VoiceNotes: An Application for a Voice-Controlled Hand-Held Computer

by

Lisa Joy Stifelman

B.S., Engineering Psychology
Tufts University
Medford, Massachusetts
1988

SUBMITTED TO THE MEDIA ARTS AND SCIENCES SECTION,
SCHOOL OF ARCHITECTURE AND PLANNING, IN PARTIAL FULFILLMENT OF THE REQUIREMENTS OF THE DEGREE OF

MASTER OF SCIENCE

AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
JUNE 1992

Signature of the Author

Media Arts and Sciences Section
May 8, 1992

Certified by

Christopher M. Schmandt
Principal Research Scientist
MIT Media Laboratory

Accepted by

Stephen A. Benton
Chairperson
Department Committee on Graduate Students

VoiceNotes: An Application for a Voice-Controlled Hand-Held Computer

by

Lisa Joy Stifelman

SUBMITTED TO THE MEDIA ARTS AND SCIENCES SECTION,
SCHOOL OF ARCHITECTURE AND PLANNING, ON MAY 8, 1992 IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS OF THE DEGREE OF

MASTER OF SCIENCE

AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Abstract

The ultimate computer has often been described as one you can always carry around with you, or wear like the "Dick Tracy watch," yet an appropriate interface to such a computer has not been designed. This thesis explores a user interface for a hand-held computer that employs speech input and output as the primary means of interaction instead of the traditional screen and keyboard. VoiceNotes is an application for creating and managing lists of recorded segments of the user's speech and was developed with the goal of exploring the design of voice interfaces to hand-held computers. Issues of input, navigation, management, and retrieval of structured audio data are addressed.

Thesis Supervisor: Christopher M. Schmandt
Title: Principal Research Scientist

This work has been sponsored by Apple Computer, Inc.

Thesis Committee

Thesis Advisor

Christopher M. Schmandt
Principal Research Scientist
MIT Media Laboratory

Reader

Eric A. Hulteen
Human Interface Group/ATG
Apple Computer, Inc.

Reader

Victor W. Zue
Principal Research Scientist
MIT Department of Electrical Engineering and Computer Science

Table of Contents

List of Figures

1. Introduction

Have you ever found yourself in a situation where you need to record an important idea, thoughts about a project, a reminder, or an appointment you've just made? You are not near a computer and you don't have a notebook or pen with you. Maybe if you're lucky you find a scrap of torn paper to jot the item down on. Later you must re-write these scattered notes or else they are lost altogether.

Now imagine a hand-held computer that could fit inside your pocket and always be with you for recording your ideas, appointments, and things to do. Whether you are in a car, the airport, at a meeting, or out to lunch with a friend, you will no longer miss these opportunities to capture timely and important information.

This thesis demonstrates a user interface for a hand-held computer that allows the creation, management, and retrieval of user-authored voice notes—recorded segments of the user's speech. The development of the VoiceNotes application serves to explore the use of stored audio as a data type and the use of speech recognition in the user interface for a hand-held computer. It is expected that the hand-held will be portable and light-weight enough to be carried everywhere. Therefore, it can be used in a variety of situations—when one's hands and eyes are busy as when driving a car; under noisy conditions such as in a subway. For this reason, two input modalities are provided for interaction with the VoiceNotes application—speech recognition and button input. While the goal of this research has been to explore the use of voice interfaces and voice-based applications in hand-held computers, the combination of voice and button input provides a more interactive interface than can be provided with either alone.

1.1 Why a Voice Interface?

How can I make notes to myself while riding the subway holding on with one hand? How can I review the things that I am supposed to do today while riding my bike or driving my car? What kind of computer can I take with me to a meeting or out to lunch for recording ideas?

There is a current trend toward scaling computers down in size. Laptop computers are small, light-weight, portable versions of desktop PCs, but the user interface has essentially remained unchanged. Notebook size computers, such as the Go, use a pen for entering hand-written text on a flat display. In addition, there are a host of small specialized address books, travel keepers, and other types of organizers. The HP 95LX "palmtop" is a complete DOS PC that can fit in a shirt pocket.

These devices have one thing in common—they all employ a visual display, and most have a small keyboard. The smaller the computer gets, the more difficult it becomes to touch-type (or even to physically press the correct keys), and to read the information on the small display. In addition, keys are typically overloaded with more than one function. As computers decrease in size, so does the utility of common personal computer input and output modalities (keyboards, mice, and high resolution displays). Thus, functionality and ease-of-use are limited on these small devices in which designers have tried to "squeeze" more and more features onto an ever decreasing product size.

The ultimate computer has often been described as one you can always carry around with you, or wear like the "Dick Tracy watch," yet an appropriate interface to such a computer has not yet been designed. This thesis considers a different concept of the ultimate ubiquitous hand-held computer—one that has no keyboard or visual display, but uses a speech interface instead. Information is stored in an audio format, as opposed to textual, and accessed by issuing spoken commands instead of typing. Feedback is also provided aurally instead of visually. By exploring interfaces employing speech input and output as the primary means of interaction, general ideas and techniques for dealing with speech in the user interface can be discovered.

1.2 Research Challenges

Why is this a difficult problem? There are two important research challenges: (1) taking advantage of the utility stored voice as a data type for a hand-held computer while overcoming its liabilities (e.g., speech is slow, serial, bulky); (2) determining the role of voice in a user interface to a hand-held computer given the limitations in current speech recognition technology.

1.2.1 Utility of Stored Voice as a Data Type for a Hand-held Computer

Project ideas, lists of things to do, or segments of speech captured during a meeting or brainstorming session are examples of audio data that would be useful to capture using a hand-held device. For recording project ideas or to-do lists, speech may provide a faster, more direct and natural means of input than text. Speech provides portability—a microphone is much smaller than a keyboard. During a meeting or brainstorming session, speech is used to communicate ideas, yet if this information is recorded, it is usually recorded textually. Information found in the original speech, such as intonation and emotion, is lost. Capturing portions of the conversational speech preserves this information and is useful for recalling the exact wording of an utterance.

Speech is a natural means of interaction, yet recording, retrieving, and navigating among such spoken information is a challenging problem. When recording information in a written format, a spatial layout is typically used to organize the information. How can the same organization be accomplished when the data is recorded speech? When retrieving information, the eye can quickly scan a page of text to obtain the important points. Speech, on the other hand, is slow and must be accessed sequentially. When navigating among textual pieces of information, visual cues

such as highlighting and spatial layout can be used to move quickly from one idea to the next. Navigating among spoken segments of information is more difficult, due to the serial nature of speech [Muller 1990][Arons 1991]. This thesis addresses the organization and navigation among spoken segments of recorded speech.

1.2.2 Role of Voice in a User Interface to a Hand-held Computer

How can voice input be used to record, retrieve, and navigate among segments of voice data using a hand-held device that has no visual display? What should the allowable set of spoken commands be and what kind of feedback should be provided?

In developing a voice interactive system, one challenge is overcoming the deficiencies in the current technology. Voice recognition technology has not been perfected to date, however, human communication is not error-free either. Human conversations are filled with misunderstandings and false starts and stops. The difference is that humans have the ability to repair the errors and come to a common understanding, while machines currently do not [Hayes 1983]. The challenge is to create a system that provides timely feedback of the recognized input and allows for error repair.

Human conversations not only employ speech, but also use many additional methods of communication. It is therefore reasonable that the hand-held provide other means of interaction in addition to speech. A voice interface can be enhanced by integrating other modalities such as button input and non-speech audio output. There are instances in which these other interaction techniques provide a more appropriate interface than speech, and their inclusion results in a more complete conversational interface.

1.3 Why Buttons?

In addition to voice input, the hand-held device also provides button controls. While all interactions with VoiceNotes can be supported using speech input alone, there are cases in which button input provides a better, or more appropriate, interface. Buttons provide support for actions that are difficult to control with speech (e.g., speed, volume). In addition, the choice of whether to use speech or buttons will depend on the situation. Speech provides the ability to operate the hand-held device when one's hands and eyes are busy. Buttons provide a means of operating the hand-held device under conditions of extreme quiet or extreme noise. In quiet conditions, such as while attending a meeting, it may be inappropriate to speak aloud to one's hand-held computer. In noisy conditions the performance of the speech recognition system will be degraded while the button interface will not be affected. Buttons may also be faster and more reliable than voice.

1.4 Overview of the Document

Chapter 2 presents background research and applications related to the development of VoiceNotes. This includes a discussion of personal recording technology, voice-only interfaces, voice hand-held computers, and notetaking programs.

Chapter 3 provides an overview of VoiceNotes, including a demonstration of the final design and a few scenarios of how the application might be used.

Chapters 4-7 provide detailed design discussions as follows: Chapter 4 discusses the overall design of the VoiceNotes application; Chapter 5 presents issues related to voice input; Chapter 6 presents issues related to button input; Chapter 7 discusses the advantages of a hybrid voice and button interface.

Chapter 8 discusses the capture of background speech or "on-the-fly" voice notes.

Chapter 9 presents the software design for VoiceNotes and a Recognition Server.

Chapter 10 presents conclusions and possible future work.

1.5 Document Conventions

In order to describe the different voice and button interactions with the VoiceNotes application, several conventions are used. First, the user's voice must be distinguishable from the system's pre-recorded prompts. Second, user voice recordings (digitized segments of the user's speech) must be distinguishable from user spoken commands to the speech recognizer. Lastly, button commands must be distinguishable from voice commands. Figure 1.1 outlines the conventions used throughout this document.

