when major conditions call for reversal in the listener's attention to the speaker's verbal and nonverbal behavior (Adapted from Rime & Schiaratura, 1991)

3.5 Summary

The theories of gestures and their roles in communication and social interaction were developed for face-to-face human interactions. Nonetheless, when considering such interactions in non-traditional manner, such as over the Internet through a virtual collaborative environment, same phenomenon of human behavior should apply. Much of the concepts of communication and the crucial role of gestures can be applied to social interactions in virtual environments through the representation of our human bodies. When humans are represented in a virtual environment within the context of social interaction amongst each other, the goal of such implementation becomes that of

replicating modes of human behavior as closely as possible to those of physical face-to-face interactions. With such an aim to naturally express ourselves in a non-face-to-face environment, it becomes imperative that certain features and elements of expression, such as gestures, are available.

This issue of expression of gestures in virtual environments will be explored further in detail in Chapter 5. However, in order to understand the perspective of how social interaction may be included in virtual collaborative systems, CAIRO is re-examined in the context of DiSEL '98-'99's efforts in the following chapter.

Chapter 4

DiSEL 1998-1999

Distributed Software Engineering Laboratory is the project team consisting of students from MIT and CICESE in Mexico. It is geographically distributed by design and nature. It also served as a working environment as well as experimental case example of distributed software engineering and distributed collaboration.

4.1 DiSEL

The goals of DiSEL for the academic year 1998-1999 was multifaceted. The context in which the team operated on was to enhance and work in a collaborative, distributed, cross-cultural environment to go through the lifecycle of a software engineering project. The idea of collaboration was crucial and served as a focal point. In defining the bounds of collaboration, social interaction, cross-cultural, and time and space aspects were dealt with. Because the nature of DiSEL was distributed, development of the project with distributed team members served as a learning example of distributed collaboration in action.

The team was organized and operated into a small software company paradigm where each team member took on a particular role from various responsibilities that included: project management, requirements analysis, design, programming, quality assurance, testing, and configuration management. This company oriented structure and the respective development roles that the members took on allowed the team to look into bringing the product, CAIRO, into the market as a real software company would

consider. The combination of creating a company oriented team structure and fostering initiatives to develop the software to sell in the market naturally led way to exploring ideas of entrepreneurship. This was accomplished by writing a business plan that was submitted to the MIT 1K and 50K Entrepreneurship Competitions with success, winning the 1K Competition in the Software Category, and reaching the Semifinalist stage of the 50K Competition (MIT 50K Entrepreneurship Competition).

To this aim of fulfilling a multi-dimensional goal of successful distributed collaboration and software development, DiSEL developed a software tool to enhance the social interaction element of collaboration, whose importance and relevance was discussed in the previous chapters. Specifically, the team worked on development of two features that would increase social interaction capabilities of the existing CAIRO system. Firstly, the group worked on a three-dimensional interface for CAIRO that would create a virtual collaboration and interaction environment where avatars would be used to represent the users. Secondly, an awareness driver was implemented to measure and detect the affective states of the involved users.

4.2 3D Interface

One of the requirements for DiSEL '98-'99 team was to design and develop a three-dimensional environment user interface to enhance personal expression and thus increase effective level of social interaction among the users. The project's goal included the development of a three-dimensional world for the users where they would be represented by avatars. Through the avatars, the users would be able to interact and conduct work as they did in the 2D version of CAIRO. It is with the aim of improving social feedback mechanisms that the three-dimensional environment was proposed and worked on.

The project team worked on one aspect of personal expression and social feedback in the context of the 3D environment – facial expression. However, as mentioned in the previous chapter, there are many components to an effective social interaction via personal expression, such as hand, body, and hand signals that are transmitted and

interpreted by the users. This thesis will extend the capability of expression in a 3D environment by exploring the technology of a prototype gesture expression tool in this 3D environment. In this environment, the users, and naturally their personal expressions, are represented by avatars. Recall that gestures encompass aspects of head, hand, and body signals in the context of personal expression and social feedback as outlined previously.

Relevant parts of the requirement analysis of DiSEL '98-'99 is included in the following section to outline the general background and scope of the 3D interface implementation, and to put the gestures and gesture representation tool into perspective with regard to the CAIRO collaborative environment.

4.3 DiSEL 1998-1999 Requirement Analysis

Through a theoretically recursive process by which distance collaborative work was conducted to enhance a distance collaboration tool, the CAIRO architecture was enhanced with features mentioned above. CAIRO version 3.0 refers to the version of CAIRO with the 3D interface and awareness feature worked on by DiSEL '98-'99.

4.3.1 Scope of the Software

CAIRO v3.0 is expected to serve as full functional stand-alone client-server software. The current software is expected to serve as a new version of CAIRO and will build upon previous versions of CAIRO. This effort is expected to reuse previous work done on CAIRO in the form of software code and relevant project documentation.

4.3.2 Definitions, Acronyms and Abbreviations

- *Casual contact*: unplanned, spontaneous social interaction among workgroup or class participants occurring outside of planned sessions and meetings

- *Social feedback*: elements of a communication which are unspoken, the so-called interpersonal "noise" (Johanson, 1977) which are an intrinsic and necessary part of communication
- *Affective Computing*: representation of emotions on a computer
- *Affective State*: internal dynamic state of an individual when he/she has an emotion (Picard, 1995).

4.3.3 CAIRO v3.0 Perspective

The DiSEL '98-'99 goal is to expand on the existing infrastructure of CAIRO. This endeavor will lead to CAIRO v3.0 by furthering the development of a virtual collaborative environment where distance collaboration would be enhanced. The social interaction capability including the casual contact capabilities of earlier version will be maintained, and the communication tools will be expanded, as will the social feedback tools. It is relevant to detail the capabilities of CAIRO v2.0 and to explain which new features was added in version 3.0. A brief list of CAIRO v2.0 features follows:

1. Communication tools
   - Chat (transient, immediate messaging)
   - Whiteboard
   - Session (meeting) Manager

2. Social Feedback tools
   - Facial expressions

3. Casual contact tools
   - Three light system

CAIRO v3.0 will expand the current functionality by adding these social feedback tools

- Objective measure of attitude with sensors

- Three-dimensional interface for gathering social feedback
- Three-dimensional environment for user interaction
- Two-dimensional interface for gathering of social feedback available if using the CAIRO 2.0 collaboration interface

Several types of users can be identified for this kind of collaborative tool:

- Designers in the commercial world collaborating with other designers that are separated across geographical and cultural boundaries.
- Home users collaborating with other home users.
- Students collaborating with other students separated across geographical and cultural boundaries.

4.3.4 Background for social interaction in virtual space and affective computing

"Computer-based teleconferencing is a highly cognitive medium that, in addition to providing technological advantages, promotes rationality by providing essential discipline and by filtering out affective components of communications. That is, computer-based teleconferencing acts as a filter, filtering out irrelevant and irrational interpersonal "noise" and enhances the communication of highly-informed "pure reasoning" – a quest of philosophers since ancient times" (Johansen, 1977).

This highly misguided view of the communicative abilities and needs of people was presented 20 years ago. Since that time, it has become evident that the "noise" and other so-called irrelevant inter-personal signals, which were so easily dismissed, do in fact constitute an integral and irreplaceable part of communication. This is because they condition a person's response to a situation, or social interaction, since they are part of the very definition of that situation (Matovani, 1996). No amount of enlightened theorizing can substitute itself for the communicative information that is provided by social feedback in a conversation.

In *The Network Nation* (Hiltz, 1978), the authors point out the importance of visual information (e.g., height, clothes, and jewelry), facial expressions (e.g., smiles, frowns, head shakes), eye contact (e.g., evasive, direct, and closed), and body movements (e.g., hands, and feet tapping) in clueing between two or more communicating individuals. In short, clues are given about a person's affective state by a combination of the factors mentioned above. In addition, Hiltz (1978) identifies several "uncontrollable" psycho-physiological responses such as yawning, blushing, eye blinks, which provide cues about the interlocutor about his or her emotional state. Clearly, these cues are only valid within a particular cultural frame of reference: who is to say what a Papuan means by shaking his head unless one has had previous encounters with Papuans?

A first-generation computer communication system, which would strive to provide social feedback in the form of clues, would not be responsible for ensuring that the cues provided by its users are understood, since that rarely happens in real life. However, it can be envisioned that future computer communication systems would allow their users to choose a culture within which they want to operate and adequately translate their behavior into the corresponding behavior for the other users, according to their own chosen cultural frame of reference. An example is useful at this point:

- In culture A, shaking one's head left to right means "no"
- In culture B, shaking one's head left to right means "yes"
- When A shakes his head (or gives the equivalent signal using the keyboard), the computer "translates" the behavior so that B will see the head movement which corresponds to "no".

Of course, this example makes assumptions about the functionality of the computer communication system, but the point is clear. Now, however, the challenge still lies in providing social feedback in computer-supported communications.

Modeling and representing the cues used for social feedback is inherently difficult. However, if one is to refrain from using teleconferencing, it must be achieved so that

collaborating individuals feel as comfortable using a computer-based communication tool as they would talking face to face (where face to face conversations include video conferences). The goal is to make the representations of real-life social feedback as close to the originals as possible, given the inherent limitations of computer-based communications.

In order to remedy the admitted lack of social feedback in computer-supported interactions, several challenges need to be overcome since computer based collaboration encompasses a variety of complex systems. However, the computer-based collaboration problem can be decomposed into four sub-challenges: the collocation, cooperation, communication, and coordination challenges. The general requirements are based on the fact that physical meetings require four essential components in order to take place:

- A physical meeting room in which the participant can meet, and where social feedback may be exchanged. This is the collocation prerequisite.
- A common agenda and an individual, or set of individuals, which will determine the format of the meeting (note that for an unplanned meeting, the instance of an agenda will be simply empty). This is the coordination prerequisite.
- A means, by which information can be exchanged, whether voice, writing, or video. This is the communication prerequisite.
- A topic of discussion, which can be impromptu or agreed upon in advance. This is the collaboration prerequisite.

These general requirements have been considered and incorporated into the CAIRO system design as discussed in Chapter 2. The co-location, cooperation, coordination, and documentation model of CAIRO supports such computer-based collaboration outlined above. At the basic level, the existing CAIRO system allows the creation of forums, meetings, and supports basic levels of communication via text chat. In addition, to enable a higher degree collaboration environment, CAIRO supports different meeting styles, chosen by the chairman who conducts the meetings in this virtual environment.

In addition to these general requirements, it is important to ensure that casual interactions occur during the distributed collaborative exchange taking place on the computer. In this context, "casual social interaction during the distributed collaborative exchange" is used to refer to unplanned, spontaneous social interactions between individuals, which occur outside of arranged sessions, reviews and meeting. The ability for people to experience casual interaction will strengthen the bonds between members of work groups, whether student or professional, thus furthering the sense of community between distributed individuals. Casual interaction can happen when individuals are aware of each other's presence in a given space. They can then decide whether to engage in a interactive activity (such as a conversation, or a simple exchange of hellos) with the other individual which has been encountered.

4.3.5 Specific Requirements

The finished product facilitated the interaction of distributed people by providing social interaction to be used in remote communications situations. Remote communications situations are defined as communications events in which at least one of the parties is not collocated with the other parties. The goal of providing personal expression and social feedback is to allow for users to communicate according to the habits which they have acquired in face to face communications. Typically, when communicating through a computer, individuals are shielded from body language messages and other kinds of social feedback. The aim of CAIRO 3.0 is to allow user to exchange this social feedback, and to be able to use it in striving towards better communications. In order to allow social feedback between the users of CAIRO the following are necessary:

0. A casual contact capability must be provided (CR0).
1. Sensing devices must provide a measure of attitude (or affective state) of the users (CR1).

2. A viewing environment must provide feedback a user about the state of the other users (CR2).

These three requirements form the basic user requirements that must be met. The specific implementation and use cases pertaining to these requirements will be discussed in the following subsections. The following requirements detailed in the following subsections are ones developed by DiSEL '98-'99 in developing the next generation of CAIRO, denoted as CAIRO version 3.0. These requirements were used in completing the project which added social interaction capabilities of a three-dimensional user interface and user's affective measurement mechanism to CAIRO, in order to enhance the effectiveness of collaboration in a virtual environment.

4.3.5.1 Customer Requirement 0

Casual contact capabilities with a basic level of personal expression through the facial expression driver were already implemented in version 2.0 of CAIRO. It is nonetheless relevant to include them as an explicit requirement of CAIRO v3.0 so that they will not be inadvertently removed during the upgrade. Because of the option to keep the CAIRO 2.0 metaphor for casual contact, this requirement does not contain the implementation specifications provided for the other requirements. It is important to realize that the designers will have the option (unlike for the other requirements whose specifics are determined in this document) to implement this requirement as they choose. Based on the requirements document for CAIRO v2.0 (DiSEL 1997), the following characteristics are required of a system enabling casual contact:

1. Casual contact must include a sense of randomness.
2. Users can set their availability for casual contact.
3. Users can either manually signal their availability for casual contact, or they can opt for the computer system to set them when a set of predetermined actions occurs.
4. All users of the system must be able to see the availability of other users
5. Users must be able to receive and display casual contact activation signals, i.e.: a user must able to elicit a response from another user. Activation signals

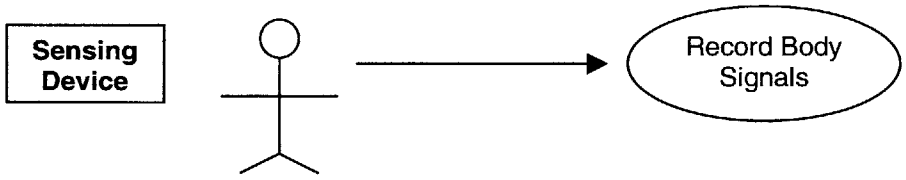
are signals send from one user wanting to engage in casual contact to the user with which he/she wants to engage in this casual contact

6. Users must be able to respond to a casual contact activation signal.
7. Users must be able to accept a response.
8. Users must be able to choose which mode of communication they can use once contact has been established.

4.3.5.2 Customer Requirement 1

A set of sensing devices will be worn by all users who have opted to submit affective state feedback to the CAIRO system. Picard (1995) states that a person's "affective state cannot be directly measured by another person, but may be inferred". This affective state of a person (i.e., the user) is transmitted and perceived by emitting certain signals. These signals are transformed into an "expression", which is then used by others to infer one's affective state. The sensors will measure bodily characteristics that will be translated into a metric representing the affective state of the user at a given time. These metrics will then be translated into expressions, which will be made available to the other users as will be explained in the details of Customer Requirement 2.

For the use case corresponding to this requirement, the only actor will be the sensing devices (which will be considered as a black box for this purpose). There is only one, very simple, use case, and it is presented below:



4.3.5.3 Customer Requirement 2

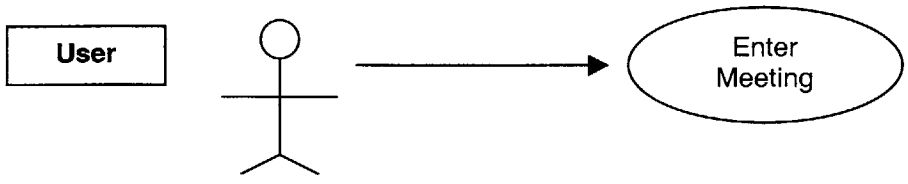
In order to use the social feedback provided by the software, the user must have access to this information. For users with access to the full capabilities of CAIRO 3.0, affective state feedback about all the users will be provided to each individual user in a

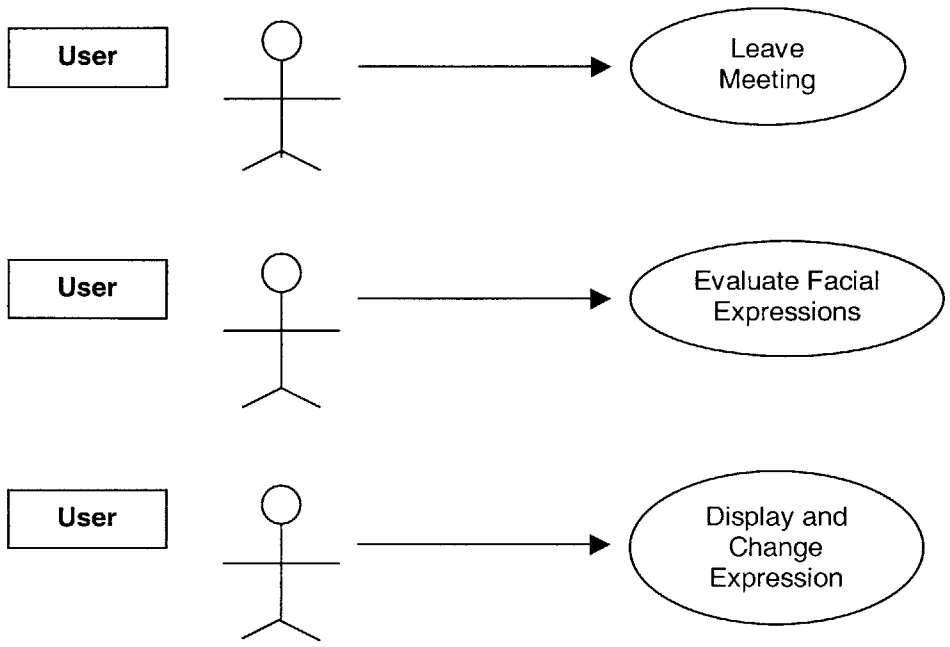
three-dimensional virtual environment. For users who will not have access to the full capabilities of CAIRO 3.0, the feedback will be presented in the form of attitude graphs which will be described below.