Type of Input/Output	**Example of Notation**
System prompts/feedback	"Recording note"
User button command	*PLAY*
User voice command	"*PLAY*"
User voice recording	"*Remember to call my mother to wish her happy birthday*"

Figure 1.1: Document conventions.

2. Background

This chapter provides a discussion of research and applications related to the development of the VoiceNotes application. VoiceNotes does not correspond directly to any single area of research but rather encompasses several areas including: personal recording technology, voice-only interfaces, the use of voice in hand-held computers, and notetaking.

Additional background research regarding the use of pauses to segment conversational speech is provided in section 8.7.

2.1 Personal Recording Technology

Many people currently use microcassette recorders for keeping track of ideas, notes on projects, or things-to-do. The current state of the technology is lacking in many ways, as it does not allow users random access to the data that they have recorded.

Interface designers at Apple Computer interviewed ten users of microcassette recorders [Degen 1992]. One of the user's worst frustrations was the inability to easily locate a piece of recorded information. They used all kinds of methods in an attempt to get around this problem, none leading to very satisfactory results. One method, zeroing out the tape counter, allowed only a single mark to be created. Users also tried to write down the counter numbers but this was difficult, especially if recording information while driving.

It appears that people desire portable access for recording and reviewing audio information as they have attempted many ways to overcome problems with microcassette recorders and continue to use them despite the deficiencies.

In addition, much of the recorded information is usually manually transcribed from speech to text. This may be due in part to the difficulty managing the information in an audio format. Most users interviewed by Degen transcribed the audio material to text using a word processor or outlining application. When using a recorder for dictation, users generally gave the tapes to someone else to transcribe.

2.1.1 Apple's Augmented Tape Recorder

Degen, Mander, and Salomon augmented a tape recorder to allow users to mark segments of recorded audio [Degen 1992]. Two additional buttons were added to a standard tape recorder that create distinct audio tones on one channel of a stereo cassette. The marked audio is then downloaded to a Macintosh computer and reviewed using a graphical SoundBrowser application.

The intent was to allow users to record and mark the audio data using the portable recorder, and then review the recordings from their "desktop" using the SoundBrowser.

In an evaluation of this prototype, users expressed the desire for more buttons for distinctly marking audio segments and the ability to customize the meanings of the different marks. In addition, users desired the ability to playback the marked segments directly from the device instead of requiring that the information be downloaded to the Macintosh to review it.

This thesis addresses the results found in this study by providing the user with the ability to create personal categories and allowing direct entry, retrieval, and organization of audio data from a hand-held device.[1] Speech Recognition technology provides the user with the ability to create voice "markers" for segments of audio, and gives the user control over both the number and the meaning of the marks.

2.1.2 Voice Mail

In addition to microcassette recorders, some people use their answering machines to record reminders to themselves. Retrieval of the reminders is awkward because they are intermixed with messages from outside callers. A voice mail system developed by the Speech Research Group at the MIT Media Lab allows users to record voice memos using a remote telephone interface [Stifelman 1991]. Users can later review these memos using a graphical interface to voice mail running on a workstation. As with Apple's augmented tape recorder, this method forces users to review the items from a personal workstation rather than from a recording device that they can carry around.

2.2 Voice-only Interfaces

2.2.1 Hyperphone

Hyperphone is a voice-I/O hypermedia environment for exploring voice documents [Muller 1990]. Voice documents are composed of nodes (small fragments of text) and links (connections to other nodes). A linear sequence of nodes is known as a Parsed Voice String or PVS. The purpose of the PVS is to organize collections of fine-grained text objects to provide a linear presentation and yet allow for interruption and "excursions" to related or subtopics. Links can be literal or virtual. A literal link is a semantic connection between two pieces of information while a virtual link is a temporal connection. An example of a virtual link is the most recently traversed node. This allows the user to issue a query such as "where were we?"

The user interface for Hyperphone is voice-only—navigation through a voice document is controlled by voice input, and the information is output using text-to-speech synthesis. Muller

[1] While hand-held prototype 1 is currently "tethered" to a Macintosh, the intent is to eliminate this tether in the future.

and Daniel found that very fine-grained objects were needed in order to provide a conversational dialog as opposed to a fixed menu structure for navigation. Fine-grained segmentation of the information allows user queries such as "tell me more about this topic," "skips the details," and topic specific queries such as "who wrote the paper you're describing?" The importance of interruptibility is emphasized due to the sequential rather than simultaneous nature of a speech interaction. Navigational commands are provided by a speaker independent recognition vocabulary. Since the recognition vocabulary lacked many necessary technical and "popular" words, the use of "voice direct manipulation" was proposed to handle these cases. For example, a voice menu would be presented and the user could select an item by saying "that one" or "do it" when the desired item was being spoken (see section 5.2.3 for further discussion of voice direct manipulation).

Hyperphone provides a good example of the considerations in building a voice-only interface. However, there are some important differences between Hyperphone and VoiceNotes. Hyperphone is a collection of hypermedia objects that are organized by the author of the system while VoiceNotes deals with segments of speech authored and organized by the user. In addition, the underlying data type for Hyperphone is ASCII text, while VoiceNotes uses stored audio. A text document can be easily searched in order to find key words and phrases. Allowing similar searches of audio data would require speech-to-text conversion or key-word spotting that are challenging areas of speech recognition research (see section 10.3.1 for more about key word spotting).

2.2.2 Hyperspeech

Hyperspeech is a voice-only hypermedia system like Hyperphone but with some important differences [Arons 1991]. Hyperspeech provides the ability to navigate through a network of digitally recorded segments of speech using isolated speech recognition. Information is therefore presented using digitized speech output as opposed to synthetic speech. The recorded information is spontaneous speech from interviews with five interface designers on the topic of the future of the human interface. Responses to questions were manually segmented into nodes and links were added between related nodes.

Users navigate through the database using a variety of link types. For example a name link allows the user to jump to comments made by a particular speaker related to the current topic. Another type of link, control, provides navigational commands such as "browse," "scan," "more," and "continue." "Browse" transitions to the next summary node (topic), while "more" transitions to the next detail node (comment about the current topic).

Arons discusses "lessons learned about navigating in speech space." Some of his conclusions are:

- Streamline speech interactions by providing concise yet informative feedback. Arons relates this to Grice's four conversational maxims: be as informative as required, be relevant, avoid ambiguity, and be brief [Grice 1975].
- Make sure that all system actions are interruptible and that the system's response to the user's interruption is timely.
- A flat recognition vocabulary (all commands are always active) takes advantage of the goal directed nature of speech.

Hyperspeech, like Hyperphone, discusses important design considerations for voice-only interfaces. Both systems, unlike VoiceNotes, are composed of information authored by the designer of the system as opposed to the user. Audio data is pre-collected and manually segmented and organized, while VoiceNotes are entered and organized by the user. In addition, VoiceNotes also automatically records and segments audio data (see Chapter 8).

2.3 Voice Hand-held Computers

2.3.1 Braille 'n Speak

The Braille 'n Speak (Figure 2.1) is a portable notetaking device for the blind [Blazie 1991]. Information is input using a seven key Perkins Braille chorded keyboard and output using text-to-speech synthesis. The Braille 'n Speak has a non-hierarchical file system that allows the user to create and name ASCII text files. Users are given the ability to change the rate of playback, and the pitch of the spoken output. Whenever these speech settings are adjusted, feedback of the new setting is given in the form of synthesized speech output. For example, when the user changes the pitch, the Braille 'n Speak prompts "lower" or "higher" in the new pitch setting. The user can also activate and deactivate letter or word echoing. If letter echoing is active, the system echoes each letter that the user types. If word echoing is active, the system speaks each word after it is typed. Users can navigate within a file moving line by line or jumping to the top or bottom of the file for example. The system also provides feedback of the current position (e.g., "Top of file").

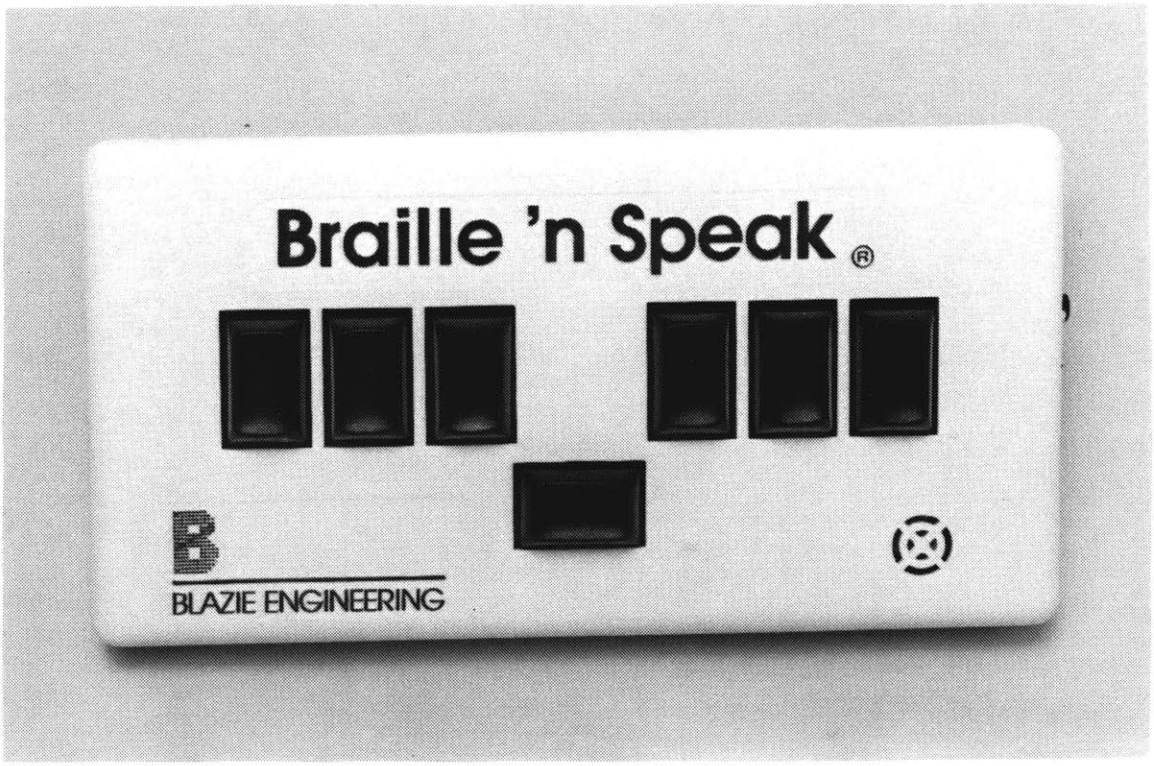


Figure 2.1: Braille 'n Speak: portable notetaking device for the blind. (Photo ©Barry Arons 1992).

The Braille 'n Speak comes with up to 640K of RAM. Using a 9-volt battery adapter, the device can be powered for up to five hours. In addition to word processing, it comes with a talking clock, calendar (files can be given a date and time stamp), calculator, and stopwatch.

The Braille 'n Speak is used by the blind community for taking notes at meetings, storing phone numbers and addresses, and for general word processing. The device does not only operate in isolation, but can be serially connected to a PC in order to download or upload data. It can also be connected to a modem or act as the keyboard for an IBM PC or compatible.

Although the Braille 'n Speak has been designed specifically for the blind community, it is an example of a very innovative portable computer. The device is small and light-weight, yet the chorded keyboard is large and the buttons are easy to press. The combination of chorded keyboard input and speech output provides portability. The data type, ASCII text, allows information to be searched and sorted and easily transferred to a desktop system. The Braille 'n Speak is also another example of an interface that provides navigational control without the use of a visual display.

2.3.2 Pencorder

The Pencorder (Figure 2.2) is a bulky ball-point pen with the ability to digitally record and playback two 20 second channels of speech [Machina 1992]. The pen is 5.5 inches long by 0.675 inches in diameter and weighs 2 ounces. There is a slider on the pen that selects between three different modes: play, record, and off. There is also a channel selector that selects between two recording channels. In order to record a voice note or message, the user must slide the mode selector to the record position and press the start/stop toggle button. An LED lights, indicating that recording has begun. The user presses the start/stop button a second time to terminate recording, or recording will terminate automatically when 20 seconds has elapsed. When recording completes, the LED is turned off. Recorded notes cannot be appended to ones already recorded. If another note is recorded on the same channel, the previous note is overwritten. Notes also get erased if the mode selector is in the off position for five minutes or more.

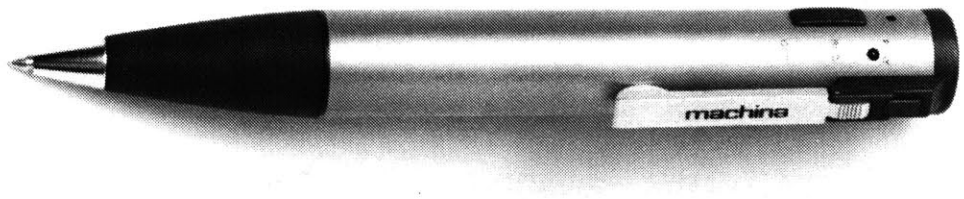


Figure 2.2: Pencorder: combination ball-point pen and voice notetaking device. (Photo ©Barry Arons 1992).

The Pencorder fails to be useful due to the limited memory capacity and lifetime for recorded voice notes. Since there is no ability to append to a voice note, the 20 seconds cannot be used to its full extent except when recording a complete 20 second segment. The short lifetime of notes is a particular problem. According to the manual, the pen can be used 4 to 6 times per day for at least 2 months, using 4 A76 alkaline batteries. This assumes that the user will turn the pen off frequently. Turning the pen off, however, causes the messages to be erased.

Even given its many limitations, the Pencorder demonstrates an interesting form factor for future portable voice notetaking devices since a pen is something many people carry with them at all times and can easily fit into a pocket. The idea for future "ubiquitous" computers is that they will be an "invisible" part of people's everyday lives just as pen and paper are today [Weiser 1991].

2.4 Notetaking Programs

2.4.1 NotePad

Notepad is a visual notetaking and activity management program [Cypher 1986]. Notepad is described as a tool for "thought-dumping—the process of quickly jotting down a flood of fleeting ideas" ([Cypher 1986] p. 260). Cypher emphasizes the importance of allowing users to quickly record or jot down an idea with a minimum amount of interference. Users are given the ability to title notes and file them into bins, but these tasks can be postponed so as not to interfere with the "thought-dumping" process. This is an important concept for any notetaking system, whether voice or text based.

The design of the Notepad system is based on ideas for "user-centered activity management." In particular, support is provided for dealing with activity interruption. For example, the user may be recording one idea, when another idea comes to mind. Using a jot command, the user can record the new idea without leaving the current one. This emphasizes the importance of maintaining the user's place amongst a set of notes or activities as interruptions tend to occur frequently.

2.4.2 ThoughtPattern

ThoughtPattern is a Macintosh graphical application for managing personal information such as ideas, meeting notes, and reminders [Bananafish 1991]. The system allows users to enter **items** (textual notes) and categorize them using **tabs** (text words or phrases). Each item can be categorized using one or more tabs. For example, the item "Discover Cardmember Service 1-800-858-5588" is categorized under the tabs "credit cards," "customer service," and "DISCOVER card." **Tab groups** (Figure 2.3) can also be created in order to provide easier management of a large number of tabs. For example, the tab group "Restaurants" might contain the tabs "Legal Seafood," and "Royal East." Users are also given the ability to filter items based on date, priority, and associated tabs. Tabs can also be displayed in a tab group window which provides an outline view that can be compressed or expanded (e.g., the user can select a tab group to see all associated tabs).

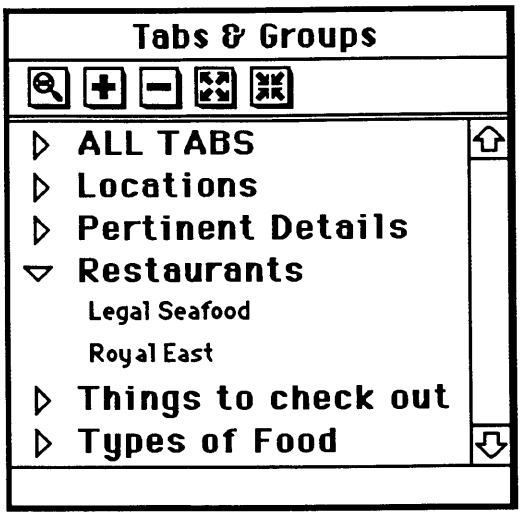


Figure 2.3: ThoughtPattern: The Tabs & Groups window shows the hierarchy of categories. Individual items cannot be displayed in this window.

ThoughtPattern provides two "views" for displaying information—list view and item view (Figure 2.4). The item view displays one item at a time, while the list view displays the entire list of tabs or item contents in a window.

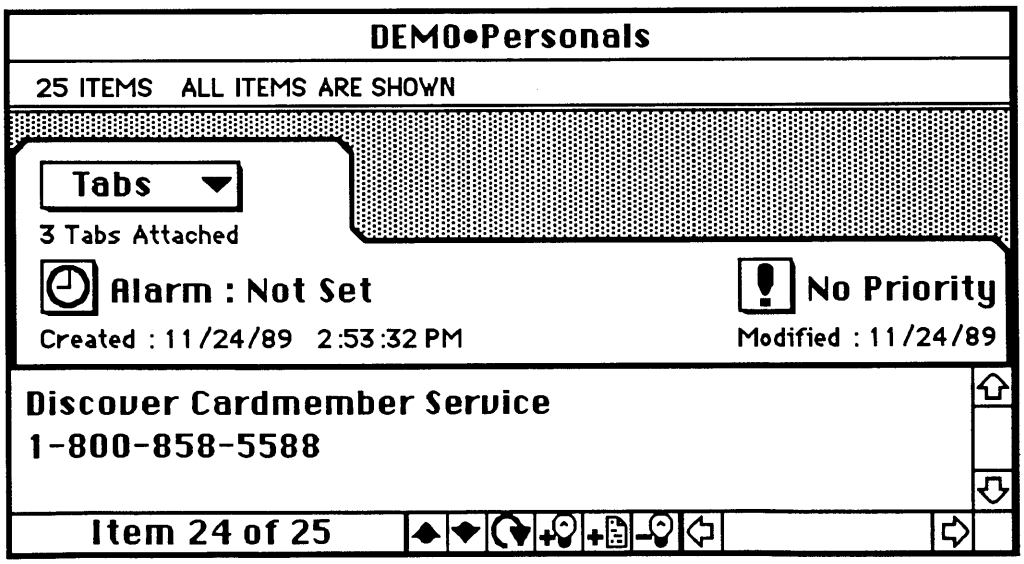


Figure 2.4: ThoughtPattern: Item View. The visual metaphor is a folder with a tab on it. Each item is displayed as a folder with a certain number of "tabs" or categories attached.