The 3D environment, which is to represent the room in which the meeting is actually taking place, will contain a representation of each of the participants. This representation (or avatar) will be modified in real time as a function of the "emotion" calculated from data produced by the sensing devices. In particular, the facial expression of the avatar will be variable, striving to replicate the facial expressions most commonly encountered in interactions between people. This three-dimensional interface will provide a life-like environment in which social feedback will be exchanged between the users of CAIRO. Each user will be able to see the meeting room in its entirety, but he will be limited to using his fixed position as the origin of his perspective. In this manner, things which are blocked from his view by objects or people will not be visible.

One room will be created for each conversation, exactly as in the present version of CAIRO. In addition, the rooms will be joined by three-dimensional hallways linking the discussion rooms, and the user will have access to a virtual posting board which will automatically hung on the wall of the hallway linking the first room to be created in a given session.

This requirement is associated with several use cases and actors, which are depicted below. More precisely, the individual user (user) can "read" or "evaluate" the facial expressions of the other users (participants). The user can also enter or leave the virtual meeting room, and the user can approach and activate the message board. In these cases the actor is the user. Lastly, the avatar can change appearance according to the input from the devices described in CR1. In this case the avatar is the actor.





For users whose computers might not support a three dimensional representation of the meeting room (for lack of an appropriate plug-in, for example), the CAIRO v2.0 version of the meeting environment will still be available. In addition to this two-dimensional environment, small attitudinal graphs will give a measure of the attitude of each participant. The graphs will depict a set of time dependent variables, which will provide to the user information such as the attention state, the level of nervousness, and the level of anger of the other participants.

Clearly, this solution is meant to be temporary, and the user will be encouraged to download free software from the Internet to support the 3D meeting rooms. Use cases will not be drawn for this requirement; the only difference with the above use faces because there is no change of the avatar in the 2D environment.

4.4 Summary

The DiSEL '98-'99 team has started the first move to implementing CAIRO as a full-fledged three-dimensional communication and collaboration tool over the Internet.

However, there remains much room for improvement in perfecting a virtual environment where effective social interaction by means of clear personal expression and social feedback can occur. The ultimate goal in any virtual environment is to recreate the face-to-face human encounters that people are used to working with in a networked computer system. This can be achieved by allowing the user to have the freedom to express themselves with minimal constraints and thereby interact efficiently and effectively, which in turn results in a better collaborative effort.

Gesturing is clearly a significant element in terms of social interaction (personal expression and social feedback). Given this contention, a three-dimensional environment as introduced in CAIRO can be enhanced by the addition of a gesturing function. Such a tool representing avatar gestures provides the users a greater degree of freedom in expression, which ultimately leads to a better communication that in turn increases the effective level of collaboration in such an environment. In the following chapters, such gesture representation in a virtual environment is explored.

Chapter 5

Gesture Expression in Virtual Systems

In any collaboration setting where people interact with one another to accomplish a task, communication is paramount in conveying one's ideas and clearly putting forward one's position or view. Extending this concept of communication in human interaction, it is natural that the group dynamics and individual participant's affective states also control and drive the effective level of the collaboration effort. In the context of collaboration effort, social interaction includes all such activities that occur when there is any kind of interaction between two or more people at a time. Communication as a whole, including transmission and perception of emotions and the subtle nuances of speech comprise a useful and effective form of social interaction. This chapter will discuss how such elements of communication in the context of social context can be applied to virtual environments and review some of the gesture systems that has been researched up to date.

5.1 Gesture representation in computer systems

We saw that in Chapter 2 that social interaction can be divided into two essential components, personal expression and social feedback, that are intrinsically tied to each other. Moreover, nonverbal elements of communication and repertoires of social interaction such as facial movements/expressions, gestures, gaze, and stance/posture play a significant role in personal expression and its transmission-perception mechanism loop denoted as social feedback.

Given this relevance and importance of nonverbal behavior including gestures, in particular, in settings where human interaction is involved, it is natural that this same conditions and theories hold for such environments recreated by networked computer systems. The basic concepts of social interaction among people and the theories of gestures in linguistics and social psychology can be extended and applied to virtual collaborative environments where multiple people log in to a computer system and interact to accomplish a common task.

5.2 Kinds of gestures relevant to computer expression

As explored in detail in Chapter 3, there are numerous types of gestures that has been researched and classified. Each of these gesture types are used at different point in time and context to serve different purposes. Gestures can be seen as being closely connected to speech and thus complementary to speech, adding unrealized value to the communication effort. For example, an iconic gesture that is concurrent with speech can be used to visually demonstrate a physical object to the other party. Another example would be a shrug gesture which can be used instead of speech at times which would indicate and convey an idea of "I don't know".

There are several types of gestures that can be attempted to be represented through animation in virtual environments (Cassell, 1994):

- *Emblems* represent body movements, such as hand waving, that can be equated with certain words or phrase, and thus can be used in place of direct speech to convey idea or feeling.

- *Iconics* represent some feature of the accompanying speech, such as sketching a small rectangular space with one's two hands while saying "so you have a blank CHECK?".

- *Metaphorics* represent an abstract feature concurrently spoken about, such as forming a jaw-like shape with one hand, and pulling it towards one's body while saying "then I can WITHDRAW fifty dollars for you".

- *Deictics* indicate a point in space. They accompany reference to persons, places, and other spatializeable discourse entities. An example might be pointing to the ground while saying "do you have an account at THIS bank?".

- *Beats* are small formless waves of the hand that occur with heavily emphasized words, occasions of turning over the floor to another speaker, and other kinds of special linguistic work. An example may be waving one's left hand briefly up and down along with the phrase "all right".

All of the above can be theoretically represented as animations in a virtual environment. However, when deciding the feasibility and appropriateness of what gesture types to implement, the system limits, constraints, and requirements must be considered. In doing so, certain types of gestures may be not desired mainly due to technology constraints and usability issues.

Problems with usability and technology or resource constraints arise when the speech-accompanying gestures (iconics, metaphorics, dectics, and beats) are concerned. This is so because there are severe limitations when trying to concurrently generate and animate gestures according to the speech and context.

There have been research that try to implement such speech-concurrent gestures, and they all require expensive and complex machinery and sensors that the user must wear in order to physically detect these physical movements (Wexelblat, 1994; Cassell, 1994; Vilhjálmsson, 1996,1997,1998). This approach of gesture representation is rather intrusive to the user, since very cumbersome and distracting equipment must be put on to accomplish this. This kind of method is especially inappropriate for such a collaborative system such as CAIRO where the main objective is effective collaboration through

realistic social interaction. In order to implement such a function into the three-dimensional environment would require much extra overhead on the system itself for extra modules and tools must be added on as shown in Figure 5-1.

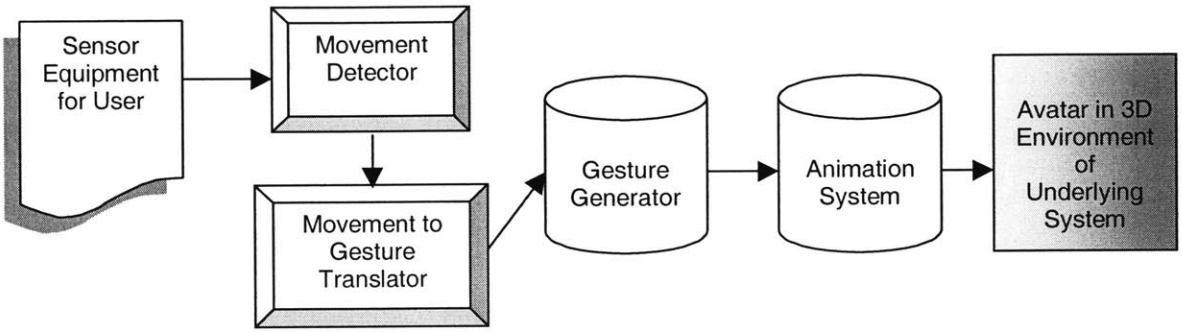


Figure 5-1: Sensor-based gesture system

Another possible approach that would not require sensors rely on the ability for the underlying text dialog system to detect and recognize certain word, words, or phrases in context to trigger animation of the relevant gesture. Once again, this poses much problems in that such an accurate speech context recognizer mechanism is hard to implement, and thus cannot be relied upon to correctly and appropriately animate gestures according to the speech that is being uttered by the user via avatar. This method requires the word and context recognizer to interpret parts of speech and in turn generate concurrent gestures. Developing such a tool that functions correctly and incorporating into the underlying system would be difficult at this point in speech and context recognition technology. Figure 5-2 overviews such a scheme.

Representation of emblematic gestures, on the other hand, does not involve such cumbersome complexities and is more suited for implementation and use by participants of collaborative virtual environments that we're concerned with. Because these emblems need not be associated with speech concurrently, it reduces the overhead that may be put

on by other approaches described above. Moreover, emblems are very appropriate in communicating feelings and ideas in such a collaborative social interaction setting where only one person may be holding the floor, and thus have the sole right to express themselves. With the use of these gestures, users can freely express certain ideas simply and directly by triggering a gesture motion, instead of typing out certain words or phrases. This kind of reaction and method of communication is much more natural and is representative of real life face-to-face interactions.

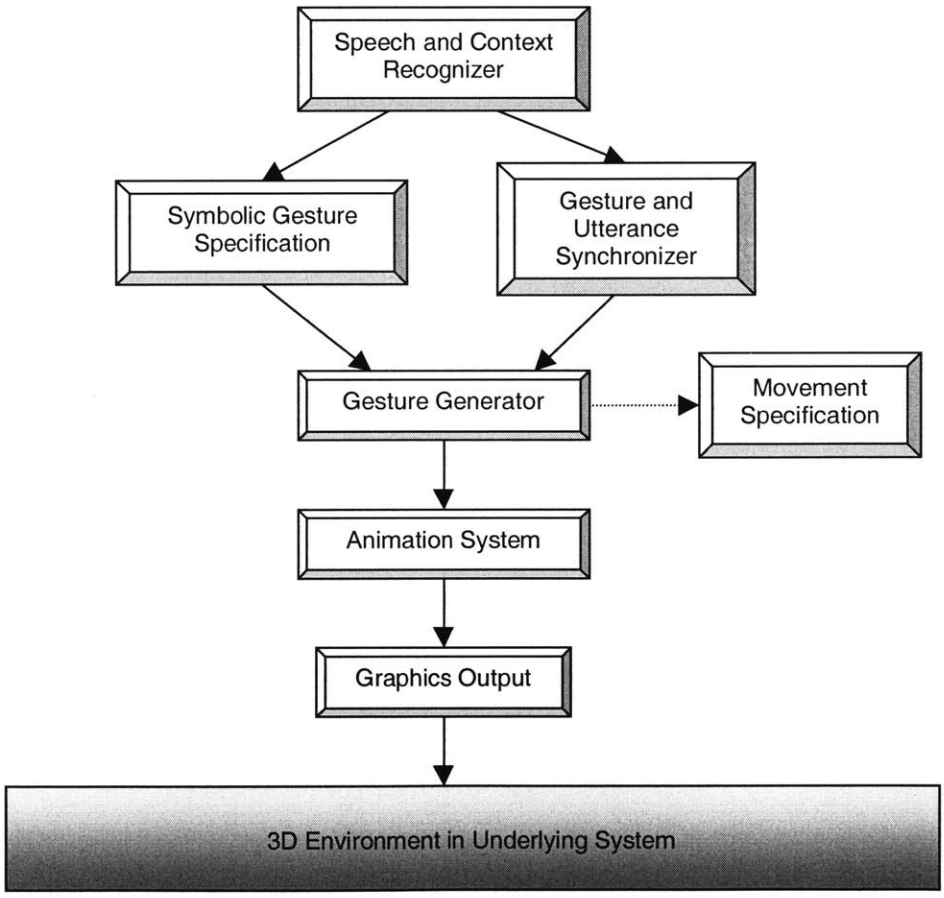


Figure 5-2: Speech recognizer approach to gesture representation

5.3 Emblematic Gesture Representation

Given that such emblematic gestures can be used to increase the level of effective communication and social interaction, it is natural to extend this to the avatar motions in the three-dimensional virtual environment.

Therefore, in order to further add value to the three-dimensional environment that has been interfaced with CAIRO by DiSEL '98-'99, an emblematic gesture expression function for the avatars is proposed. Such a functionality will undoubtedly allow a greater degree of transmission and perception of users' thoughts, feeling, or persuasions especially in situations where speech channel (i.e., text dialog) cannot be used and where a simple gesture can convey a meaning more directly and simply.

Chapter 6

Gesturing Expression Tool Prototype Design

In designing such a prototype, several factors are involved. Two factors that present itself as the biggest and more significant are: (1) requirements of the prototype, and (2) the technology and methods by which it will be implemented.

As proposed, the gesturing prototype will be a stand alone capable function that incorporates a Virtual Reality Markup Language (VRML) canvas or browser which will be interfaced with Java elements for user control in displaying VRML-based avatars in the virtual environment. The gesture animations are generated through the animation techniques of VRML, and thus, the VRML canvas should support and display animations readily.

This chapter will discuss the design of the gesture expression prototype. An overview and requirements of the prototype, as well the technical methodology that enables this will be explored. Specifically, technology and design behind the VRML avatar representation, VRML execution model and avatar animation model, and VRML-Java interface will be focused on.

6.1 Requirements

In order to enhance social interaction functionality of users in a virtual environment, a gesturing representation tool that expresses a predefined set of emblematic gestures is required. The user should readily have access to this gesture expression by the click of a mouse button which will generate and animate the gesturing motion on the avatar. This gesturing animation then would be able to be perceived by other users that are present in the three-dimensional world.

6.1.1 Technology

The design of this gesture expression prototype is based on the assumption that it will be implemented in conjunction with an existing real-time collaboration system with social interaction capabilities, such as CAIRO. Because such a system is used over the Internet and its underlying TCP/IP protocols, the gesture expression system that will enhance social interaction capabilities via a three-dimensional interface will also have to be optimized to run over the Internet.

Having this technical requirement, the choice of VRML and Java technology makes sense because of its Internet-based features that optimize its performance. Virtual Reality Markup Language has become the standard for three-dimensional representation of objects and space over the Internet, and Java has also taken a prominent position in proving itself as a robust programming language that is geared for the Internet specifically. Although the technology of VRML and Java integration is fairly young and developing, for the purposes of designing and implementing a gesture expression application where VRML avatars would be animated to display a particular gesture as a form of communication, this methodology would work fairly well.

This particular design that is detailed in the following sections is of a avatar gesture expression prototype that would be built on top of an existing distributed virtual environment, as the one provided by CAIRO version 3.0 as discussed in Chapter 4. This being the case, the strategy is as follows: it is suitable to implement the avatar

geomoetries and gesture animations in VRML 97 and Java to interface the user interface of the gesture behavior control pieces with the VRML avatar components. With this done, a VRML/Java compatible canvas or browser would be used to display and view the result.

6.1.2 Representation

First, in order to consider a gesture expressing capability of the avatar in a virtual environment, we must define the range of gestures that it will enable. As outlined in Chapter 5, emblematic gestures will be used in this prototype design, and in this context, a limited collection of commonly used gestures in face-to-face human social interaction needs to be represented in the prototype. The considered gestures include:

- Hand wave
- Shoulder shrug
- Arm cross
- Finger point
- Fist down
- Head-hand shake
- Hand raise
- Shoulders down and head turned
- Head tilt and hand up to head
- Arms to hips, head tilt and shake
- Head nod

Each of these gestures convey an idea or state of emotion to the other users, thus generating personal expression and in return social feedback. These gestures, being emblematic in nature, have certain meanings that may be interpreted by other users in the social interaction scenario.

With a selected avatar, the user should be able to invoke any gesture at one time at will. The avatar will then be animated and displayed in the VRML canvas which acts as the interface to the three-dimensional environment.