A reminding feature is also provided. An alarm can be set for a single date and time or can be daily, weekly, or monthly. "Ringing Alarms" allows the user to select any sound to be played for the reminder. "To-do lists" can also be created using the "silent alarms" option.

2.4.3 DayMaker

DayMaker is another personal organizing program for the Macintosh [Pastel 1991]. DayMaker and ThoughtPattern provide a similar set of features: priorities, to-do lists, date and time specifications, etc. Although the mechanism for entry and display of the information differs between the two applications, the conceptual design is very similar. For example, DayMaker also has the concept of views (e.g., list view, month view). Another important similarity is the use of the tabs concept which are known as **tags** in the DayMaker application. A hierarchy of tags can be created in a similar manner to ThoughtPattern's concept of tab groups. The visual metaphor is different, however. A tag is visually represented as the kind of tag that one would find attached to a piece of merchandise in a store (e.g., a price tag).

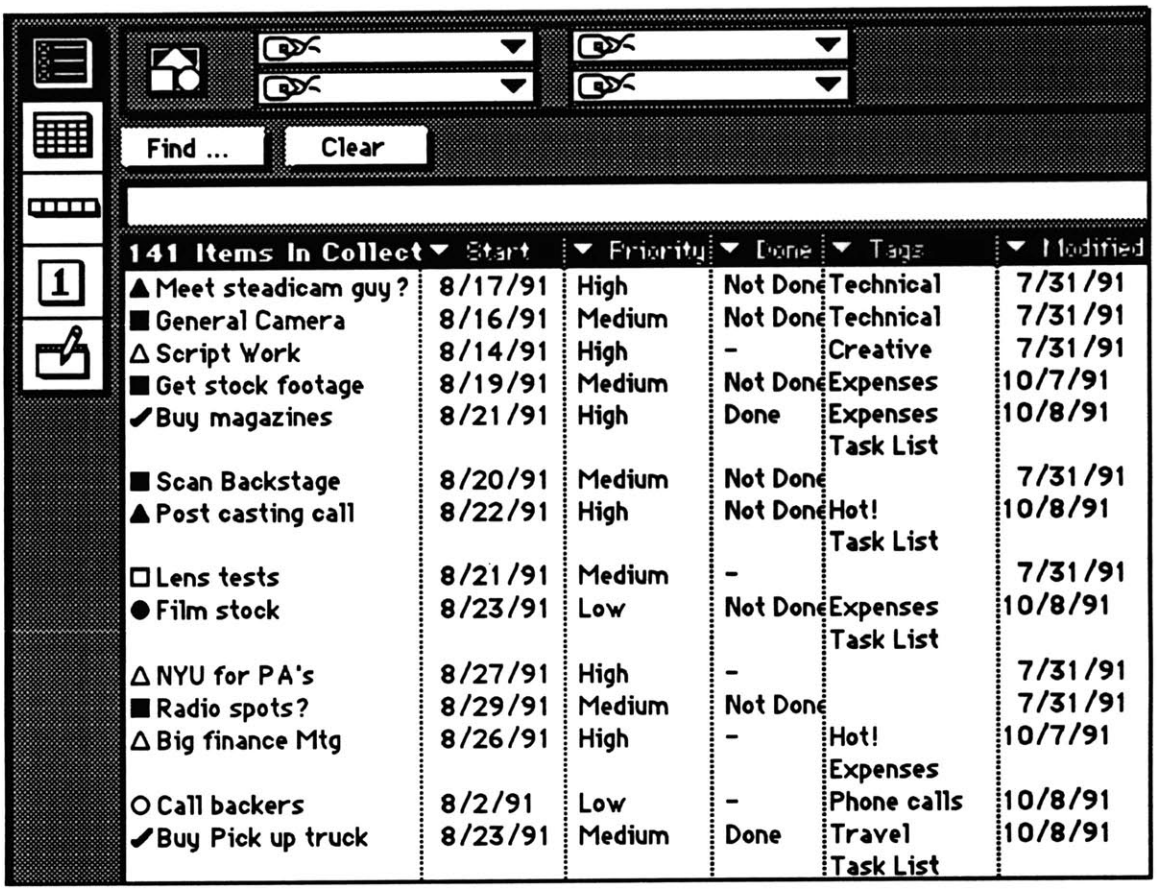


Figure 2.5: DayMaker list view.

Both systems provide the ability to browse and edit tabs/tags at a high level. In DayMaker this is provided with a browser window that displays different levels of tags in different columns while ThoughtPattern uses an outline format similar to the outline feature provided by MS-Word. Neither application allows the items associated with each tab/tag to be edited or moved from group to group within the outline window.

2.5 Summary

The following is a summary of the background research as it relates to the development of the VoiceNotes application. In developing VoiceNotes, an attempt has been made to build on the findings made by prior research and to overcome some of the limitations in the current technology.

- Current personal recording technology does not provide users with random access to voice data that they have recorded. Given the ability to "mark" segments of audio, users would like control over the number and meaning of these marks as well as the ability to access the data directly from a portable device. VoiceNotes addresses this issues by providing users with the ability to create personal categories, enter, review, and organize audio data directly from a hand-held device.

- Voice-only hypermedia systems such as Hyperphone and Hyperspeech provide insights into navigating through a voice database without a visual display. VoiceNotes differs from these systems in that it deals with recorded voice data that is authored and organized by the user, not the system designer.

- The Pencorder is limited due to the small amount of storage and short lifetime for recorded notes. The intention of VoiceNotes is to provide the ability to create and manage a large number of notes classified into many categories with an unlimited lifetime. However, a pen may be a desirable form for a future "ubiquitous" voice-controlled hand-held computer.

- The Braille 'n Speak achieves portability through the use of chorded keyboard input and synthetic speech output. VoiceNotes uses speech and button input rather than a chorded keyboard. In addition, VoiceNotes uses stored voice instead of ASCII text as a data type, and digitized rather than synthetic speech output.

- Notepad, ThoughtPattern, and DayMaker provide examples of terminology, metaphors, and capabilities provided by visual notetaking programs. The underlying concepts in each of these applications are similar to those used by VoiceNotes (notes can be organized using bins/tabs/tags/names). VoiceNotes, like Notepad, is intended to allow "thought-dumping," so the interactions must be efficient—the "tool" should not impede the user's thought process. However, the considerations for designing a voice interface are very different from those for designing a visual interface. VoiceNotes must be more efficient in its presentation of information since voice is slow and serial, while a visual interface can present information simultaneously in a multitude of windows. The goal of VoiceNotes is also to provide a simple, yet powerful interface. Therefore, an attempt is made to only support the most essential functionality, while screening out a host of minor features that might only serve to dilute the power of the interface.

3. VoiceNotes Conceptual Design

This chapter presents an overview of the VoiceNotes application and hand-held hardware. Basic concepts such as list structure and terminology are presented at an overview level in order to lay the framework for the detailed interface design discussions that follow in Chapter 4. A description of the hand-held device is provided in order to allow the reader to "picture" how this device might be used. In addition, a taxonomy of voice notes is described along with scenarios showing possible uses of VoiceNotes in everyday situations.

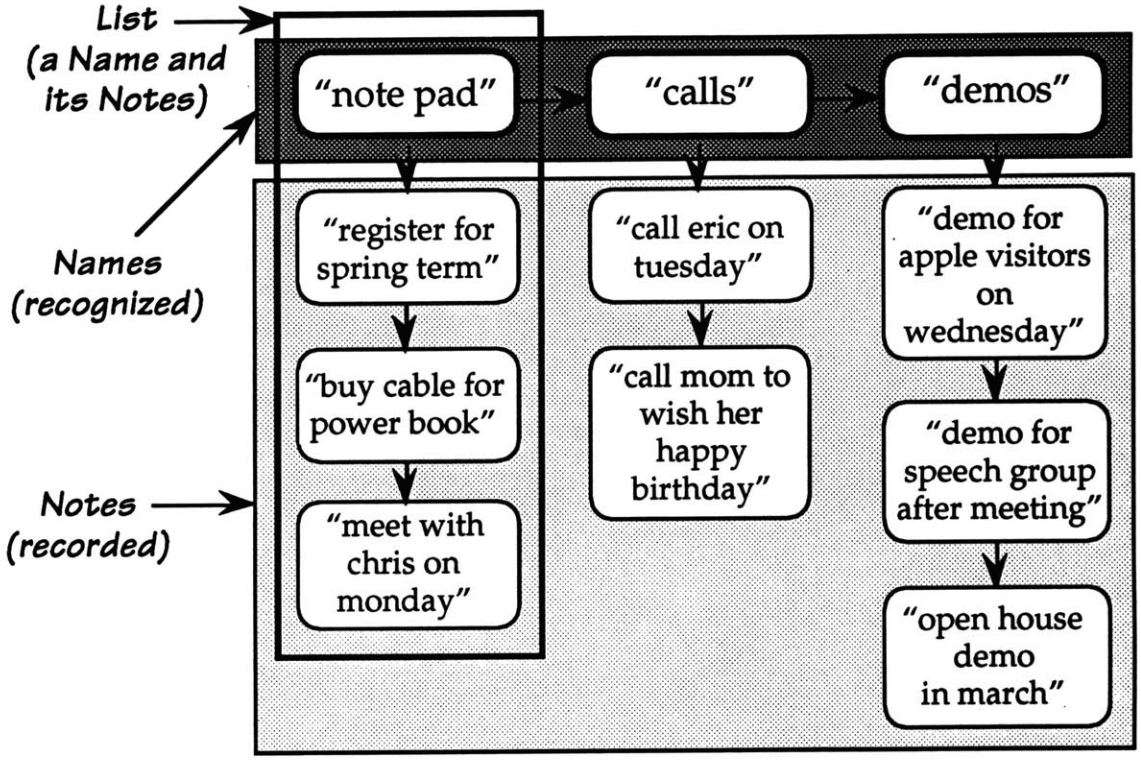


Figure 3.1: VoiceNotes list structure. A list is composed of a name and notes. Notes are recorded segments of speech. Names are both recorded segments of speech and voice commands. The set of names also composes a list.

3.1 VoiceNotes List Structure and Terminology

VoiceNotes provides a simple digital audio file system (Figure 3.1). The user first creates a list **name**. This acts as a method for categorizing and a handle for accessing a collection of **notes**. List names are made up of a single utterance composed of one or two words. The name is spoken by the user and then simultaneously recorded for playback and trained for voice recognition. List names provide random access across lists; in order to select and playback a list of voice notes, the

user simply speaks a list's name. Once a name is created, notes can be recorded into the list. The set of names also constitutes a list. All operations that can be performed on a notes list can be performed on the names list. Voice or button commands always apply to the list that is currently selected.

3.2 VoiceNotes: A Demonstration

A demonstration of the final VoiceNotes design is provided here so that the reader may better understand the design discussions that follow. Many design iterations were performed before this design was reached. In addition, see Chapter 6 for a more detailed presentation of the button interface.

The first time VoiceNotes is used, there is one list name that is automatically created for the user. The **notepad** list is provided so that the user may begin to record voice notes immediately without having to create a list name first.

VoiceNotes at startup:	
Hand-held:	"Starting in *Notepad*"

Figure 3.2: When the VoiceNotes application is launched, the user is placed in the notepad list. Automatic creation of the notepad list allows first time VoiceNotes users to begin recording voice notes immediately upon starting the application.

In order to playback all of the list names, the user can either speak the "*NAMES*" command or slide the list selector button on the hand-held to the names position (see section 6.1.5).

User:	"*NAMES*"
Hand-held:	"Moving into *names*"
	"*Notepad*," "*Calls*," "*Demos*"
	"Done"

Figure 3.3: In order to select and playback the list of names the user issues the "*NAMES*" command. In this example, the user has three lists: notepad, calls, and demos. The list name acts as a handle for accessing the voice notes on a list. Whenever a list has finished playing, the system responds "Done."

In order to playback the notes on a particular list, the user can speak the name of the list or slide the list selector button into the notes position when the desired list name is heard (see section 6.1.5).

User:	"*DEMOS*"
Hand-held:	"Moving into *Demos*"
	"*Demo for Apple visitors on Wednesday*"
	"*Demo for Speech Group after meeting*"
	"*Open house demo in March*"
	"*National Semiconductor demo*"
	"Done"

Figure 3.4: In this example, speaking the list name command "*DEMOS*" causes demos to become the current list and all of the voice notes in the list to be played back from beginning to end if not interrupted by another user command.

The user may interrupt the spoken output of the list at any time to stop playback, repeat a note, or move to the previous, next, first, middle, or last note on the list.

User:	"*FIRST*"
Hand-held:	"*Demo for Apple visitors...*"
User Interrupts:	"*LAST*"
Hand-held:	"*National...*"
User Interrupts:	"*STOP*"
Hand-held:	"Stopped"

Figure 3.5: The "*FIRST*" voice command plays the first item in the current list. The first note in the "*Demos*" list is "Demo for Apple visitors on Wednesday" and the last note is "*National Semiconductor demo*."

The user can add a note to the end of the current list by speaking the "*RECORD*" or "*ADD*" command or by pressing the *RECORD* button on the hand-held.

User:	"*RECORD*"	
Hand-held:	"Recording note"	LED is turned on
User:	"*Show VoiceNotes to Victor Zue on Thursday at 2*"	LED is turned off
Hand-held:	"Note added to *Demos*"	

Figure 3.6: After the user speaks the "*RECORD*" command, speech is recorded until a pause is detected. The user is given feedback as to what is being recorded (a name or a note). In addition to digitized speech feedback, an LED lights indicating when the user may begin speaking. Once the user stops speaking, the LED is turned off.

The user can add a new list name by selecting the names list and recording a new name. Recording a name is accomplished in the same manner as recording a note except that names can

be only one or two words and notes can be up to 20 seconds long. Names are both recorded and trained for recognition. Since VoiceNotes uses an isolated word recognizer[2], names must be short isolated utterances. The system does not have a textual representation of each list name, only the stored audio data.

User:	"*NAMES*"	
Hand-held:	"Moving into *Names*"	
User:	"*RECORD*"	
Hand-held:	"Recording name"	LED is turned on
User:	"*Shopping*"	LED is turned off
Hand-held:	"New name added"	

Figure 3.7: The user's utterance of the word "*Shopping*" is simultaneously digitized for playback and trained for recognition. The digitized recording of the list name is associated with a voice template used for recognition.

The user can now select the shopping list simply by speaking its name and record a list of shopping related voice notes.

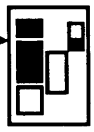
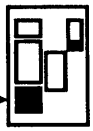
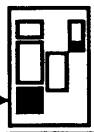
	User:	"*SHOPPING*"	
	Hand-held:	"Moving into *Shopping*"	
		"list is empty"	
Record → 	User:	Presses *RECORD*	
	Hand-held:	"Recording note"	LED is turned on
	User:	"*Lettuce*"	
Stop → 	User:	Presses *STOP*	LED is turned off
	Hand-held:	"New note added"	
Record → 	User:	Presses *RECORD*	
	Hand-held:	"Recording note"	LED is turned on
	User:	"*Salad dressing*"	
Stop → 	User:	Presses *STOP*	LED is turned off
	Hand-held:	"New note added"	

Figure 3.8: After the list name "*Shopping*" has been recorded, it can be used as a voice command for moving into (selecting) the shopping list. Once the shopping list has been selected, the user can begin recording notes (or deleting, playing, etc.). In this example, the *RECORD* button is used to initiate recording, and the *STOP* button to terminate recording. Each time another note is recorded, it is added to the end of the shopping list.

[2]VoiceNotes uses a Voice Navigator speech recognizer made by Articulate Systems, Inc. See section 5.1.3 for a discussion of recognizer selection.

3.3 Description of Hand-held Prototype 1

The Speech Research Group is developing several hardware prototypes of a hand-held computer.[3] The first such prototype acts as a remote front-end to a Macintosh workstation that has speech recognition, synthetic speech, and speech coding capabilities. All speech data is stored on the workstation. The body of the hand-held device is that of a commercial microcassette recorder. The original contents of the microcassette recorder were replaced by a Motorola 68HC11 micro controller. The buttons on the original recorder were retained and connections were made from the 6811 to the microcassette recorder's PC board on which the switches are mounted. A program downloaded to EEPROM sends a two byte code via a serial connection when any of the buttons are pressed, and can also receive a two byte message for turning on or off a single LED. In addition, an A/D converter is used to read the value of a potentiometer. Each time the potentiometer's state changes, a two byte message is sent to the Macintosh; one byte indicates a potentiometer change and the other indicates the current value. The original microphone element was replaced by an omni-directional microphone,[4] and a pre-amplifier was installed, while the original speaker was retained. A "tether"—a single cable containing 8 wires (2 analog audio input, 2 analog audio output, 2 serial, 1 power, and 1 ground) connects the hand-held to a "black box."

The black box splits the audio output from the hand-held into two outputs, each going through an isolation transformer, and controlled by separate potentiometers. The purpose is to allow the audio output signal from the hand-held to be routed to the Voice Navigator for recognition and the MacRecorder for digitization.[5] The black box also routes audio output from the Macintosh to the audio input on the hand-held unit to allow audio output from the Macintosh to be played out of the hand-held's speaker. A mono headphone can be plugged into the hand-held for private listening. An additional microphone input was added to the black box to allow a separate microphone (e.g., standard head-mounted noise-canceling microphone) to be used instead of the microphone built into the hand-held device. The black box also contains a serial connector for communicating with a back-end processor (in this case, a Macintosh).

Hand-held prototype 1 acts as an affordance (i.e., the thing that you touch and feel that activates the functionality behind it). The function of this prototype was to allow initial exploration of a voice-controlled hand-held computer with the ultimate goal of eliminating the tether. As such, future hand-held prototypes under development include: an autonomous hand-held unit and a wireless version of prototype 1. The current VoiceNotes application has been implemented with the prototype 1 hand-held hardware.

---

[3]The first prototype was built by Derek Atkins and Barry Arons of the Speech Research Group.