Table 6-1: Emblems Represented with their Meaning

GESTURE MOVEMENT	CORRESPONDING MEANING
Hand wave	"Hello" or "Goodbye"
Shoulder shrug	"I don't know"
Arm cross	Listening
Finger point	Pointing something out
Fist down	"Dammit!" – anger or surprise
Head-hand shake	"No"
Hand raise	"Excuse me" – attention
Shoulders down and head turned	Disinterest
Head tilt and hand up to head	"I'm bored"
Arms to hips, head tilt and shake	Disapproval
Head nod	"Yes" - agreement

6.1.3 Use Case Diagrams

The use case diagrams that are shown below denote possible use case scenarios for the gesturing. The main functionality of the prototype is to provide the user with a way to express a certain type of gesture via an avatar in an interactive collaborative work session and/or in a casual contact social interaction case scenarios. The user here is defined as the person that the avatar represents in the virtual environment. The first set of use case diagrams (Figure 6-1) illustrates the simple basic gesturing behaviors that can be invoked by the user. The second set of use case diagrams (Figure 6-2) reflects the various possible scenarios, or the different types of gesture expressions, that the user can perform through the prototype when he is in the virtual environment supported by a networked collaborative system such as CAIRO.

In Figure 6-2, several situational environments are illustrated in the collaborative virtual environment. As is the case with Figure 6-1, the actors represent the user, or the participant, in the virtual environment who is represented by an avatar. The first scenario is when a participant enters a room and makes a casual contact with another person in the room. The greeting can be a "hi" represented by a hand waving gesture. "Participant 2"

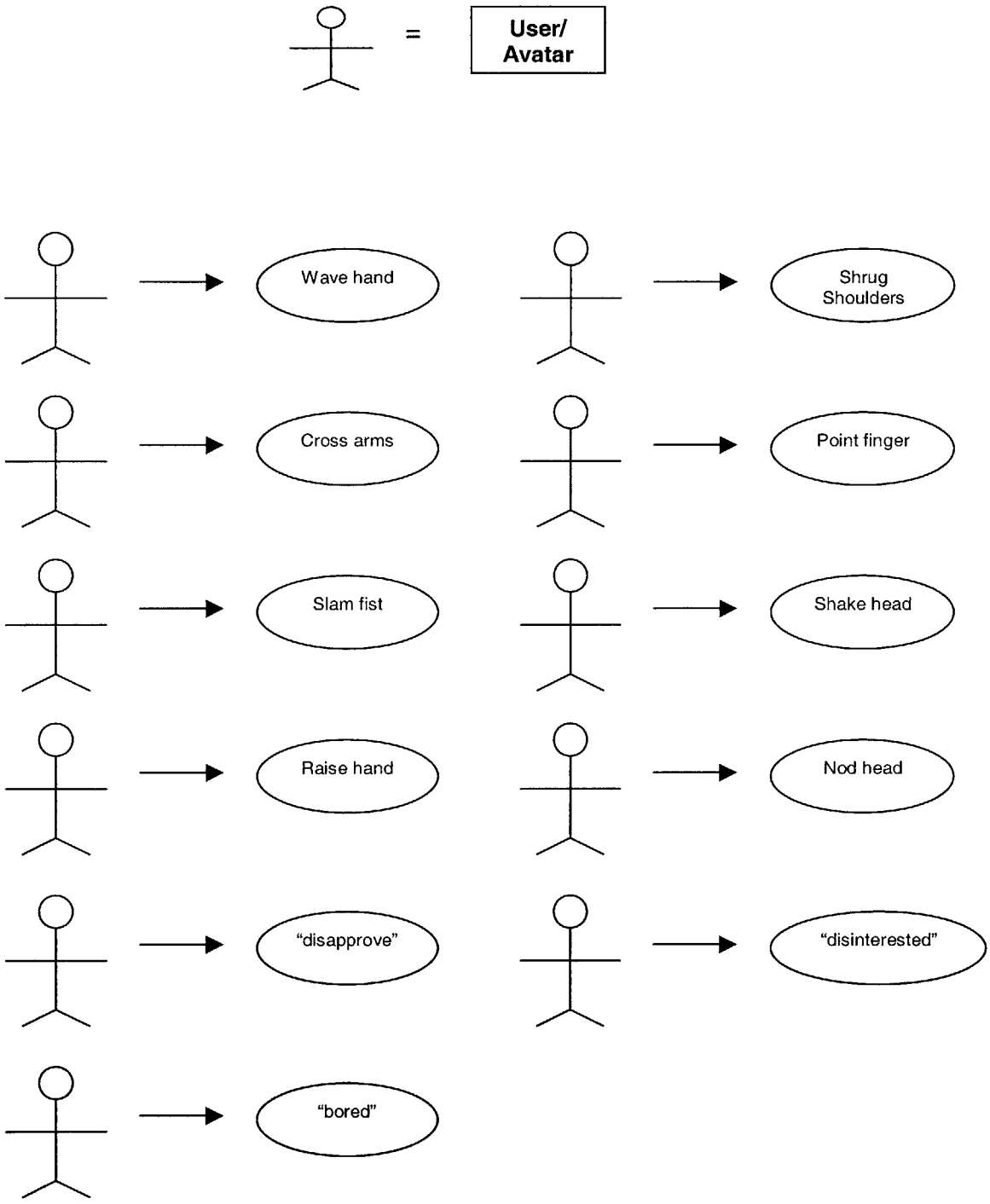


Figure 6-1: Use Case Diagrams for Basic Gesture Motions

perceives this and may also greet "participant 1" with a wave as well. The second scenario takes place in a collaborative meeting session. A participant in the meeting has several options to express himself during the meeting, even when someone else has the control of the floor. Since gesturing is not intrusive or takes the floor away from someone else, it serves as a convenient way to express oneself to others in the group. For example, in order to get attention from others one may raise his hand. The participant can also show his mood or attention state through the crossing his arm (attentively listening), evoking the "bored" gesture, or by displaying the "disinterested" gesture. The third scenario is also during a collaborative meeting session where an idea is proposed. To show an immediate informal response to the proposed idea, the participant may choose to nod his head in agreement to the proposal, or shake his head to disagree. The final scenario is applicable in any one to one situation, whether it be formal or casual form of contact. The situation is as follows: "participant 2" asks "participant 1" a question. If "participant 1" happens not to know the answer to the particular question, he may simply respond by shrugging his shoulder, indicating that he does not know.

Other scenarios are certainly possible where emblematic gesture can be used to achieve simple, direct channel of communication by means of increasing the freedom of the users to express themselves. Gesturing becomes a natural part of social interaction in virtual environments this way, as it is in face-to-face human interactions.

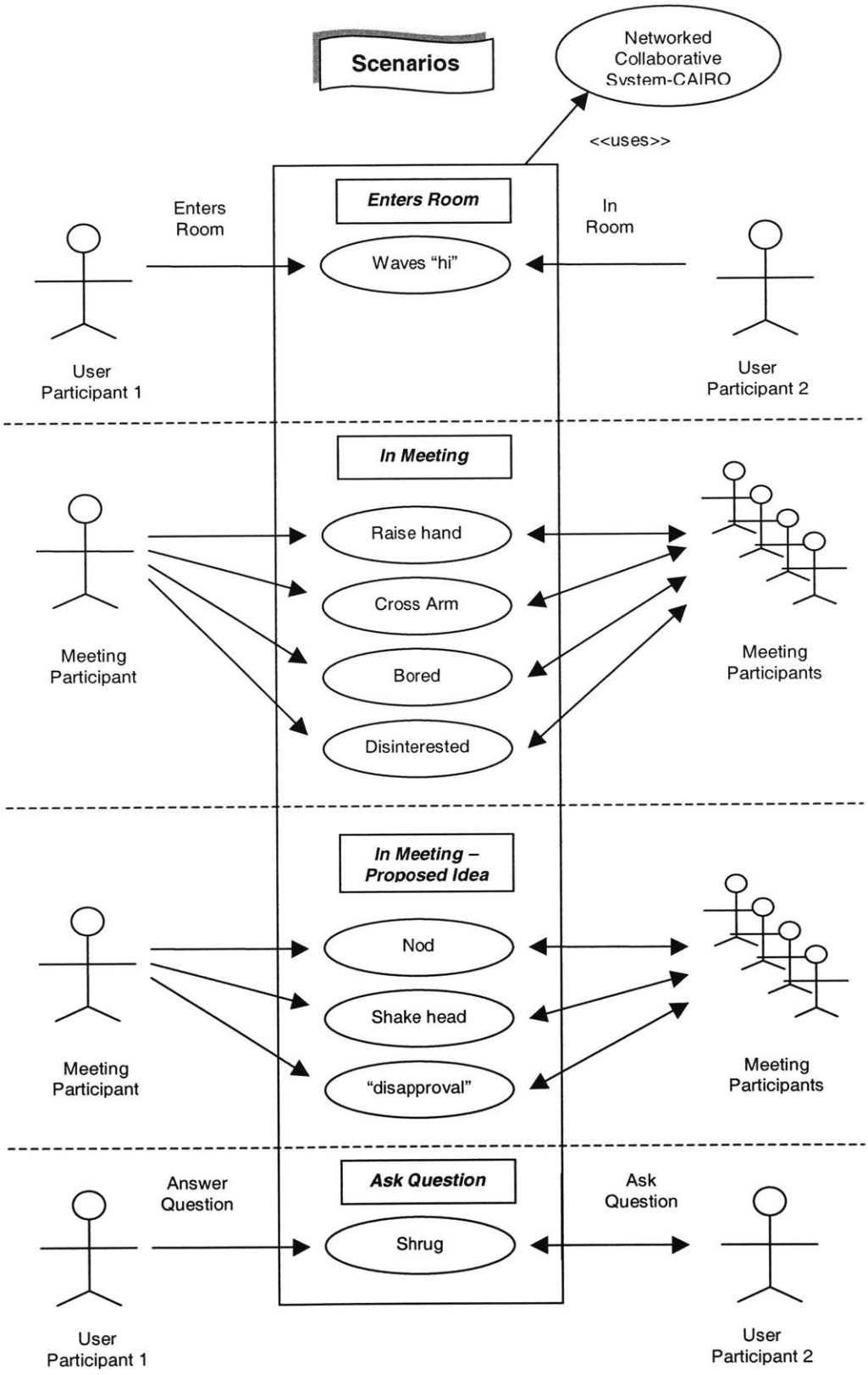


Figure 6-2: Use Case Diagrams for Situational Environments

6.1.4 User Interface

The user interface would be that resembling a Java applet with tabs and/or buttons that will be constructed using the Java AWT. In addition to the VRML canvas/browser, the user will have a navigation toolbar, and then the following set of push buttons to activate and generate the gesture animations on the avatar:

- Wave
- Shrug
- Cross Arms
- Point
- Fist
- Shake Head
- Raise Hand
- Turned Head
- Boredom
- Disapproval
- Nod

The basic user interface window will be as shown in Figure 6-3. Certain elements of the user interface should be noted. The canvas that will display the avatars and the virtual environment that the user will be immersed in is the large component of the displayed window. The buttons that will invoke the gesture motions of the avatars are below the canvas, organized by situational uses for each specific gesture. For example, a gesture that would most commonly be used during a formal meeting session, such as raising hand to get attention, will be placed under the "formal" tab, while a waving hi gesture would fall under that category of "informal". Although the gesture buttons may be organized in this manner, this arrangement certainly would not prevent the user from using any gesture in any tab at any time he desires to.

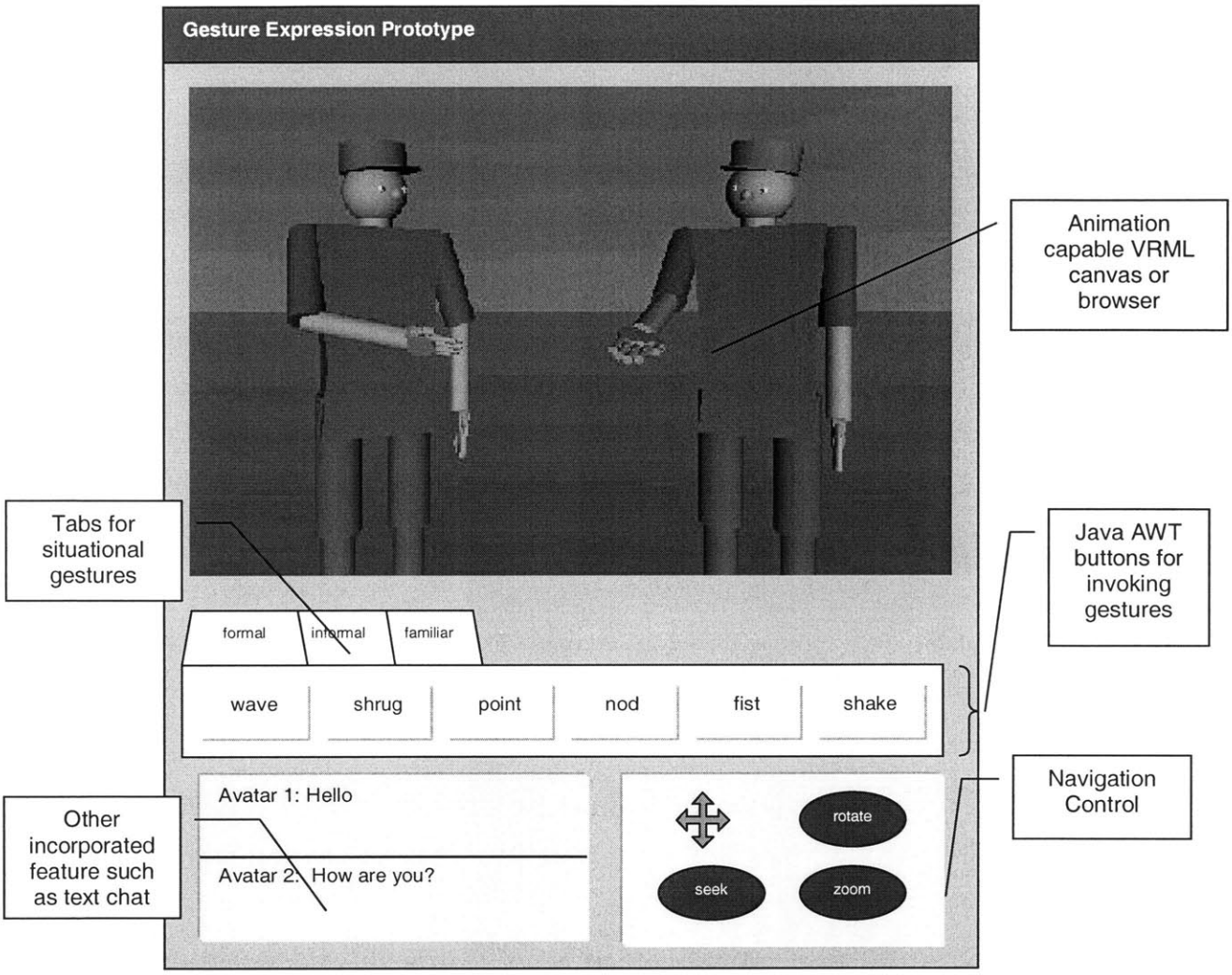


Figure 6-3: User Interface

Other elements of the three-dimensional interface should be also included in the user interface. For example, a text chat that is implemented should be placed in the same window to allow users to interact and communicate with each other. Lastly, navigation control should also be placed within the window to give the user the ability to move around in the virtual environment. Common navigation features, such as rotate, pan, zoom, and seek should be included, and are shown in Figure 6-3 in the bottom right-hand corner of the window.

6.2 Standardized VRML avatar

In applying the concepts of social interaction and collaboration in online virtual environments, it is necessary to consider the actual representation of the user as avatars in this scenario. Specifically, avatars will be used as the primary mode of representation by the user in this environment, and relevantly, it would be the avatars that will display gesture movements which will be seen and interpreted by the other users. Avatars would be human-like in order to recreate a serious, collaborative setting. Achieving that goal will require the creation of libraries of interchangeable humanoids, as well as techniques that make it easy to create new humanoids and animate them in various ways.

The following sections specify a standard way of representing humanoids in VRML 97 as proposed by the VRML Humanoid Animations Working Group (1997). These sections will detail the VRML avatar standard specifications and is based on the Specification for a Standard VRML Humanoid Version 1.0. This standard will allow humanoids created using authoring tools from one system to be animated using tools from another, and have a common structure behind all the avatars that may be used. Since the animation technique and structure of the various avatars is same, it would make sense to adapt a standard as this one to facilitate implementation of gesture animations for various number of avatars that the user may choose to use. VRML humanoids can be animated using keyframing, inverse kinematics, performance animation systems and other techniques.

6.2.1 Overview

The human body consists of a number of *segments* (such as the forearm, hand and foot) which are connected to each other by *joints* (such as the elbow, wrist and ankle). In order for an application to animate a humanoid, it needs to obtain access to the joints and alter the joint angles. The application may also need to retrieve information about such things as joint limits and segment masses.

Each segment of the body will typically be defined by a mesh of polygons, and an application may need to alter the locations of the vertices in that mesh. The application may also need to obtain information about which vertices should be treated as a group for the purpose of deformation.