[4]Although the microphone acts somewhat like a directional microphone because it is mounted inside the casing of the microcassette recorder.

[5]The MacRecorder is not needed when using a Macintosh computer that comes with a built-in sound input device.

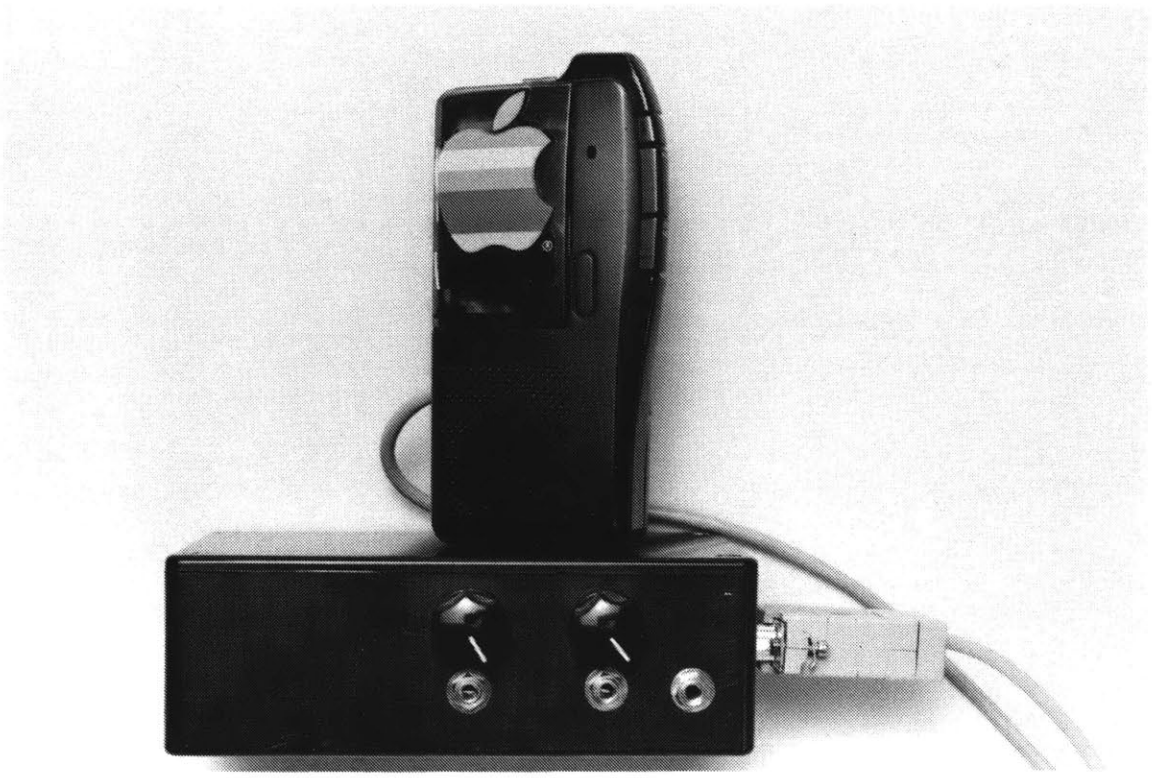


Figure 3.9: Photograph of hand-held prototype 1 and the hand-held "black-box" that connects it to the Macintosh, MacRecorder, and the Voice Navigator. (Photo ©Barry Arons 1992).

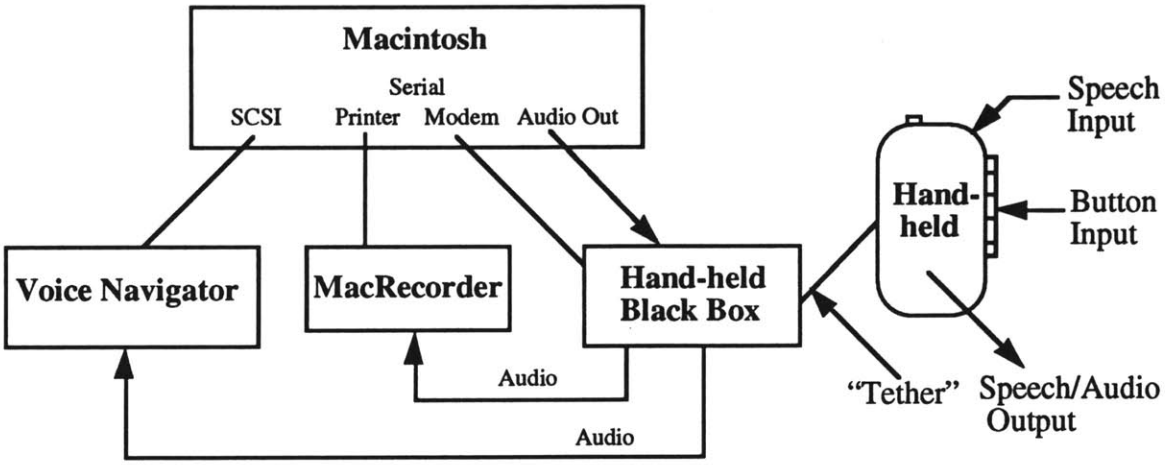


Figure 3.10: Schematic diagram of the hand-held hardware configuration. The user records a voice note by speaking into the hand-held's built-in microphone. This speech input goes into the hand-held "black box" where it is then routed to the Voice Navigator for recognition and the MacRecorder for digitization. Button input goes from the hand-held to the "black box" to the modem port on the Macintosh. Audio output from the Macintosh is routed to the hand-held "black box" and then to the speaker on the hand-held device.

Figure 3.11: Photograph of hand-held prototype 1. (Photo ©Barry Arons 1992).

3.4 Taxonomy of Voice Notes

Voice notes can be separated into two categories: those **intentionally** recorded and those captured **on-the-fly** (Figure 3.12). Intentionally recorded voice notes further break down into those that have a **random** order versus those that are ordered **linearly**.

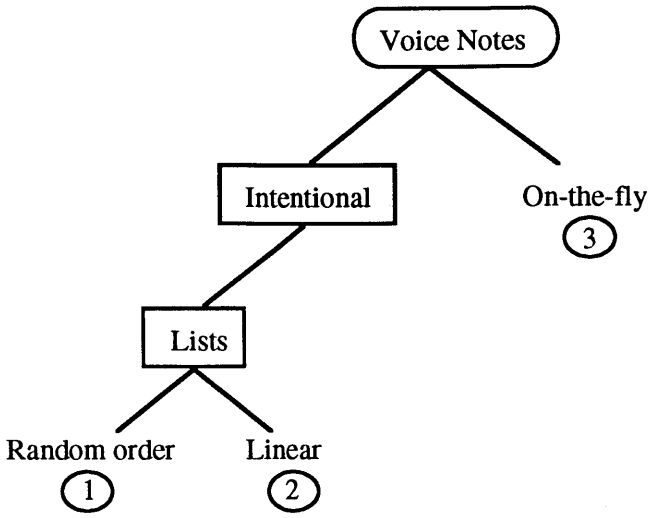


Figure 3.12: Taxonomy of voice notes: The VoiceNotes application addresses three types of voice notes, (1) Randomly ordered lists, (2) Linear lists, and (3) On-the-fly.

For each type of voice note, a scenario is provided to demonstrate possible uses and the interaction provided by the VoiceNotes application.

3.4.1 Randomly Ordered Lists of Voice Notes

Intentionally recorded voice notes refer to when the user knows in advance that there is a particular thought, idea, or reminder that s/he needs to record. For example, "do the laundry" is an example of a short reminder that could be recorded as a voice note.

There are several categories of intentionally recorded voice notes. One category of voice notes are those that can be managed in lists such as: to-do lists, lists of project ideas, and shopping lists. The "do the laundry" note might be one item on a list of household chores. A characteristic of these lists is that they have a relatively random order.

Time:	Morning
Location:	At home before leaving for work
Task:	Retrieving the day's activities and adding new ones
User:	"*DEMOS*"
Hand-held:	"Moving into *Demos*" "*Demo for Speech Group after meeting this week*" "*Demo for Apple Visitors at 12:30*"
User interrupts:	"*APPLE*"
Hand-held:	"Moving into *Apple*": "*Make posters for the open house*" "*Schedule a conference call with Eric*" "Done"
User:	"*RECORD*"
Hand-held:	"Recording note"
User:	"*Call Gitta Salomon to discuss her paper*"
Hand-held:	"New note added to *Apple*"

Figure 3.13: Scenario part 1, Randomly ordered lists of voice notes.

Time:	Later that evening
Location:	Car
Task:	On the way home from work and want to retrieve personal items
User:	"*NAMES*"
Hand-held:	"Moving into *Names*" "*Apple*," "*Demos*," "*Personal*"
User:	"*PERSONAL*"
Hand-held:	"*Buy birthday card for Mom*" "*Drop off rent check*" "Done"

Figure 3.14: Scenario part 2, Randomly ordered lists of voice notes.

3.4.2 Linear Lists of Voice Notes

Another category of voice lists are those that are linearly ordered such as directions or step-by-step instructions. For example, driving directions fall into this category since each instruction must be followed in order until the desired location is reached.

All voice notes are in fact ordered linearly. When a note is recorded, it is added to the bottom of the current list of voice notes. A linear list structure supports both linear and random lists of voice notes. Both voice and button commands support linear movement through a list (*PREVIOUS*,

NEXT). Voice commands support random access within a list ("*FIRST*," "*MIDDLE*," "*LAST*") and between lists (e.g., "*SHOPPING*," "*DEMOS*").

Time:	Afternoon
Location:	Car
Task:	Driving to Apple
	Directions have been recorded on the hand-held
User:	"*Directions to Apple*"
Hand-held:	"Moving into *Directions to Apple*"
	"*Take 280 South*"
	"*Get off at the Saratoga/Sunnyvale exit*"
User interrupts:	"*STOP*"
Hand-held:	"Stopped"
	stops playing and waits for the user's next command
10 minutes later:	
User:	"*PLAY*"
Hand-held:	"*Get off at the Saratoga/Sunnyvale exit*"
	"*At the end of the ramp turn right onto Saratoga/Sunnyvale*"
	"*Go through approximately 3 or 4 traffic lights until you reach Steven's Creek Boulevard*"
User interrupts:	"*PREVIOUS*"
Hand-held:	"*At the end of the ramp turn right...*"
User interrupts:	"*NEXT*"
	"*At Steven's Creek, turn right*"
	"*Building is on the right*"
	"Done"
User:	"*REPEAT*"
Hand-held:	"*Building is on the right*"

Figure 3.15: Scenario, Linear lists of voice notes.

3.4.3 Voice Notes Captured On-the-fly

On-the-fly voice notes refer to instances when, upon hindsight, the user realizes that something important has been said, and subsequently wants to capture this information (Figure 3.16). For example, during a brainstorming session in which two people are discussing a paper they are authoring, one person may suggest the wording of a sentence that turns out to be "just right," but neither person can remember exactly what was said. These types of voice notes are described in detail in Chapter 8.

Time:	12 Noon
Location:	Office hallway
Situation:	Geek passes Lisa in the hallway

Lisa to officemate:	"I'm going to go photocopy something, I'll be back in a minute"
Lisa:	walks through the hallway towards the photocopier...
Geek:	"Hi Lisa, I was just looking for you, I found a bug in voice mail. After listening to the menu, if you press the 2 key and then immediately begin entering the name and then hit the pound key to cancel, it exits you instead. Can you take a look at this? Thanks, cheerio."
Lisa:	(caught by surprise, Lisa has not had a chance to write any of this information down. Luckily, the hand-held computer she was carrying in her pocket, captured the information.)

Lisa to Hand-held:	"*CAPTURE*"
Hand-held:	"Moving into *capture*"
	"*I'm going to go...*"
Lisa interrupts:	Presses the *NEXT* button
Hand-held:	"*Hi Lisa...*"
Lisa:	"*BEGIN-MARK*"
Hand-held:	"Beginning marked as *Hi Lisa...*"
Lisa:	"*LAST*"
Hand-held:	"*Thanks, cheerio*"
Lisa:	"*END-MARK*"
Hand-held:	"End marked as *Thanks, cheerio...*"
Lisa:	"*MOVE*"
	"Move to what list?"
Lisa:	"*Notepad*"

Figure 3.16: Scenario, Capturing voice notes on-the-fly.

4. VoiceNotes Interface Design

The VoiceNotes interface was developed through an iterative design process. VoiceNotes provides a structure and an interface for recording, navigating, and managing lists of recorded segments of speech. Each of these aspects of the interface went through a careful design process.

The design issues presented in this chapter are those that apply to both voice and button input. Chapter 5 goes on to discuss issues relating specifically to voice input, and Chapter 6 discusses button input. In Chapter 7, a hybrid voice and button interface is discussed.

There were two major phases in the design of the VoiceNotes interface. Phase 1 of the design was a "moded" interface. In Phase 2, a "modeless" design was adopted and further iterations to the VoiceNotes list management interface continued from this point. The final voice notes interface design is presented in section 3.2, VoiceNotes: a Demonstration.

4.1 Design Criteria

- User-definable categories. Users must be given the ability to create their own list names. The ability to personalize the application is important. Since a list name might contain the name of a friend or colleague, an acronym, or company name, for example, list names cannot come from a fixed vocabulary.

- Timeliness. The entry of the spoken item must occur early in the user interaction for recording a voice note without many steps being required. Ades and Swinehart [Ades 1986] found this to be especially important for voice annotation so users do not lose their trend of thought. Voice notes may be fleeting thoughts and will be forgotten if not recorded immediately.

- Portability. Voice notes are spontaneous segments of the user's speech. For example, on the way home from work the user may recall several tasks that need to be addressed the following day at work. The interface must support all aspects of interacting with voice notes from outside the office—in locations such as a car, airport, or noisy restaurant where users may want to record and access their voice notes.

- Use with one hand. An important characteristic of audio is that it provides "a means to capture an idea even when physical constraints make writing impossible" ([Degen 1992] p. 413). In order to support use of VoiceNotes from a variety of locations, the interface should not require more than one hand to be operated at any time.

- Ability to interrupt. The spoken output must be immediately interruptible at all times. Due to the slow and serial nature of speech, it is essential to provide the user with the ability to jump between notes on a particular list, between different lists, or to stop the playback of a list at any instant. In addition, there is a high probability that the user will be interrupted during the task of entering or playing back a list of voice notes (telephone rings, baby starts to cry, someone else enters the room, etc.). Therefore, it is essential that users be able to stop playback while maintaining their current position in a list.

- Always listening. The recognizer should always be listening for the user's input unless the user specifically requests that the recognizer "stop listening." This is important for the interface to be natural and responsive. The user should not have to push a button in order to issue a voice command.

- Voice or buttons. The application should be completely operable using either voice or button input in isolation. This does not mean that every voice command must have a button equivalent or that every button command must have a voice equivalent. It does mean that the basic set of voice notes functions (recording, navigating, playing) must be accessible by voice or button input.

4.2 Design of VoiceNotes Terminology

4.2.1 Notes and Names

Notes are recorded digitized segments of the user's speech. Notes are organized into lists, each with its own list **name**.

Names were originally referred to as **tags**, and notes as **items**. Each item had an associated tag. Items with the same tag made up a list. When VoiceNotes was first demonstrated, some people had difficulty understanding the term tag. The original concept of a tag was that each item could have one or more tags associated with it. This is similar to the use of the term tags in the DayMaker application described in Chapter 2. For example, the voice note "*Do the laundry*" might be given the tags "*Personal*" and "*Chores*." Due to the difficulty of navigating without a visual display, a simplified design is desirable. In the current design, each item has only one associated tag and therefore belongs to a single list.

In this design, a tag becomes more like a directory, or a folder containing a collection of items. For this reason, the word **folder** was proposed as a replacement for the word tag. The rationale was that Macintosh users are accustomed to the concept of a folder and this would help them to better understand the interface. In addition, MS-DOS and UNIX users are familiar with the concept of a directory. The problem is that a folder is a visual metaphor used by the Macintosh. Using the word folder may have created false expectations of the interface's capabilities. For example, VoiceNotes uses a simple one-level hierarchy so it is not possible to create a folder within a folder as can be accomplished under the Macintosh hierarchical file system.

In order to avoid conflicting with the Macintosh's definition of a folder, and still replace the word tag, the term **list names** was chosen. To further simplify the term for voice recognition purposes, list names was shortened to just **names.**

The term item was found to be too general since a list item could be a name or a voice note. Therefore, items came to be referred to as notes. Users also had difficulty distinguishing when they were in a list of names or notes. By referring to the items more specifically (as names or notes), this confusion might be resolved.

Even though the original terms tags and items were used throughout much of the early designs of the VoiceNotes interface, the new terms, names and notes will be used throughout the remainder of this document to avoid confusing the reader.

4.2.2 The Notepad List

When presenting VoiceNotes to some users, the first thing they tried to do was record a voice note. It was confusing to have to first create a list before any voice notes could be recorded. The purpose of the notepad list is to allow first time users to immediately begin to record voice notes. Once a collection of notes has been recorded, the goal may then be to categorize these notes by giving them a name. To facilitate first time use, the notepad list is automatically created for the user. When the application first starts up, the user is placed inside the notepad list, where they can immediately begin recording voice notes.

The notepad provides a default or scratch list for the user to begin recording notes. In deciding what to call this list, the following additional names were considered: initial list, default list, miscellaneous, temp list, clipboard, and scratch. Clipboard was ruled out for the same reason that folder was not used—due to its specific meaning on the Macintosh. Scratch could not be used, since this is often used to mean cancel or delete. For example, Articulate Systems provides the command "Scratch that" with many of the Voice Navigator standard vocabularies to allow the user to cancel a previously recognized command [ASI 1991]. The term notepad was selected since it seemed to connote a place for a random collection of notes.

4.3 Phase 1: Moded vs. Modeless

In Phase 1 of the design process, the VoiceNotes interface was moded. Moded interfaces are designed such that only a subset of all available actions is valid at any given time. The valid subset is dependent on the current mode that the system is in. Many argue against moded designs because they require the user to be aware of the current mode and to understand the constraints it places on the actions that can be taken. Since each user's mental model of the system is likely to differ from that of the designers, moded designs tend to be problematic. Another source of confusion is that the same command can have different meanings depending on the mode. In defense of moded designs, it may be the case that subsetting the set of actions simplifies the interface and makes it clearer to the user the action that needs to be taken next.