A VRML Humanoid file contains a set of *Joint* nodes that are arranged to form a hierarchy. Each Joint node can contain other Joint nodes, and may also contain a *Segment* node which describes the body part associated with that joint. Each Segment can also have a number of *Site* nodes, which define locations relative to the segment. Sites can be used for attaching clothing and jewelry, and can be used as end-effectors for inverse kinematics applications. They can also be used to define eyepoints and viewpoint locations.

Each Segment node can have a number of *Displacer* nodes, which specify which vertices within the segment correpond to particular feature or configuration of vertices. The file also contains a single *Humanoid* node which stores human-readable data about the humanoid such as author and copyright information. That node also stores references to all the Joint, Segment and Site nodes, and serves as a "wrapper" for the humanoid. In addition, it provides a top-level Transform for positioning the humanoid in its environment.

Keyframe animation sequences can be stored in the same file, with the outputs of various VRML interpolator nodes being ROUTEd (See Section 6.3.2) to the joints of the body (Scott 1996). Alternatively, the file may include Script nodes which access the joints directly. In addition, applications can obtain references to the individual joints and segments from the Humanoid node. Such applications will typically animate the humanoid by setting the joint rotations.

6.2.2 The Nodes

In order to simplify the creation of humanoids, several new node types are introduced. Each node is defined by a PROTO (Sonstein, 1996). The prototype element (PROTO) of VRML is used to create a customized node that can be treated like a black box in a scene (Sonstein, 1996). PROTO declarations follow the idea of objects of a classes in C++ and Java programming languages. Once declared and defined, a PROTO can be used like any of the standard VRML nodes in the scene. PROTO's can also be used to provide a standard method that allows writers to incorporate their own custom nodes without breaking adherence to the specification. (Roehl, 1997) The basic implementation of all the nodes is very straightforward, yet each provides enough flexibility to allow more advanced techniques to be used.

6.2.2.1 The Joint Node

Each joint in the body is represented by a Joint node. The most common implementation for a Joint will be a Transform node, which is used to define the relationship of each body segment to its immediate parent. However, that's just one possible implementation -- humanoid authors are free to implement the Joint node however they choose. In particular, some systems may choose to use a single polygonal mesh to represent a humanoid, rather than having a separate IndexedFaceSet for each body segment. In such a case, a Joint would be responsible for moving the vertices that correspond to a particular body segment and all the segments descended from it. Many computer games use such a single-mesh representation in order to create smooth, seamless figures. The Joint node is also used to store other joint-specific information. In particular, a joint *name* is provided so that applications can identify each Joint node at runtime.

In addition, the Joint node may contain hints for inverse-kinematics systems that wish to control the H-Anim figure. These hints include the upper and lower joint limits, the orientation of the joint limits, and a stiffness/resistance value. Note that these limits are not enforced by any mechanism within the scene graph of the humanoid, and are

provided for information purposes only. Use of this information and enforcement of the joint limits is up to the application.

The Joint PROTO looks like this:

```
PROTO Joint [
    exposedField     SFString     name                "" 
    exposedField     SFVec3f      center              0 0 0
    exposedField     SFRotation   rotation            0 0 1 0
    exposedField     SFVec3f      scale               1 1 1
    exposedField     SFRotation   scaleOrientation    0 0 1 0
    exposedField     SFVec3f      translation         0 0 0
    exposedField     MFFloat      ulimit              []
    exposedField     MFFloat      llimit              []
    exposedField     SFRotation   limitOrientation    0 0 1 0
    exposedField     MFFloat      stiffness           [ 1 1 1 ]
    exposedField     MFNode       children            []
]
```

Notice that most of the fields correspond to those of the Transform node. This is because the typical implementation of the Joint PROTO will be:

```
{
    Transform {
        translation IS translation
        rotation IS rotation
        scale IS scale
        scaleOrientation IS scaleOrientation
        center IS center
        children IS children
    }
}
```

Other implementations are certainly possible. The only requirement is that a Joint be able to accept the events listed above.

The *center* exposedField gives the position of the Joint's center of rotation, relative to the root of the overall humanoid body description. Note that the *center* field is not intended to receive events. The locations of the joint centers are available by reading the *center* fields of the Joints.

Since the locations of the joint centers are all in the same coordinate frame, the length of any segment can be determined by simply subtracting the locations of the joint centers. The exception will be segments at the ends of the fingers and toes, for which the Site locations within the Segment must be used.

The *ulimit* and *llimit* fields of the Joint PROTO specify the upper and lower joint rotation limits. Both fields are three-element MFFloats containing separate values for the X, Y and Z rotation limits. The *ulimit* field stores the upper (i.e. maximum) values for rotation around the X, Y and Z axes. The *llimit* field stores the lower (i.e. minimum) values for rotation around those axes.

The *limitOrientation* exposedField gives the orientation of the coordinate frame in which the *ulimit* and *llimit* values are to be interpreted. The *limitOrientation* describes the orientation of a local coordinate frame, relative to the Joint center position described by the *center* exposedField.

The *stiffness* exposedField, if present, contains values ranging between 0.0 and 1.0 which give the inverse kinematics system hints about the "willingness" of a joint to move a particular degree of freedom. For example, a Joint node's stiffness can be used in an arm joint chain to give preference to moving the left wrist and left elbow over moving the left shoulder, or it can be used within a single Joint node with multiple degrees of freedom to give preference to individual degrees of freedom. The larger the *stiffness* value, the more the joint will resist movement.

Each Joint should have a DEF name that matches the *name* field for that Joint, but with a distinguishing prefix in front of it. That prefix can be anything, but must be the same for all the Joints in a particular humanoid. The distinguishing prefix is useful in the case of static routing to the Joints of multiple humanoids in the same file. If only a single humanoid is stored in a file, the prefix should be "hanim_" (for Humanoid Animation). For example, the left shoulder would have a DEF name of "hanim_l_shoulder".

The DEF name is used for static routing, which would typically connect OrientationInterpolators in the humanoid file to the joints. The *name* field is used for identifying the joints at runtime, since the DEF names would not necessarily be available. It will occasionally be useful for the person creating a humanoid to be able to add additional joints to the body. The body remains humanoid in form, and is still generally expected to have the basic joints described later in this document. However, they may be thought of as a minimal set to which extensions may be added (such as a prehensile tail).

6.2.2.2 The Segment Node

Each body segment is stored in a Segment node. The Segment node will typically be implemented as a Group node containing one or more Shapes or perhaps Transform nodes that position the body part within its coordinate. The use of LOD nodes is recommended if the geometry of the Segment is complex.

```
PROTO Segment [
    exposedField    SFString    name              ""
    exposedField    SFVec3f     centerOfMass      0 0 0
    exposedField    SFVec3f     momentsOfInertia  1 1 1
    exposedField    SFFloat     mass              0
    exposedField    MFNode      children          [ ]
    exposedField    SFNode      coord             NULL
    exposedField    MFNode      displacers        [ ]
    eventIn         MFNode      addChildren
    eventIn         MFNode      removeChildren
]
```

This will typically be implemented as follows:

```
{
    Group {
        children IS children
        addChildren IS addChildren
        removeChildren IS removeChildren
    }
}
```

The *mass* is the total mass of the segment, and the *centerOfMass* is the location within the segment of its center of mass.

Humanoids that are modeled as a continuous mesh will still have Segment nodes, in order to store per-segment information. In such a case, the Segment wouldn't necessarily contain any geometry, though it should still be a child of a Joint node.
For Segments that have deformable meshes, the *coord* field should contain a Coordinate node that is utilized with the USE tag in the IndexedFaceSet for the Segment. The Coordinate node should be given the same name DEF name as the Segment, but with a "_coords" appended (e.g. "skull_coords").

6.2.2.3 The Site Node

A Site node serves three purposes. The first is to define an "end effector" location which can be used by an inverse kinematics system. The second is to define an attachment point for accessories such as jewelry and clothing. The third is to define a location for a virtual camera in the reference frame of a Segment (such as a view "through the eyes" of the humanoid for use in multi-user worlds).

Sites are stored within the *children* exposedField of a Segment node. The *rotation* and *translation* fields of the Site node define the location and orientation of the end effector within the coordinate frame of the Segment. The *children* field is used to store any accessories that can be attached to the segment.

The Site PROTO looks like this:

```
PROTO Site [
    eventIn         MFNode      addChildren
    eventIn         MFNode      removeChildren
    exposedField    MFNode      children          []
    exposedField    SFString    name              ""
    exposedField    SFVec3f     center            0 0 0
    exposedField    SFRotation  rotation          0 0 1 0
    exposedField    SFVec3f     scale             1 1 1
    exposedField    SFRotation  scaleOrientation  0 0 1 0
    exposedField    SFVec3f     translation       0 0 0
]
```

The typical implementation for a Site will be:

```
{
    Transform {
        children IS children
        addChildren IS addChildren
        removeChildren IS removeChildren
        center IS center
        rotation IS rotation
        scale IS scale
        scaleOrientation IS scaleOrientation
        translation IS translation
    }
}
```

If used as an end effector, the Site node should have a name consisting of the name of the Segment to which it's attached, with a "_tip" suffix appended. For example, the end effector Site on the right index finger would be named "r_index_distal_tip", and the Site node would be a child of the "r_index_distal" Segment. Sites that are used to define camera locations should have a "_view" suffix appended. Sites that are not end effectors should have a "_loc" suffix. Sites that are required by an application but are not defined in this specification should be prefixed with "x_".

Sites that intended to be used as attachment points for Viewpoint nodes (such as the left and right eyes) should be oriented so that they face in the direction the eye should be looking. In other words, attaching the following Viewpoint to the Site at the left eye will result in a view looking out from the humanoid's left eye:

```
Viewpoint {
    position 0 0 0
}
```

6.2.2.4 The Displacer Node

Applications may need to alter the shape of individual Segments. At the most basic level, this is done by writing to the *point* field of the Coordinate node that's found in the *coord* field of the Segment node.

In some cases, the application may need to be able to identify specific groups of vertices within a Segment. For example, the application may need to know which vertices within the skull Segment comprise the left eyebrow. It may also require "hints" as to the direction in which each vertex should move. That information is stored in a node called a Displacer. The Displacers for a particular Segment are stored in the *displacers* field of that Segment.

The Displacer PROTO looks like this:

```
PROTO Displacer [
    exposedField SFString name           ""
    exposedField MFInt32  coordIndex     [ ]
    exposedField MFVec3f  displacements  [ ]
]
```

The *name* field provides a name for the Displacer, by which it can be identified by the application at runtime. That name should also be used as the DEF name of the Displacer node itself.

The *coordIndex* field contains the indices into the coordinate array for the Segment of the vertices that are affected by the displacer. For example,

```
Displacer {
    name "l_eyebrow_feature"
    coordIndex [ 7, 12, 21, 18 ]
}
```

would mean that vertices 7, 12, 21 and 18 of the Segment form the left eyebrow.
The *displacements* field, if present, provides a set of 3D values that should be added to the neutral or resting position of each of the vertices referenced in the *coordIndex* field of the Segment. These values correspond one to one with the values in the *coordIndex* array. The values should be maximum displacements, and the application is free to scale them as needed before adding them to the neutral vertex positions. For example,

```
Displacer {
    name "l_eyebrow_raiser_action"
```

```
    coordIndex [ 7, 12, 21, 18 ]
    displacements [ 0 0.0025 0, 0 0.005 0, 0 0.0025 0, 0 0.001 0 ]
}
```

would raise the four vertices of the left eyebrow in a vertical direction. Vertex number 7 would be displaced up to 2.5 millimeters in the vertical (Y) direction, vertex number 12 would be displaced up to 5 millimeters, vertex 21 would be displaced up to 2.5 millimeters, and vertex number 18 would be displaced by just one millimeter. The application may choose to uniformly scale those displacements.

A Displacer can be used in three different ways. At its most basic level, it can simply be used to identify the vertices corresponding to a particular feature on the Segment, which the application can then displace as it sees fit. At the next level, it can be used to represent a particular muscular action which displaces the vertices in various directions (linearly or radially).

The third way in which a Displacer can be used is to represent a complete configuration of the vertices in a Segment. For example, in the case of a face, there might be a Displacer for each facial expression.
Displacers that are used to identify features should have a name with a "_feature" suffix. Displacers that are used to move a feature should be given a name with an "_action" suffix, usually with an additional pre-suffix to indicate the kind of motion (such as "l_eyebrow_raiser_action"). Displacers that correspond to a particular configuration of the vertices should have a "_config" suffix.

Note that while Displacers are most often used to control the shape of the face, they can certainly be used for other body parts as well. For example, they may be used to control the changing shape of an arm Segment as the arm flexes, simulating the effect of muscle inflation.

6.2.2.5 The Humanoid Node

The Humanoid node is used to store human-readable data such as author and copyright information, as well as to store references to the joints, segments and views and to serve as a container for the entire humanoid. It also provides a convenient way of moving the humanoid through its environment.

```
PROTO Humanoid [
    exposedField    SFString    name                "" 
    exposedField    MFString    info                [ ]
    exposedField    SFString    version             "1.1"
    exposedField    MFNode      joints              [ ]
    exposedField    MFNode      segments            [ ]
    exposedField    MFNode      sites               [ ]
    exposedField    MFNode      viewpoints          [ ]
    exposedField    MFNode      humanoidBody        [ ]
    exposedField    SFVec3f     center              0 0 0
    exposedField    SFRotation  rotation            0 0 1 0
    exposedField    SFVec3f     scale               1 1 1
    exposedField    SFRotation  scaleOrientation    0 0 1 0
    exposedField    SFVec3f     translation         0 0 0
]
```

The Humanoid node is typically implemented as follows:

```
{
  Transform {
    center           IS center
    rotation         IS rotation
    scale            IS scale
    scaleOrientation IS scaleOrientation
    translation      IS translation
    children [
      Group {
        children IS viewpoints
      }
      Group {
        children IS humanoidBody
      }
    ]
  }
}
```

The Humanoid node can be used to position the humanoid in space. Note that the HumanoidRoot Joint is typically used to handle animations within the local coordinate system of the humanoid, such as jumping or walking. For example, while walking, the

overall movement of the body (such as a swagger) would affect the HumanoidRoot Joint, while the average linear velocity through the scene would affect the Humanoid node. The *humanoidBody* field contains the HumanoidRoot node.

The *version* field stores the version of this specification that the Humanoid file conforms to. The document you're now reading describes version "1.1" of the specification. The *info* field consists of an array of strings, each of which is of the form "*tag=value*".

The following tags are defined so far:

- authorName
- authorEmail
- copyright
- creationDate
- usageRestrictions
- humanoidVersion
- age
- gender (typically "male" or "female")
- height
- weight

The HumanoidVersion tag refers to the version of the humanoid being used, in order to track revisions to the data. It is not the same as the *version* field of the Humanoid node, which refers to the version of the H-Anim specification which was used when building the humanoid.

The *joints* field contains references (i.e. USEs) of each of the Joint nodes in the body. Each of the referenced joints should be a Joint node. The order in which they are listed is irrelevant, since the names of the joints are stored in the joints themselves. Similarly, the *segments* field contains references to each of the Segment nodes of the body, the *viewpoints* field contains references to the Viewpoint nodes in the file, and the *sites* field contains references to the Site nodes in the file.

6.2.3 The Joint Hierarchy

The body is typically built as a series of nested Joints, each of which may have a Segment associated with it. For example:

```
...
DEF hanim_l_shoulder Joint {  name "l_shoulder"
  center 0.167 1.36 -0.0518
  children    [
    DEF hanim_l_elbow Joint {  name "l_elbow"
      center 0.196 1.07 -0.0518
      children        [
        DEF hanim_l_wrist Joint {  name "l_wrist"
          center 0.213 0.811 -0.0338
          children    [
            DEF hanim_l_hand Segment {  name "l_hand"
              ...
            }
          ]
        }
        DEF hanim_l_forearm Segment {  name "l_forearm"
          ...
        }
      ]
    }
    DEF hanim_l_upperarm Segment {  name "l_upperarm"
      ...
    }
  ]
}
...
```

6.2.3.1 The Body

The names of the Joint nodes for the body are listed in the following Table 6-2:

Table 6-2: Names of Body Joints (Roehl, 1999)

l_hip	l_knee	l_ankle	l_subtalar	l_midtarsal	l_metatarsal	
r_hip	r_knee	r_ankle	r_subtalar	r_midtarsal	r_metatarsal	
vl5	vl4	vl3	vl2	vl1		
vt12	vt11	vt10	vt9	vt8	vt7	
vt6	vt5	vt4	vt3	vt2	vt1	
vc7	vc6	vc5	vc4	vc3	vc2	vc1
l_sternoclavicular	l_acromioclavicular	l_shoulder	l_elbow	l_wrist		
r_sternoclavicular	r_acromioclavicular	r_shoulder	r_elbow	r_wrist		
HumanoidRoot	sacroiliac (pelvis)	skullbase				

6.2.3.2 The Hands

The hands, if present, should use the following naming convention:

Table 6-3: Names of the Joints of the Hand (Roehl, 1999)

Left Hand							
l_pinky0	l_pinky1	l_pinky2	l_pinky3	l_ring0	l_ring1	l_ring2	l_ring3
l_middle0	l_middle1	l_middle2	l_middle3	l_index0	l_index1	l_index2	l_index3
l_thumb1	l_thumb2	l_thumb3					
Right Hand							
r_pinky0	r_pinky1	r_pinky2	r_pinky3	r_ring0	r_ring1	r_ring2	r_ring3
r_middle0	r_middle1	r_middle2	r_middle3	r_index0	r_index1	r_index2	r_index3
r_thumb1	r_thumb2	r_thumb3					

As shown below in Figure 6-4, the hand of the avatar can be divided into a number of joints marked in Table 6-3 by "0" through "3" following the name of the finger. The "l_" denotes left hand, while "r_" denotes the right hand. The crosshairs shown in Figure 6-4, simply suggest possible locations for the joint centers and finger tips. It also should be noted that the movement of the "0" joint of the thumb, which is not listed in Table 6-3, is typically quite limited, and this not shown. Similar observations can be made on the other fingers as well, as the rigidity of those joints varies from finger to finger. Further details about the placement, orientation and movement of the "0" joints can be obtained from any anatomy reference text (Roehl, 1999).

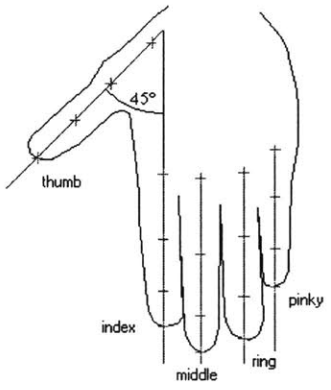


Figure 6-4: Joints of the Fingers of a Hand (Roehl, 1999)

6.2.3.3 The Face

Many humanoid implementations have made use of jointed facial structures to simulate facial expression. These work in a similar fashion to the facial parts of a ventriloquist's dummy. The following in Table 6-4 is a basic set of facial joints and segments that support this type of facial animation.

The suffix "_joint" is used here because in reality most of these features are controlled by muscle groups instead of actual joints, the exception being the temporomandibular joint. The "_joint" suffix provides a distinction between the joint name and the name of the corresponding segment.

All facial joints are children of the skullbase joint. The center of rotation of the eye and the eyelid is the geometric center of the eyeball. The eyelid rotation defaults to zero, and a positive rotation of PI radians will completely close the eyelid. The eyebrows are at zero degrees rotation by default, and can be rotated around the middle of the eyebrow. The mouth is closed when the temporomandibular joint is at zero degrees.

Table 6-4: Joints of the Face (Roehl, 1999)

l_eyeball_joint	r_eyeball_joint
l_eyebrow_joint	r_eyebrow_joint
l_eyelid_joint	r_eyelid_joint
temporomandibular	

6.2.4 Hierarchy

The complete hierarchy is shown in Figure 6-5, with the segment names listed beside the joints to which they're attached.

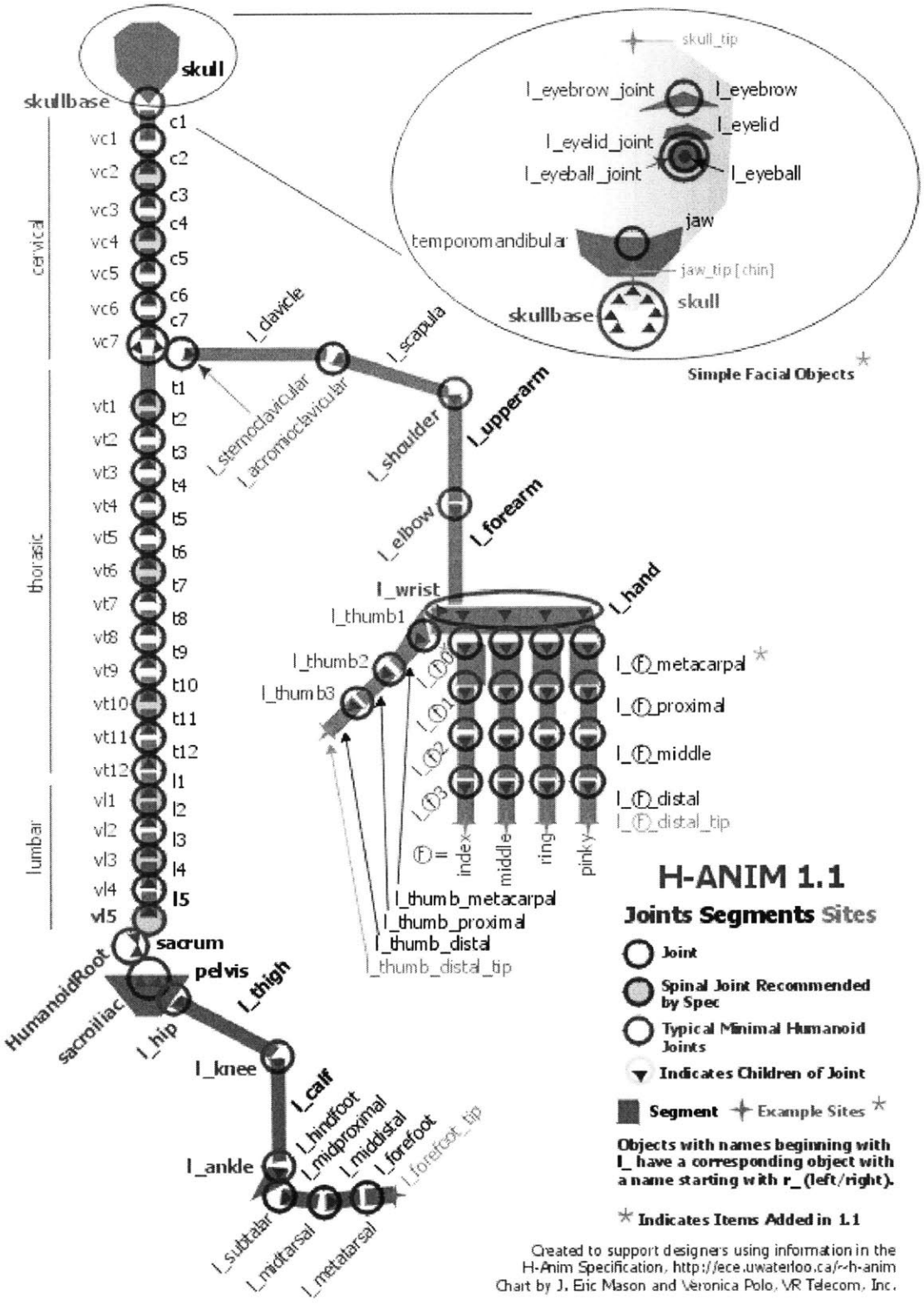


Figure 6-5: Hierarchy of H-Anim Humanoid Avatar (H-Anim Working Group, 1997)

6.3 VRML Execution Model

VRML consists of two fundamental parts: The scene description language and a mechanism to create behavior (Roehl, 1997). Behavior might be anything that dynamically changes the properties of the scene. There could be something as simple as a color changing box to an animation of a complex artificial intelligence avatar in a virtual world.

6.3.1 Events

When one node wants to pass information to another node, it creates an event. An event is something between a function call and an assignment in a programming language. It is a way of encapsulating a piece of data to pass it from one node to another. The event contains the *data* and the *timestamp*. The timestamp is an important concept in cases where networking is involved, as is the case here. It is the time at which the event was originally started. This helps the browser to keep track of the sequence of events in such an environment.

6.3.2 Routes

When a VRML scene (object), such as a humanoid avatar, is created consisting of a collection of nodes, information must be passed between them. VRML does not have a function call mechanism for passing information. Instead it involves the creation of an explicit connection between two fields of nodes. There are four different types of access types that using which the behavior of the nodes can be controlled.

Table 6-5: VRML Field Access Types

ACCESS TYPE	ACCESS AVAILABLE TO OTHER NODES
field	No external access
eventIn	Write only
eventOut	Read only
exposedField	Read and write

6.3.3 Sensors

Unlike Java where the programs throw events, in VRML we have to generate the events and pass them around. We do this once we have a series of nodes connected together by routes. The sensor group of nodes, sense some sort of non-VRML input and generate a VRML event which is then passed around the scene. There are three basic types of sensors namely *user input, visibility* and *time* sensors. The Time sensors and the user input sensors are very important for the product that is being developed. The Touch sensors are a kind of user input sensors which are very important to define a boundary for the environment, that restricts objects (avatars) from navigating out of the environment. For instance, using the touch sensor, we could introduce a condition by which the avatar does not leave its chair once it is seated on it, until the user logs out of the meeting.

VRML time is based on seconds. Time zero corresponds to the UNIX standard of midnight GMT, January 1, 1970. All time is absolute. There is no relative time in VRML at all. The time-dependent nodes share four properties: *startTime*, *stopTime*, *isActive*, and *loop*. Start and stop times control when the node is to start and stop its output. Loop is a boolean that indicates whether the node should loop its output. IsActive indicated whether the node is playing (active) at that time. In generating gesture movement animation, time dependent node would be implemented to define the length of the movement and to control the timing of the proceeding body positions.

6.3.4 Interpolators

Interpolators are one of the most useful nodes in VRML for performing animations. These nodes are the glue between time and getting an object to do something. Interpolators take a time input and turn it into something that performs animation in the scene, for example an avatar movement.

All interpolators have the same basic structure. Each key (the *keyValue* field) has a list of key values and a list of corresponding output values. Each key must have an

associated value that range from zero to one. The interpolator receives input through the *set_fraction* eventIn. The interpolator then checks this value against the keys, calculates the output value, and passes out in the *value_changed* eventOut.

6.3.5 Building an Animation

The building of the animation involves the existence if the basic scene layout. It also consists of the object to be animated and a reference point with respect to which the animation is done. There will be a set of transforms that define the animation to be executed on the object of interest. Then we need to layout the path that the object will take which is done using a set of interpolators. There are different kinds of interpolations based on the type of animation that is being done. Finally we need to provide a time sensor to drive the animation. ROUTEs will also need to be added at the end to pass events around.

6.4 Animating the Avatar

The structure of the avatar that will be used in the gesture expression would be following the VRML Humanoid specifics as a means to standardize the component structure of each avatar that the user may use. With a standardized avatar model, it would be easier to implement the gesture animation processes for each of the different avatars that may be used in the prototype.

In addition to the geometries specified in the VRML (.wrl) file, it will also contain information about the various forms of animations that will be implemented for the avatar to express the desired gestures. The geometries of the avatar is modeled using the standard body parts-joins-names as provided by the H-Anim specs. As described in Section 6.3, VRML Execution Model, a set of interpolators and ROUTEs are used for each distinct gesture movement to pass events around within the VRML scene.

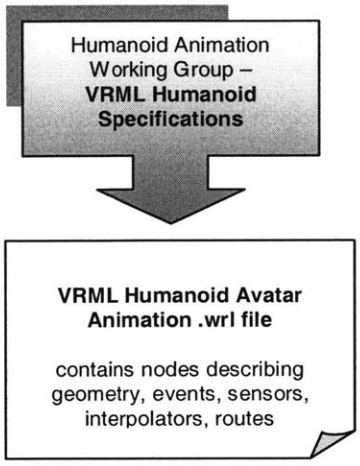


Figure 6-6: Standardization of VRML avatar format

An animation sequence can be understood as a cycle with a beginning (the first frame) and an end (the last frame) and everything else in between. In order to build animations of this type, called keyframe animation, a set of keyframes or the specific moments in an animation cycle needs to be defined. Then, interpolators are used to compute all the intermediary positions that are necessary for a smooth animation in function of the fraction of the animation cycle computed. To do this, the ouput of the *fraction_changed* field of TimeSensor as input for an interpolator engine. The *set_fraction* eventIn receives an event that can be any kind of floating point number. It usually is a number between 0 and 1 produced by a TimeSensor's *fraction_changed* field. For each critical fraction of the whole animation cycle, the fraction is added to the array of *key*s and for each key, an interpolator-specific *keyValue* must be specified. Finally, the interpolator will produce the interpolated *value_changed* eventOuts.

The basic structure of this animation method is to be used for the gesture animations. Each distinct animation sequence is encapsulated in an *Animation* node, these nodes are gathered into an *Animations* node. The object to be animated has a "Humanoid" node, or equivalent, specifying the names of Joints to animate. The code structure of the animation for the suggested gestures of the prototype would follow this format shown in Appendix A. Each joint of the avatar is defined for each animation group, i.e., for each

gesture movement. The humanoid node is then defined by specifying the joints that are used. The joints are then connected through eventIn's and eventOut's via ROUTEs to pass around the events and geometry transformations that are associated with the animation of the avatar. While the specific geometries of the avatar aren't shown, this is inconsequential because it is assumed that they are defined elsewhere in the VRML file.

The various types of gesture animation that needs to be invoked is grouped under one subgroup under the root of *Animations* group. The interpolators for each gesture is defined within these *Animation* subgroups. The joints of the corresponding relevant parts of the H-Anim humanoid avatar is listed, followed by passing information between the nodes using ROUTE --- TO --- with the values in the fields such as *value_changed* and *set_fraction*.

Simplified, the code structure for the gesture animation follows this basic route/events model in Figure 6-7 (Schneider, 1998):

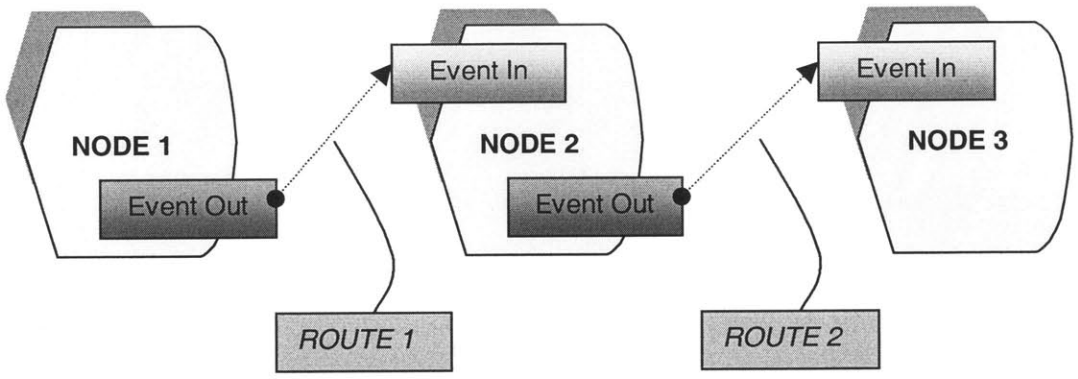


Figure 6-7: ROUTE/event model in VRML execution

6.5 Java-VRML Interface

As mentioned in Section 6.1.1, the architecture of the prototype implementation is to implement the avatar geomoetries and gesture animations in VRML 97 and use Java to

interface the user interface of the gesture behavior control pieces with the VRML avatar components. The basic user interface components, such as the user navigation control of the avatar and the gesture invoking commands, will be implemented through Java code that interacts with the VRML file which contains the information about the avatar as well as the prescribed gesture animation sequences.

In this architecture model, there is a clear separation of animated geometry (VRML) and gesture behavior control (Java). From the user's control interface enabled by Java, the VRML avatar would then implement the set of basic gesture behaviors such as shrug and wave, as detailed in Section 6.1.2. The Java module would take care of communicating with the user, the virtual environment including avatar and scene, as well as with other clients in the system to coordinate a collaborative interaction environment. This model of the proposed prototype as explained above is illustrated in Figure 6-8.

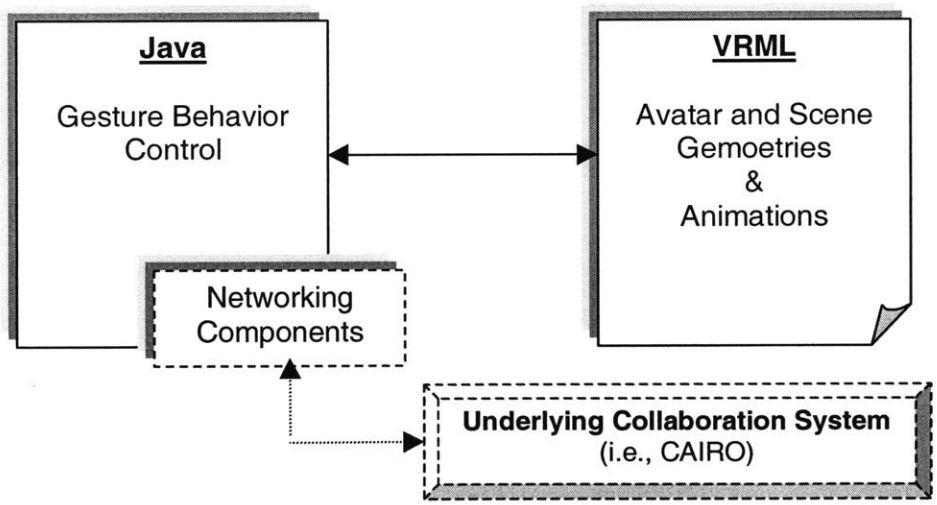


Figure 6-8: VRML and Java Components of the Prototype Design

It is clarified here that this design will not discuss the aspects of networking with other clients of the virtual collaborative system as indicated by the dashed line

components of Figure 6-8. One of the underlying assumptions of this design was that it would be implemented on top of an existing system such as CAIRO which would already have and support networking and other functions to enable collaboration in a virtual environment.

Java can be used with VRML in two ways. VRML has a Script node and Java can be used as the scripting language for that node. The Java script interacts with the VRML world through the Java Script Authoring Interface (JSAI) (Marrin, et al., 1997). The JSAI allows Java to send events to other nodes in the VRML world, create new scene components, and query for information about the scene (Marrin, 1997). The Java script receives events from other nodes, which stimulates it into execution, where it can perform algorithms, or utilize other Java packages (such as the Network package). The script can then send the results of this execution to other nodes as events. This allows Java to provide complex behaviors to the objects in a scene. A good example of this is a scene in which Java performs complex physics operations to control the waving of a flag or the bouncing of a ball.

Java can also control a VRML world externally, using the External Authoring Interface (EAI). The EAI allows one or more VRML worlds to be added to a Java applet or application. This interface is designed to allow an external environment to access nodes in a VRML scene using the existing VRML event execution model that was discussed in Section 6.3. In this model, an eventOut of a given node can be routed to an eventIn of another node. When the eventOut generates an event, the eventIn is notified and its node processes that event. The EAI allows much of the same functionality as the JSAI. You can send events to VRML nodes, create new nodes, and query the VRML world about its state and characteristics. But rather than being stimulated by an event coming into a Script node, the EAI allows a currently running Java applet to control a VRML world, just like it would control any other media. Examples would include, pressing an AWT widget under control of the Java applet/application could cause an event to be sent to the VRML world which would change the color of a sphere, the height

of a bar graph, the price readout of a stock, or in the case of this thesis, invoke a gesture animation.

Because of EAI's model of allowing control from outside of the VRML scene, for example through an AWT widget placed in an application containing the VRML, it is appropriate to use EAI as the interface model in designing the gesture expression prototype.

6.5.1 External Authoring Interface

The communication between the VRML world containing the avatar and an external environment an interface between the two is provided by *External Authoring Interface.* and it allows an external program to communicate with a VRML scene. Essentially, to Java, the EAI is a set of classes with methods that can be called to control the VRML world. To VRML the EAI is just another mechanism that can send and receive events, just like the rest of VRML.

The EAI adapts this interface by giving access to the top-level named nodes in the system to the external environment. The interface then mimics the access of the Script node with a reference to the named node. To facilitate the sending of events *to* the external environment, a new mechanism is made available. To receive notification when an event is sent to an eventOut, the external environment registers interest in it. Because the EAI model is NOT language dependent, this interface can be applied to any programming language. For this particular design, however, the Java programming language is considered.

The following EAI framework that the prototype design follows is based on Proposal for a VRML 2.0 Informative Annex - External Authoring Interface Reference by Chris Marrin (1997).

6.5.1.1 Conceptual Framework

Conceptually, the External Authoring Interface allows 4 types of access into the VRML scene:

1. Accessing the functionality of the browser interface.
2. Sending events to eventIns of nodes inside the scene.
3. Reading the last value sent from eventOuts of nodes inside the scene.
4. Getting notified when events are sent from eventOuts of nodes inside the scene.

The EAI operates as a set of classes and its methods that extend the Java platform for access to one or more VRML scenes within this environment. An applet/application that embeds the VRML canvas or browser, as well as all the user's control interface is written in Java. The applet/application then uses the EAI interface to access and control the VRML scene.

The concepts of EAI are similar to those of Script Authoring Interface, which provides the interface for languages supported by the VRML Script node. There are two conceptual differences between the two. The first has to do with obtaining a node reference through which eventIns and eventOuts can be accessed. When creating a VRML file a **Script** node (and therefore the script it contains) can get a node reference with the USE. Since the external environment has no implicit access to this mechanism an explicit method is provided to get the reference from its DEF name string. EventOut notification is also conceptually different since creating a ROUTE is not possible between the VRML scene and the external environment. The application must create a method to be called when the eventOut occurs. This method is registered with an eventOut of a given node. When the eventOut generates an event the registered method is called.

6.5.1.2 The Browser Object

The application communicates with the VRML world by first obtaining a reference to a browser object. This allows the application to uniquely identify a particular VRML avatar or object in an environment where multiple scenes are available.

A Java application communicates with a VRML world by first obtaining an instance of a **Browser** or **CBrowser** class, both of which implement the **IBrowser** interface. The **IBrowser** interface is the Java encapsulation of the VRML scene. It contains the entire Browser Script Interface as well as a **getNode()** method which returns a **Node** when given a DEF name string. The **IBrowser** class also contains **beginUpdate( )** and **endUpdate( )** methods to allow grouping of events to be sent to the scene.
An instance of an existing scene is obtained by instantiating a **Browser** class. This class is used when the application is embedded in another environment (such as a HTML browser), which has already instantiated one or more VRML scenes. This class may be instantiated in two ways (Marrin, 1997):

```
public                    Browser(Applet                    pApplet);
public Browser(Applet pApplet, String frameName, int index);
```

The **Applet** class is a standard Java class which is the encapsulation of the application in the environment. The first form associates the first or only VRML scene with the instantiation. The second form is more general and allows a *frameName* string identifying the container owning the desired VRML scene to be specified. Passing NULL or the empty string selects the first or only container in the environment. Additionally, an *index* of the VRML scene within the container can be specified. This is used in environments where a given container can own more than one sub-component. If this is not the case, a value of 0 shall be specified.

To create a new instance of a VRML scene, the **CBrowser** class is used. This class implements the **Browser** interface and also extends the AWT **Component** class. Therefore it can have all the capabilities of accessing the scene, plus it can be added to an

AWT **Container** class. When creating an instance of **CBrowser**, the **replaceWorld( )** or **loadURL( )** methods of the **Browser** class are used to add the initial scene.

6.5.1.3 Initialize and Shutdown

Upon instantiation of a browser class, the scene may not yet be loaded. When loading is complete, but before any events are processed or the scene is displayed, the **initialize( )** method is called. This method may be subclassed to receive notification that the scene has been loaded and nodes can be accessed using the **getNode( )** method. Similarly, when the scene is replaced or unloaded (e.g., when the application has been terminated) the **shutdown( )** method is called. Once this method is called, further access to the current scene shall produce undefined results. In the case where one scene is unloaded and another is loaded [e.g., when **replaceWorld( )** is called, **shutdown( )** is called to signal the unloading of the previous scene, then **initialize( )** is called to signal the loading of the new scene].

Node Access

Once an instance of the browser is obtained and a reference established, it can be used to gain access to specific nodes in the VRML world currently contained by the browser. This is done using the **getNode( )** method of the **Browser** class, which is passed the DEF name of the desired node and returns a reference to that node. This reference can then be used to access eventIns and eventOuts of that node, or in calls to the Browser Script Interface.

The Browser Script Interface

All methods in the Browser Script Interface are available to an EAI application. The interface is the same as for Script node scripts. Any node reference obtained with the **getNode( )** method can be used in these methods.

Sending events

Once a node reference is obtained, all its eventIns are accessible using the **getEventIn( )** method. This method is passed the name of the eventIn and returns a reference to an **EventIn** instance, if an eventIn with that name is found. ExposedFields can also be accessed, either by giving a string for the exposedField itself (such as "*translation*") or by giving the name of the corresponding eventIn (such as "*set_translation*").

Once an instance of the desired **EventIn** is obtained, an event can be sent to it by calling any of various methods related to the eventIn reference. However, **EventIn** has no methods for sending events. It must first be cast to the appropriate eventIn subclass, which contains methods for sending events of the given type.

Sending an eventIn to a VRML scene containing this node will take on the form:

```
DEF Mover Transform { ... }
```

Here is the Java code for sending an event to change the *translation* field (assume *browser* is the instance of a **Browser**, **ABrowser**, **or CBrowser** class gotten from a previous call) (Marrin, 1997):

```
Node mover = browser.getNode("Mover");
  EventInSFVec3f translation =
              (EventInSFVec3f) mover.getEventIn("set_translation");

  float value[3] = new float[3];
  value[0] = 5; value[1] = 0; value[2] = 1;
  translation.setValue(value);
```

In the above example, the translation value (5, 0, 1) is sent to the *translation* field of the **Transform** node.

Reading eventOut values

Once a node reference is obtained, all its eventOuts are accessible using the **getEventOut( )** method. This method is passed the name of the eventOut and returns a reference to and EventOut instance, if an eventOut with than name is found. ExposedFields can also be accessed, either by giving a string for the exposedField itself (such as *"translation"*) or by giving the name of the corresponding eventOut (such as *"translation_changed"*).

Once an instance of a desired **EventOut** is obtained, two operations can be performed. The current value of the eventOut can be retrieved, and a callback can be setup to be invoked whenever the eventOut is generated. **EventOut** does not have any methods for getting the current value so it must be cast into the appropriate eventOut subclass type, which contains appropriate access methods.
Using the eventIn example above, the current value of the translation field can be read like this (Marrin, 1997):

```
float current[] = ((EventOutSFVec3f)
          (mover.getEventOut("translation_changed"))).getValue();
```

The array *current* now contains 3 floats with the x, y, and z components of the current translation value.

Notification of eventOut changes

To receive notification when an eventOut is generated from the scene, the applet must first subclass the **EventOutObserver** class, implementing the **callback()** method. Next the **advise()** method of **EventOut** is passed the **EventOutObserver**. Then whenever an event is generated for that eventOut, the **callback()** method is executed and is passed the value and timestamp of the event. The **advise()** method is also passed a user defined object. This value is passed to the **callback()** method when an event is generated and can be used by the application author to pass user defined data to the callback. It allows a

single **EventOutObserver** subclass to handle events from multiple sources. It is a subclass of the standard java **Object** class so it can be used to hold any data.

Using the above example again, the applet can get notified when the translation field of the Transform is changed like this (Marrin, 1997):

```
public class MyObserver implements EventOutObserver {
    public void callback(EventOut value,
                         double timeStamp,
                         Object data)
    {
        // cast value into an EventOutSFVec3f and use it
    }
}

...

MyObserver observer = new MyObserver;
mover.getEventOut("translation_changed").advise(observer, null);
```

6.5.1.4 Interface Hierarchy

The Java implementation of the External Authoring Interface is specified in three Java packages: `vrml.external`, `vrml.external.field`, and `vrml.external.exception` which need to be included in the Java file. All of the members of package `vrml.external.exception` are classes derived from `java.lang.RuntimeException`; the rest of the members of the packages are specified as interfaces (with the exception of `vrml.external.field.FieldTypes`, which merely defines an integer constant for each EventIn/EventOut type). This allows the compiled Java applet to be used with any VRML browser's External Authoring Interface implementation. The hierarchy for the three packages is shown in Appendix B.

6.6 Prototype Model

Following the EAI interface framework to connect the VRML scene (i.e., avatar) with Java, Figure 6-9 shows the layout of the prototype. The VRML file will contain the

avatar geometries as well as their animation sequences. Java AWT components will be used to create the user interface that will control the avatar gesture invocation. These two components will be interfaced through the External Authoring Interface framework which has been outlined in Section 6.5. Of course, all the components will be wrapped together as an applet or an application that will embed these components and then display the results on a VRML canvas or browser.

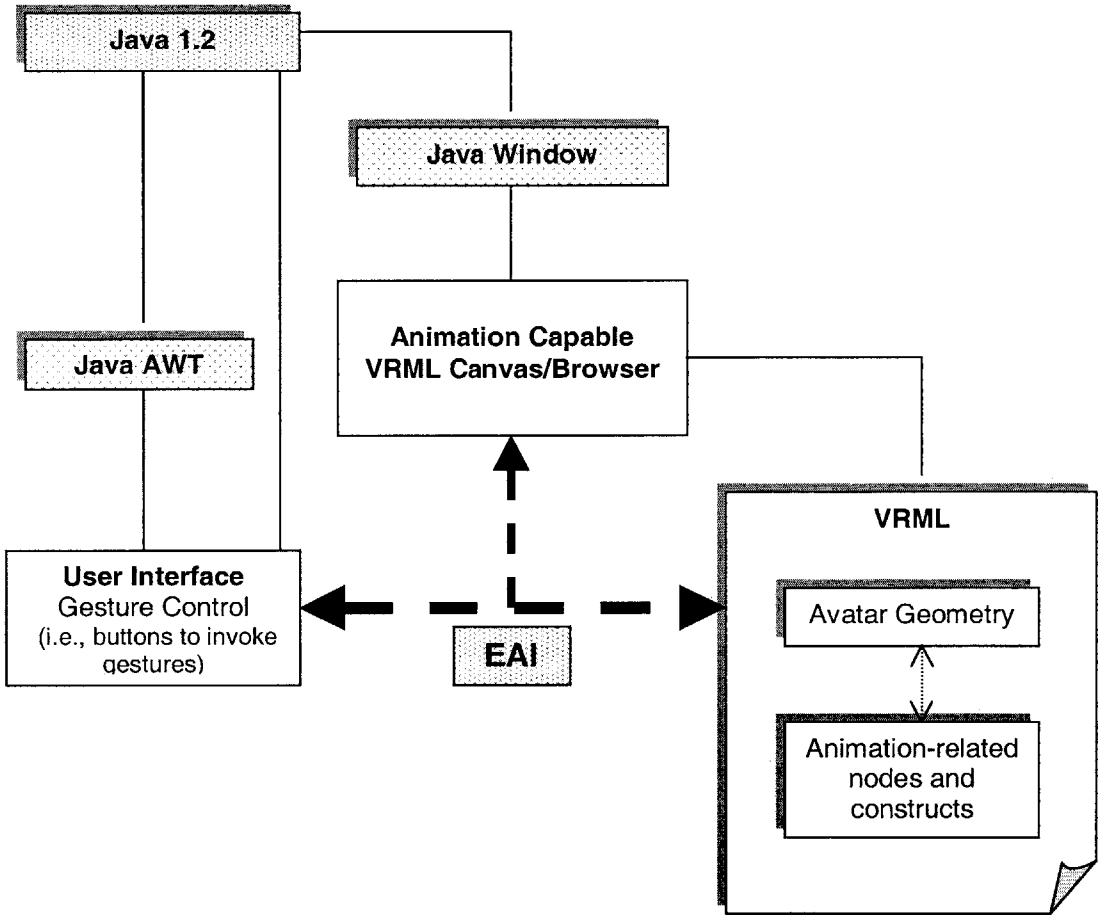


Figure 6-9: Prototype Design Model

The flow of the prototype can be summarized as follows, as illustrated in Figure 6-10. First, a handle to the VRML canvas is made so that the Java applet/application can

communicate with it to receive and send messages. Second, the scene (i.e., avatar) is initialized and loaded into the canvas. Because there exists a reference or handle to the canvas, the VRML object may be referenced to from Java code. Third, the relevant nodes of the scene are accessed. Fourth, events are sent to the relevant nodes to trigger an effect. Specifically, an event is sent to invoke the animation nodes and constructs that has been implemented in the VRML file, which leads to the chain of events that result in the animation of the avatar as it was modeled in Section 6.4. Once the animation cycle is completed, another event is sent out from the VRML scene which the Java portion listens for to know what the current state is.

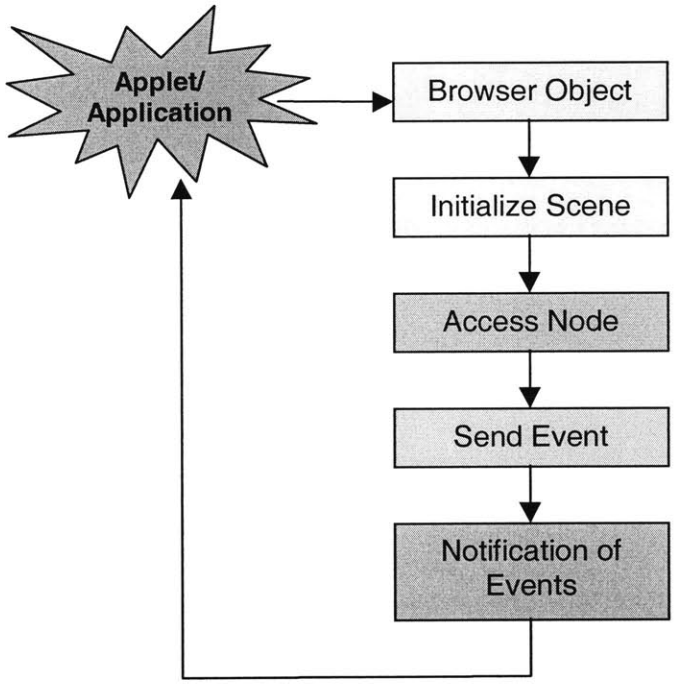


Figure 6-10: Event Flow Layout

6.7 Summary

The gesture expression prototype was designed to be an enhancing component of a collaborative system that will further enable effective social interaction through a greater degree of personal expression by the means of gesturing. The designed followed a

standard VRML-Java interface model which allows the elements of a virtual environment, represented through VRML, be controlled and implemented through Java. This prototype design consciously made the point of excluding forms of gestures other than emblematic ones due to the burdensome equipment and/or mechanisms that other forms of gestures, such as speech-accompanying types, representation would require.

6.7.1 Positives of the Design

The most significant capability of this prototype design is that it would provide the user the ability to express gestures to convey certain meaning and emotional states in a very simple and natural manner. By having the ability to gesture certain messages to others, a greater sense of natural face-to-face human interaction is achieved. In such artificial representation of humans in virtual environments, it is very difficult to have the natural freedom to express oneself, thus reducing the realistic social interaction experience. With a gesturing ability, it gives avatars and thereby the users in the environment the ability to express themselves (personal expression component of social interaction) which in turn increases social feedback. Finally, because this prototype is designed to be implemented on top of a collaboration system such as CAIRO, the architecture may very easily be connected into the CAIRO structure as part of the 3D interface driver as completed by DiSEL '98-'99.

6.7.2 Shortfalls

One possible concern about the design would be the technology that is involved in it, namely the VRML-Java interface. Despite its use, the technology is relatively young and thus not as robust as it can be. Moreover, the technology behind VRML and Java coupling has not been formally standardized, and many are experimenting with proposed ideas for specifications. Although in theory, the designed prototype should run smoothly once implemented, the interface may still pose to be unstable to a degree, resulting in unexplained crashes, for example. Moreover, a robust VRML-Java browser or canvas

needs to be embedded as well in order to fully take advantage of the ideas proposed behind the gesturing prototype.

Another shortfall may be that although emblematic gestures can be represented consciously by the user, other types of gesturing, such as iconics and beats (refer to Chapter 3) cannot be expressed readily. While this is largely a matter of having the user connected to the system through sensor equipment, when considering personal expression and social interaction as a whole in virtual environments, the lack of these human behaviors present an element of unnaturalness. Despite the fact that those types of gestures were purposely excluded in this thesis, that does not mean that it is not important in the general perspective of collaboration and social interaction in virtual environments. The user ideally should, in theory, be able to express a full range of gestures and expressions to convey and perceive ideas to and from other users to attain the most realistic interaction experience. Section 7.2 of the following chapter will extend the discussion of future work that is needed in gesture expression for the user in a virtual environment. Moreover, methods of how user can communicate with the computer to produce natural gestures that will be expressed through the user's representation in the virtual world will also be explored further in Chapter 7.

Chapter 7

Conclusion

Anyone who has taken part in a collaborative process can attest to the importance of communication among the participants in the successfulness of the effort. As we have seen through the discussion of social interaction, there is little doubt that the ability to transmit and perceive elements of human behavior is integral in achieving a full degree of effective collaboration. These conditions hold true for both face-to-face meetings as well as meetings conducted over a virtual environment through a collaboration enabling system such as CAIRO.

7.1 Overview

This thesis has given an in-depth overview of one of the fundamental concepts of effective collaboration, namely social interaction, and how this element can be enhanced in a virtual environment where collaborative meeting effort is conducted. A current research-based collaboration system, CAIRO, was introduced as a working example of how social interaction can be implemented in a networked system. Distributed Software Engineering Laboratory was also discussed to bring to light some of the work that has been conducted to enhance social interaction in the context of distributed collaboration. Finally, a specific aspect of personal expression, gestures, that act to increase social feedback and thus, social interaction, was explored. Significance and relevance of gestures to social interaction and communication was discussed, concluding with a

proposed design of a gesture expression prototype that would allow a user of a system such as CAIRO to control gesturing behavior of his representative avatar.

7.2 Future

A emblematic gesture expression capability that is designed in this thesis will enhance the ability of a user in a virtual environment to express himself, which effectively will result in a greater degree of social interaction among the participants. However, emblematic gesture capabilities in such a collaborative system as CAIRO are only a step toward achieving a level of social interaction that is as natural and effective as a face-to-face one. As discussed in Section 6.7.2, other types of gestures such as beats, deictics, and iconics need to be represented in order for the system to successfully achieve a level of natural human interaction in the virtual environment that is put forth by the three-dimensional interface of CAIRO. In addition to gesturing ability of the avatars, other forms of non-verbal communication, such as facial expression and gaze, needs to be incorporated into the user's real-time control over the avatar's behavior. Moreover, there is also work that needs to be done to actually implement the design into the existing three-dimensional interface and the underlying networked collaborative system. These are some of the future work that needs to be done in this field.

7.2.1 Implementation of Design

As part of work that will be done in the future, the implementation of the gesture prototype design needs to be incorporated into the three-dimensional interface developed by the DiSEL '98 team. By doing so, there would be a connection to a network that is provided by the underlying features of CAIRO. The proposed design should be able to fit into the 3D interface readily when the standardized avatars and their animations are produced successfully, as the implemented 3D interface utilizes Java and VRML technologies already.

With the gesture expression ability incorporated, social interaction would be greatly enhanced as users would be able to see and interact each other in a three-dimensional space through a three-dimensional representation of themselves via an avatar. Personal expression would be increased, social feedback would also increase as a result, and the ability to collaborate would also increase with this enhanced level of interaction and communication in the virtual environment.

7.2.2 Methods of Representing Other Types of Gestures

As mentioned in Section 6.7.2, one of the shortfalls of the prototype is that it excludes the user from representing what would be categorized as speech-accompanying gestures that include those gestures that are iconic, metaphoric, deictic, and beats. In order for these gesture types to be expressed and represented via an avatar in the virtual environment, some sort of communication between the user and the computer is necessary. This is so because in order for the system to represent real-time gesturing behavior of the user, there must be some kind of connection between what the user is saying or doing in that virtual environment and the computer system that takes that information to provide the relevant avatar motion. There are several methods to model this communication between the user and the computer to transfer such information.

7.2.2.1 Sensor Equipment

One method by which a user can communicate with the computer to feed information about his physical gesturing motions is by wearing sensor equipment. An example of such a gesture recognition system is Gandalf, developed at the Gesture and Narrative Language Group at the MIT Media Lab (Thórisson, 1997). Such system by which the user would wear a sensor equipment, for example an electromagnetic tracking system, on his body is possible. Another type of sensor equipment that has been used for hand gestures is a DataGlove, which is put on the user's hand to interact with the computer system which then uses a recognition software to generate and express the gestures (Wexelblat, 1994). However, for obvious physical constraints posed by the intrusive and

bulky nature of the method, such an implementation for a real-time collaboration system seems far-fetched.

7.2.2.2 Wireless Gesture Detection

A next generation of sensor equipment that can be used to detect gesture motion uses video cameras (Gesture and Narrative Language Group). Such a method uses approaches from image processing and speech recognition technology to recognize hand gestures. One method involves producing a chain code using the position and orientation of hands and fingers as captured in video images (CHI '95). Motion is determined through determination of differences in frame-to-frame. A neural network then can be setup that can be trained on a number of gestures, relying on an assumption that gesture recognition is a pattern recognition process (CHI '95).

7.2.2.3 Gesture Recognition via Speech and/or Text Input

Another method by which a user and the computer can interact to generate relevant real-time gestures is through a recognition mechanism by which speech, text, or the combination of the two can be recognized automatically by the system and consequently generate a matching gesture motion. If the user is interacting with others in the virtual environment by verbally speaking into a microphone, a speech recognizer can be implemented along with an interpreter that will extract certain semantic features of the speech to produce a relevant gesture representation. Similarly, if a text dialog is used, a recognizer will observe the speech that is typed, analyze the context of speech, and then use an interpreter to translate it to a relevant gesture motion. The combination of speech and text can be utilized by using the two methods together.

A problem with this approach is that the robustness and effectiveness of the recognizer totally depends on the capability of the speech recognizer and the interpreter. The approach and the result is only as good and reliable as the ability of the recognizer to correctly identify words and/or phrases, recognize their context in speech, and associate them with the appropriate gestures. Such task is enormous and complicated, and it is no

surprise that much of the existing technology for this is not nearly as dependable and robust as one would like it to be (CHI '95).

7.3 Final Thoughts

Enabling an effective form of collaboration for a distributed team over a networked system is a challenging task to perfect. Many issues of communication and human interaction arise when addressing such a topic as seen in Chapters 2 and 3. Specifically, elements of social interaction as it pertains to collaborative virtual environments were considered in this thesis. Gestures were also shown to play an important role in social interaction in virtual environments. To the aim of enhancing social interaction elements in a collaborative system like CAIRO, DiSEL '98 developed a 3D interface and this thesis explored the design of a gesturing capability within this context.

There still remains much work to be done in perfecting the social interaction capabilities over a virtual environment in a collaboration setting. Technology still limits us in bringing certain aspects of this to life in a computer system. However, with advancements in technology and through continuous efforts, as the DiSEL teams provided, collaboration and social interactions within the collaborative context will undoubtedly improve.

Bibliography

Blaxxun Interactive, Inc (1999)
http://www.blaxxun.com/vrml/avatarden/
Last visited on May 10, 1999.

Cassell, J., Pelachaud, C., Badler N., Steedman, M., Achorn, B., Becket, T., Douville, B., Prevost, S., Stone, M. (1994) "Animated Conversation: Rule-based Generation of Facial Expression, Gesture & Spoken Intonation for Multiple Conversational Agents." *Proceedings of SIGGRAPH '94.*

Cassell, J., Stone, M., Douville, B., Prevost, S., Achorn, B., Steedman, M., Badler, N., Pelachaud, C. (1994) "Modeling the Interaction between Speech and Gesture." *Proceedings of the Cognitive Science Society Annual Conference 1994.*

CHI '95 (1995) *1995 Conference on Human Factors in Computing Systems*
http://wex.media.mit.edu/people/wex/CHI95-workshop-writeup.html
last visited on April 20, 1999.

Descarte, A. (1996) "Interfacing Java and VRML." UK Java Developer's Conference Paper: November, 1996.

DiSEL 1997 http://cscw.cicese.mx/~disel/
Last visited on May 10, 1999.

DiSEL 1998 http://cscw.cicese.mx/~disel98/
Last visited on May 10, 1999.

Efron, D. (1941/1972) *Gesture and Environments.* Morningside Heights, NY: Kings Crown Press. *Republished as Gesture, race, and culture.* The Hague, Mounton.

Ekman, P. & Friesen W. (1969) "The Repertoire of Nonverbal Behavior: Categories, Origins, Usage, and Coding." *Semiotica,* 1: 49-98.

Ekman, P. & Friesen W. (1972) "Hand Movements." *The Journal of Communication,* 22: 353-374.

Feldman, R. & Rime, B. [Eds] (1991) *Fundamentals of Nonverbal Behavior.* New York: Cambridge University Press.

Feyereisen, Pierre & Jacques-Dominique de Lannoy (1991) *Gestures and Speech: Psychological Investigations.* New York: Cambridge University Press.

Freedman, N. (1972) "The analysis of movement behavior during clinical interviews." In Siegman, A. & Pope, B. [Eds], *Studies in Dyadic Communication.* Elmsford, NY: Pergamon Press.

Gesture and Narrative Language Group – MIT Media Lab (1999)
http://avatars.www.media.mit.edu/avatars/
Last visited on May 5, 1999.

Hiltz, S. R., & Turoff, M. (1978) *The Network Nation, Human Interaction via Computer.* Reading, MA: Addison-Wesley Publishing Company.

Hussein, Karim Mohie El Din (1998) *Computer supported interaction in distributed design teams.* Sc. D. Thesis, Massachusetts Institute of Technology, Department of Civil and Environmental Engineering.

Jarvella, R. & Wolfgang, K. [Eds] (1982) *Speech, Place, and Action: Studies in Deixis and Related Places.* Chichester: John Wiley & Sons Ltd.

Johansen, R., Vallee, J., and Collins, K. (1977) "Learning the limits of Teleconferencing: Design of a Teleconference Tutorial," Proceedings of *NATO Symposium on Evaluation and Planning of Telecommunications Systems* Univ. of Bergamo, Italy, Sept 1977. In *The Network Nation*, (Hiltz, 1978).

Kendon, Adam (1997) "Gesture." *Annual Review of Anthropology*, 26: 109-128.

Kendon, Adam (1992) "Abstraction in Gesture." *Semiotica*, Volume 90, Number 3/4: 225-250.

Kendon, Adam (1994) "Do Gestures Communicate?" *Research on Language and Social Interaction*, 27, 3: 175-200.

Kendon, Adam (1993) "Human Gesture." In Gibson, Kathleen R & Ingold, Tim [Eds], *Tools, Languages, and Cognition in Human Evolution.* Cambridge, Englnad: Cambridge University Press.

Kendon, Adam (1983) "Gestures and Speech: How They Interact." In Weimann, J & Harrison, R [Eds], *Nonverbal Interaction.* Beverly Hills: Sage Publications.

Kendon, Adam (1988) "How Gestures Can Become Like Words." In Poyatos, F, *Cross-cultural Perspectives in Nonverbal Communication.* Toronto: CJ Hogrefe.

Lea, Roger (1997) "Java and VRML 2.0 Parts 1&2."
http://www.vrmlsite.com/feb97/a.cgi/spot2.html
Last visited on March 25, 1999.

Levy, Elena T. & McNeill, David (1992) "Speech, Gesture, and Discourse." *Discourse Processes*, Volume 15, Number 3, July-Sept: 277-301.

Marrin, Chris (November, 1997) "Proposal for a VRML 2.0 Informative Annex - External Authoring Interface Reference."
http://www.vrml.org/WorkingGroups/vrml-eai/ExternalInterface.html#1.2
Last visited on April 26, 1999.

Marrin, C., McCloskey, B., Sandvik, K., Chin, D. (May, 1997) "EAI White Paper."
http://cosmosoftware.com/developer/index_allabout.html
Last visited on April 26, 1999.

Mantovani, G. (1996) *New Communication Environments, from Everyday to Virtual.* Bristol, PA: Taylor & Francis.

McNeill, David (1985) "So You Think Gestures Are Nonverbal?" *Psychological Review*, 92: 350-371.

McNeill, David (1992) *Hand In Mind: What Gestures Reveal About Thought.* Chicago: University of Chicago Press.

McNeill, David & Levy, Elena T (1982) "Conceptual representations in language activity and gesture." In R. Javella & W. Klein [Eds.], *Speech Place, and Action* (pp271-295). New York: Wiley.

MIT 50K Entrepreneurship Competition
http://50k.mit.edu
Last visited on May 8, 1999.

Nespoulous, J. & Lecours, AR (1986) "Gestures: Nature and Function." In Nespoulous, Perron, and Lecours [Eds], *Biological Foundations of Gestures: Motor and Semiotic Aspects.* Hilldale, NJ: Lawerence Erlbaum Associates.

Picard, R. W. (1995) "Affective Computing" MIT Media Laboratory Perceptual Computing Section Technical Report No 321, revised November 26, 1995, submitted for publication.

Rime, B. & Schiaratura, L (1991) "Gesture and Speech." In Feldman & Rime [Eds], *Fundamentals of Nonverbal Behavior.* New York: Press Syndicate of the University of Cambridge.

Rodriguez, S. (1998) "Requirements For A Social Interaction and Casual Contact Tool."

Roehl, B. (1999) Humanoid Animation Working Group, Specification for a Standard VRML Humanoid Version 1.1
http://ece.uwaterloo.ca/~h-anim/newspec.html
Last visited on May 1, 1999.

Roehl, B., Couch, J., Reed-Ballreich, C., Rohaly, T., Brown, G. (1997) *Late Night VRML 2.0 with Java.* Emeryville: Ziff-Davis Press.

Schneider, D., Martin-Michiellot, S. (1998) *VRML Primer and Tutorial.* TECFA, Faculte de Psychologie et des sciences de l'education, University of Geneva.

Scott, Adrian (1996) "The Marriage of Java and VRML."
http://www.vrmlsite.com/sep96/spotlight/javavrml/javavrml.html
Last visited on May 10, 1999.

Sonstein, Jeff (1996) "VRML Goes Dynamic."
http://www.vrmlsite.com/oct96/spotlight/proto/proto.html
Last visited on May 10, 1999.

Su, Christine Hui (1998) *Distributed Software Design for Collaborative Learning System Over the Internet.* S.B. & M.Eng. Thesis, MIT Department of Electrical Engineering and Computer Science.

Thórisson, K. R. (1997) "Gandalf: An Embodied Humanoid Capable of Real-Time Multimodal Dialogue with People." *First ACM International Conference on Autonomous Agents*, Mariott Hotel, Marina del Rey, California, February 5-8, 1997, 536-7.

Vilhjálmsson, H. (1996) "Autonomous Communicative Behaviors in Avatars." (Abstract) *Proceedings of Lifelike Computer Characters* '96, Utah, Oct 8-11: 49.

Vilhjálmsson, H. (1997) *Autonomous Communicative Behaviors in Avatars.* S.M. Thesis, MIT Media Arts and Sciences.

Vilhjálmsson, H., Cassell, J. (1998) "Body Chat: Autonomous Communicative Behaviors in Avatars." *Proceedings of ACM Autonomous Agents '98*, Mineapolis, May 9-13: 269-276.

Wexelblat, A. (1994) *A Feature-Based Approached to Continuous Gesture Analysis.* S.M. Thesis, MIT Media Arts and Sciences.

Wexelblat, A. (1994) "Natural Gestures in Virtual Environments."

Appendix A

VRML code structure for avatar animation (adapted from Blaxxun, 1999)

```
DEF Animations Group {
 children [
  DEF WalkAnimation Group {
   children [
    DEF r_shoulderRotWalk OrientationInterpolator {
     key [ 0, … ]
     keyValue [1 0 0 0, …]
    }

    DEF r_hipRotWalk OrientationInterpolator {
     key [ 0, … ]
     keyValue [1 0 0 0, …]
    }

    DEF r_kneeRotWalk OrientationInterpolator {
     key [ 0, … ]
     keyValue [1 0 0 0, …]
    }

    DEF r_elbowRotWalk OrientationInterpolator {
     key [ 0, … ]
     keyValue [1 0 0 -0.4, …]
    }

    DEF l_shoulderRotWalk OrientationInterpolator {
     key [ 0, … ]
     keyValue [1 0 0 0, …]
    }

    DEF l_hipRotWalk OrientationInterpolator {
     key [ 0, … ]
     keyValue [1 0 0 0, …]
    }

    DEF l_kneeRotWalk OrientationInterpolator {
```

```
   key [ 0, … ]
   keyValue [1 0 0 1, …]
  }

  DEF l_elbowRotWalk OrientationInterpolator {
   key [ 0, … ]
   keyValue [1 0 0 0, …]
  }

  DEF v15RotWalk OrientationInterpolator {
   key [ 0, … ]
   keyValue [1 0 0 0, …]
  }

  DEF HumanoidRootTransWalk PositionInterpolator {
   key [ 0, … ]
   keyValue [ 0 0 0, … ]
  }
 ]
}

DEF StopAnimation Group {
 children [
  DEF jointTransStop PositionInterpolator {
   key [ 0, … ]
   keyValue [ 0 0 0, … ]
  }

  DEF jointRotStop OrientationInterpolator {
   key [ 0, … ]
   keyValue [ 1 0 0 0, … ]
  }
 ]
}

DEF NoAnimation Group {
 children [

  DEF r_shoulderRotNo OrientationInterpolator {
   key [ 0, … ]
   keyValue [1 0 0 0, …]
  }

  DEF r_elbowRotNo OrientationInterpolator {
   key [ 0, … ]
   keyValue [1 0 0 0, …]
  }

  DEF r_wristRotNo OrientationInterpolator {
   key [ 0, … ]
   keyValue [0 0 1 0, …]
  }

  DEF skullbaseRotNo OrientationInterpolator {
   key [ 0, … ]
   keyValue [0 1 0 0, …]
  }
```

```
 ]
}

DEF HiAnimation Group {
 children [

  DEF v15RotHi OrientationInterpolator {
   key [ 0, … ]
   keyValue [1 0 0 0, …]
  }

  DEF 1_shoulderRotHi OrientationInterpolator {
   key [ 0, … ]
   keyValue [1 0 0 0, …]
  }

  DEF 1_elbowRotHi OrientationInterpolator {
   key [ 0, … ]
   keyValue [1 0 0 0, …]
  }

  DEF 1_wristRotHi OrientationInterpolator {
   key [ 0, … ]
   keyValue [0 0 1 0,…]
  }
 ]
}

DEF HeyAnimation Group {
 children [

  DEF HumanoidRootTransHey PositionInterpolator {
   key [ 0, … ]
   keyValue [ 0 0 0, …]
  }

  DEF HumanoidRootRotHey OrientationInterpolator {
   key [ 0, … ]
   keyValue [1 0 0 0, …]
  }

  DEF skullbaseRotHey OrientationInterpolator {
   key [ 0, 0.25, 0.5, 0.75, 1 ]
   keyValue [1 0 0 0, …]
  }

  DEF shoulderRotHey OrientationInterpolator {
   key [ 0, … ]
   keyValue [1 0 0 0, …]
  }

  DEF elbowRotHey OrientationInterpolator {
   key [ 0, … ]
   keyValue [1 0 0 0, …]
  }

  DEF hipRotHey OrientationInterpolator {
```

```
      key [ 0, … ]
      keyValue [1 0 0 0, …]
     }

     DEF kneeRotHey OrientationInterpolator {
      key [ 0, …]
      keyValue [1 0 0 0, …]
     }

     DEF v15RotHey OrientationInterpolator {
      key [ 0, … ]
      keyValue [1 0 0 0, …]
     }

    ]
   }

   DEF YesAnimation Group {
    children [

     DEF l_elbowRotYes OrientationInterpolator {
      key [ 0, … ]
      keyValue [1 0 1 0, …]
     }

     DEF l_shoulderRotYes OrientationInterpolator {
      key [ 0, …]
      keyValue [1 0 0 0, …]
     }

     DEF vc4RotYes OrientationInterpolator {
      key [ 0, … ]
      keyValue [0 0 1 0, …]
     }
    ]
   }
  ]
}

DEF Humanoid Humanoid {
 version "1.0"
 name "humanoid avatar"
 info [
  "authorName=Bluxxun/Joon Hor",
  "authorEmail=gords@mit.edu",
  "creationDate=April 20, 1999",
 ]
 joints [
  USE hanim_HumanoidRoot,
  USE hanim_sacroiliac,
  USE hanim_r_hip,
  USE hanim_r_knee,
  USE hanim_r_ankle,
  USE hanim_l_hip,
  USE hanim_l_knee,
  USE hanim_l_ankle,
  USE hanim_v15,
```

```
  USE hanim_vc7,
  USE hanim_r_shoulder,
  USE hanim_r_elbow,
  USE hanim_r_wrist,
  USE hanim_l_shoulder,
  USE hanim_l_elbow,
  USE hanim_l_wrist,
  USE hanim_vc4,
  USE hanim_skullbase
 ]
 segments [
  USE hanim_pelvis,
  USE hanim_l_thigh,
  USE hanim_l_calf,
  USE hanim_l_hindfoot,
  USE hanim_r_thigh,
  USE hanim_r_calf,
  USE hanim_r_hindfoot,
  USE hanim_c7,
  USE hanim_l_upperarm,
  USE hanim_l_forearm,
  USE hanim_l_hand,
  USE hanim_r_upperarm,
  USE hanim_r_forearm,
  USE hanim_r_hand,
  USE hanim_c4,
  USE hanim_skull
 ]
}

# Walk
ROUTE TimeWalk.fraction_changed TO HumanoidRootTransWalk.set_fraction
ROUTE TimeWalk.fraction_changed TO v15RotWalk.set_fraction
ROUTE TimeWalk.fraction_changed TO r_shoulderRotWalk.set_fraction
ROUTE TimeWalk.fraction_changed TO r_elbowRotWalk.set_fraction
ROUTE TimeWalk.fraction_changed TO r_hipRotWalk.set_fraction
ROUTE TimeWalk.fraction_changed TO r_kneeRotWalk.set_fraction
ROUTE TimeWalk.fraction_changed TO l_shoulderRotWalk.set_fraction
ROUTE TimeWalk.fraction_changed TO l_elbowRotWalk.set_fraction
ROUTE TimeWalk.fraction_changed TO l_hipRotWalk.set_fraction
ROUTE TimeWalk.fraction_changed TO l_kneeRotWalk.set_fraction

ROUTE HumanoidRootTransWalk.value_changed TO
hanim_HumanoidRoot.set_translation
ROUTE v15RotWalk.value_changed TO hanim_v15.set_rotation
ROUTE r_shoulderRotWalk.value_changed TO hanim_r_shoulder.set_rotation
ROUTE r_elbowRotWalk.value_changed TO hanim_r_elbow.set_rotation
ROUTE r_hipRotWalk.value_changed TO hanim_r_hip.set_rotation
ROUTE r_kneeRotWalk.value_changed TO hanim_r_knee.set_rotation
ROUTE l_shoulderRotWalk.value_changed TO hanim_l_shoulder.set_rotation
ROUTE l_elbowRotWalk.value_changed TO hanim_l_elbow.set_rotation
ROUTE l_hipRotWalk.value_changed TO hanim_l_hip.set_rotation
ROUTE l_kneeRotWalk.value_changed TO hanim_l_knee.set_rotation

# Stop
ROUTE TimeStop.fraction_changed TO jointTransStop.set_fraction
ROUTE TimeStop.fraction_changed TO jointRotStop.set_fraction
```

```
ROUTE jointTransStop.value_changed TO
hanim_HumanoidRoot.set_translation
ROUTE jointRotStop.value_changed TO hanim_v15.set_rotation
ROUTE jointRotStop.value_changed TO hanim_r_shoulder.set_rotation
ROUTE jointRotStop.value_changed TO hanim_r_elbow.set_rotation
ROUTE jointRotStop.value_changed TO hanim_r_hip.set_rotation
ROUTE jointRotStop.value_changed TO hanim_r_knee.set_rotation
ROUTE jointRotStop.value_changed TO hanim_l_shoulder.set_rotation
ROUTE jointRotStop.value_changed TO hanim_l_elbow.set_rotation
ROUTE jointRotStop.value_changed TO hanim_l_hip.set_rotation
ROUTE jointRotStop.value_changed TO hanim_l_knee.set_rotation

# Hello
ROUTE Time_1.fraction_changed TO v15RotHi.set_fraction
ROUTE Time_1.fraction_changed TO l_shoulderRotHi.set_fraction
ROUTE Time_1.fraction_changed TO l_elbowRotHi.set_fraction
ROUTE Time_1.fraction_changed TO l_wristRotHi.set_fraction

ROUTE v15RotHi.value_changed TO hanim_v15.set_rotation
ROUTE l_shoulderRotHi.value_changed TO hanim_l_shoulder.set_rotation
ROUTE l_elbowRotHi.value_changed TO hanim_l_elbow.set_rotation
ROUTE l_wristRotHi.value_changed TO hanim_l_wrist.set_rotation

# Hey
ROUTE Time_2.fraction_changed TO HumanoidRootTransHey.set_fraction
ROUTE Time_2.fraction_changed TO skullbaseRotHey.set_fraction
ROUTE Time_2.fraction_changed TO shoulderRotHey.set_fraction
ROUTE Time_2.fraction_changed TO elbowRotHey.set_fraction
ROUTE Time_2.fraction_changed TO HumanoidRootRotHey.set_fraction
ROUTE Time_2.fraction_changed TO hipRotHey.set_fraction
ROUTE Time_2.fraction_changed TO kneeRotHey.set_fraction
ROUTE Time_2.fraction_changed TO v15RotHey.set_fraction
ROUTE HumanoidRootTransHey.value_changed TO
hanim_HumanoidRoot.set_translation
ROUTE skullbaseRotHey.value_changed TO hanim_skullbase.set_rotation
ROUTE shoulderRotHey.value_changed TO hanim_l_shoulder.set_rotation
ROUTE shoulderRotHey.value_changed TO hanim_r_shoulder.set_rotation
ROUTE elbowRotHey.value_changed TO hanim_l_elbow.set_rotation
ROUTE elbowRotHey.value_changed TO hanim_r_elbow.set_rotation
ROUTE HumanoidRootRotHey.value_changed TO
hanim_HumanoidRoot.set_rotation
ROUTE hipRotHey.value_changed TO hanim_l_hip.set_rotation
ROUTE hipRotHey.value_changed TO hanim_r_hip.set_rotation
ROUTE kneeRotHey.value_changed TO hanim_l_knee.set_rotation
ROUTE kneeRotHey.value_changed TO hanim_r_knee.set_rotation
ROUTE v15RotHey.value_changed TO hanim_v15.set_rotation

# Yes
ROUTE Time_3.fraction_changed TO skullbaseRotHey.set_fraction
ROUTE Time_3.fraction_changed TO vc4RotYes.set_fraction
ROUTE Time_3.fraction_changed TO l_shoulderRotYes.set_fraction
ROUTE Time_3.fraction_changed TO l_elbowRotYes.set_fraction
ROUTE vc4RotYes.value_changed TO hanim_vc4.set_rotation
ROUTE l_elbowRotYes.value_changed TO hanim_l_elbow.set_rotation
ROUTE l_shoulderRotYes.value_changed TO hanim_l_shoulder.set_rotation
```

```
# No
ROUTE Time_6.fraction_changed TO skullbaseRotNo.set_fraction
ROUTE Time_6.fraction_changed TO r_shoulderRotNo.set_fraction
ROUTE Time_6.fraction_changed TO r_elbowRotNo.set_fraction
ROUTE Time_6.fraction_changed TO r_wristRotNo.set_fraction

ROUTE skullbaseRotNo.value_changed TO hanim_skullbase.set_rotation
ROUTE r_shoulderRotNo.value_changed TO hanim_r_shoulder.set_rotation
ROUTE r_elbowRotNo.value_changed TO hanim_r_elbow.set_rotation
ROUTE r_wristRotNo.value_changed TO hanim_r_wrist.set_rotation
```

Appendix B

Java EAI Package Hierarchy (Marrin, 1997)

The Java implementation of the External Authoring Interface in Java packages: **vrml.external**, **vrml.external.field**, and **vrml.external.exception**

```
vrml.external
  |
  +- vrml.external.IBrowser
  |       +- vrml.external.Browser
  |       +- vrml.external.CBrowser
  |
  +- vrml.external.Node
  +- vrml.external.field
  |       +- vrml.external.field.EventIn
  |       |       +- vrml.external.field.EventInMFColor
  |       |       +- vrml.external.field.EventInMFFloat
  |       |       +- vrml.external.field.EventInMFInt32
  |       |       +- vrml.external.field.EventInMFNode
  |       |       +- vrml.external.field.EventInMFRotation
  |       |       +- vrml.external.field.EventInMFString
  |       |       +- vrml.external.field.EventInMFVec2f
  |       |       +- vrml.external.field.EventInMFVec3f
  |       |       +- vrml.external.field.EventInSFBool
  |       |       +- vrml.external.field.EventInSFColor
  |       |       +- vrml.external.field.EventInSFFloat
  |       |       +- vrml.external.field.EventInSFImage
  |       |       +- vrml.external.field.EventInSFInt32
  |       |       +- vrml.external.field.EventInSFNode
  |       |       +- vrml.external.field.EventInSFRotation
  |       |       +- vrml.external.field.EventInSFString
  |       |       +- vrml.external.field.EventInSFTime
  |       |       +- vrml.external.field.EventInSFVec2f
  |       |       +- vrml.external.field.EventInSFVec3f
  |       |
  |       +- vrml.external.field.EventOut
  |       |       +- vrml.external.field.EventOutMField
  |       |       |       +- vrml.external.field.EventOutMFColor
  |       |       |       +- vrml.external.field.EventOutMFFloat
  |       |       |       +- vrml.external.field.EventOutMFInt32
```

```
|       |       |       +- vrml.external.field.EventOutMFNode
|       |       |       +- vrml.external.field.EventOutMFRotation
|       |       |       +- vrml.external.field.EventOutMFString
|       |       |       +- vrml.external.field.EventOutMFVec2f
|       |       |       +- vrml.external.field.EventOutMFVec3f
|       |       |
|       |       +- vrml.external.field.EventOutSFBool
|       |       +- vrml.external.field.EventOutSFColor
|       |       +- vrml.external.field.EventOutSFFloat
|       |       +- vrml.external.field.EventOutSFImage
|       |       +- vrml.external.field.EventOutSFInt32
|       |       +- vrml.external.field.EventOutSFNode
|       |       +- vrml.external.field.EventOutSFRotation
|       |       +- vrml.external.field.EventOutSFString
|       |       +- vrml.external.field.EventOutSFTime
|       |       +- vrml.external.field.EventOutSFVec2f
|       |       +- vrml.external.field.EventOutSFVec3f
|       |
|       +- vrml.external.field.EventOutObserver
|       +- vrml.external.field.FieldTypes
|
+- vrml.external.exception
        +- vrml.external.exception.InvalidEventInException
        +- vrml.external.exception.InvalidEventOutException
        +- vrml.external.exception.InvalidNodeException
        +- vrml.external.exception.InvalidVrmlException
```