The following is a comparison of a moded and modeless design for the VoiceNotes application. Advantages and disadvantages for each design are presented and discussed. As a result of this analysis, the original moded interface was redesigned to be modeless, and further design iterations continued from this base point.

4.3.1 VoiceNotes—Moded Design

The moded design has three modes: top level, name, and note. When the program first begins, the user is in the top level mode of the interface. In each of the modes (top level, name, and note) only a subset of the commands is valid and commands may have different meanings depending on the mode (Figure 4.1).

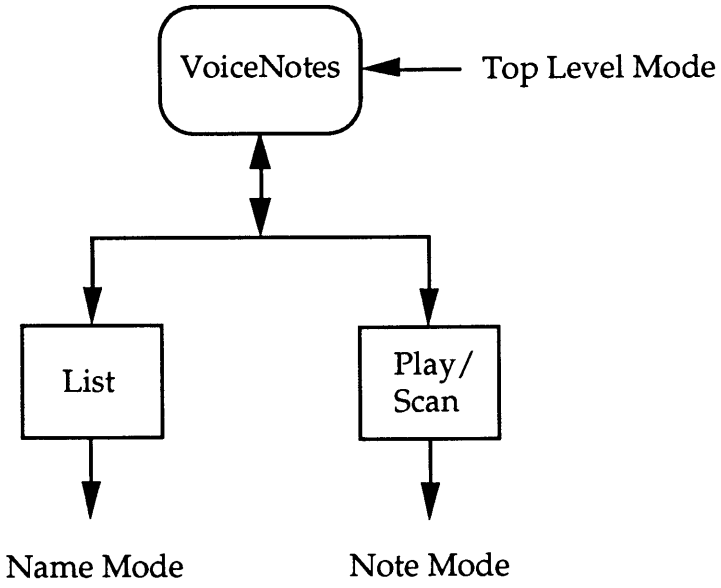


Figure 4.1: Three modes of the moded interface design: top level, name, and note.

4.3.1.1 Top Level Mode

The valid commands in this mode are "*PLAY*," "*SCAN*," "*LIST*," and "*TAKE-A-NOTE*."[6]

"*PLAY*" causes all of the notes associated with a particular name to be played. In the top level mode, the system response to the "*PLAY*" command is: "Play what list?" at which point, the user must say the name of the list to be played (Figure 4.2).

[6] At this point in the design, the interface was "voice-only"—all commands were issued using voice input. Button input was added in Phase 2 of the design process.

User:	"*PLAY*"
Hand-held:	"Play what list?"
User:	"*CALLS*"
Hand-held:	"Playing *calls*"
	"*Call Eric on Tuesday*"
	"*Call mom to wish her happy birthday*"
	"Done"

Figure 4.2: Top level mode: playing a list.

The "*SCAN*" command plays the first 1.0 seconds of each note on a list without pausing after each note. As with the "*PLAY*" command, the system response to the "*SCAN*" command is: "Scan what list?" at which point the user must say the name of the list to be scanned.

Two different commands are used for playing a list. While the *PLAY* command is used to play a list of notes, the "*LIST*" command is used to play the list of names (Figure 4.3).

User:	"*LIST*"
Hand-held:	"*Notepad*," "*Calls*," "*Trips*," "*Demos*"
	"Done"

Figure 4.3: Top level mode: The "*LIST*" command plays all of the names that the user has created.

The "*TAKE-A-NOTE*" command records a single voice note and prompts for the associated list name. The system first prompts the user to record the note. When a pause is detected, the system prompts "What list?" at which point the user may say an old or a new list name. In this design an attempt is made to determine whether the user's response is a new list name or a name that already exists (Figure 4.4).

User:	"*TAKE-A-NOTE*"
Hand-held:	"Recording"
User:	"*Dinner with Ian and Michael on Tuesday night*"
Hand-held:	"What list?"
User:	"*Open house*"
Hand-held:	"New name added"

Figure 4.4: After receiving the "*TAKE-A-NOTE*" command, the system first prompts the user for the note and second for the associated list name. If the list name does not already exist, the system feedback is "New name added." If "*Open house*" was an old list name, the system would have responded "Note added to *Open house*."

4.3.1.2 Name and Note Modes

After issuing the "*LIST*" command, the user is placed in name mode. In this mode, *PLAY* and *SCAN* become target style commands—they apply to the current list name that is being played (Figure 4.5).

User:	"*LIST*"
Hand-held:	"*Demos,*" "*Apple*"...
User:	"*PLAY*"
Hand-held:	"Playing *Apple*"...

Figure 4.5: Name mode: In this example, the user selects the "*Apple*" list to be played by uttering the "*PLAY*" command upon hearing the list name "*Apple.*"

"*TAKE-A-NOTE*" also becomes a target style command in name mode. The system associates the newly recorded voice note with the current list name (Figure 4.6).

User:	"*LIST*"
Hand-held:	"*Demos,*" "*Apple*"...
User:	"*TAKE-A-NOTE*"
Hand-held:	"Recording into *Apple*"...

Figure 4.6: Name mode: In this example, the user adds a note to the "Apple" list by uttering the "*TAKE-A-NOTE*" command upon hearing the list name "*Apple.*"

When the names list has finished playing, the system leaves name mode and returns to the top level mode (Figure 4.7).

User:	"*LIST*"
Hand-held:	"*Demos,*" "*Apple*"
	"Done"

Figure 4.7: The "*LIST*" command switches the system to name mode. When the list has finished playing ("Done") the system returns to the top level mode.

After issuing the "*PLAY*" or "*SCAN*" command, the user is placed in note mode. In this mode, commands apply to the current list of notes that is being played. As in name mode, when the list has finished playing, the system responds "Done" and returns to the top level mode (Figure 4.8).

User:	"*PLAY*"
Hand-held:	"Play what list?"
User:	"*APPLE*"
Hand-held:	"Playing *Apple*"...
	"*Demo for Apple Visitors on Wednesday*"
	"*Demo for Speech Group after meeting*"
	"Done"

Figure 4.8: In this example, once the *Apple* list starts playing, the system is in note mode. When the list has finished playing, the system returns to the top level mode.

The valid commands in note mode are the same as in name mode, except that commands apply to the list of notes being played instead of the list of names. In both note and name mode, additional commands are activated (Figure 4.9).

Command	Definition
"*NEXT*"/"*PREVIOUS*"	skips to playing the next/previous name/note on the list
"*TOP*" /"*MIDDLE*" / "*BOTTOM*"	plays the first, middle, or last name/note on the list
"*PAUSE*"	pauses playing the list of names/notes and maintains position in the list
"*CONTINUE*"	continues playing the list beginning with the name/note that was playing when the pause command was issued
"*REPEAT*"	replays the name/note that was just played
"*ADD*"	adds a new name/note to the list
"*DELETE*"	deletes the current name (and all associated notes) or the current note
"*STOP*"	stops playing the current list of names/notes and returns to the top level mode
"*MORE*"[7]	replays the current name/note entirely

Figure 4.9: Name and Note modes: In addition to the "*PLAY*," "*SCAN*," "*LIST*," and "*TAKE-A-NOTE*" commands, the above additional commands are valid in name and note modes.

[7]The "*MORE*" command was intended to be used when scanning a list. When an item of interest is heard, the user issues the "*MORE*" command, to hear the item played all the way through. This command was eliminated since the "*REPEAT*" command serves the same purpose.

4.3.2 VoiceNotes—Modeless Design

In a modeless design the user is always inside a list of names or notes (Figure 4.10). All commands are active at all times and always have the same meaning (Figure 4.11). However, because notes and names differ, it may not be possible to create a completely "modeless" interface. Being inside a list of names versus a list of notes may be considered a mode but the operations a user can perform are the same.

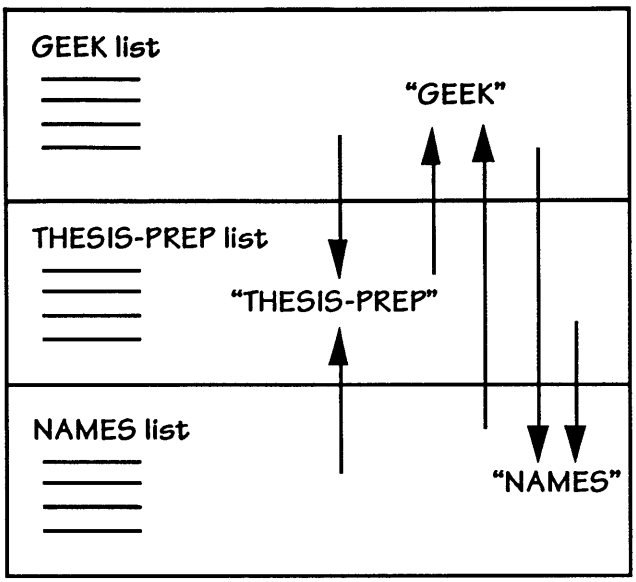


Figure 4.10: Pictorial representation of moving between lists in the modeless voice notes design. In this design there is always one list that is currently active.

The key element in making the interface modeless was in allowing the names list to be directly accessed with the "*NAMES*" command. In this way, instead of requiring a special command ("*LIST*") to play the list of names, the user can treat the names list the same as any other list. To select any list, the user simply utters its name. To play or record into a list, the user simply selects the list and then issues the "*PLAY*" or "*RECORD*" command. When inside the names list, the user can record a new name. When inside a list of notes, a new note can be recorded.

Command	Definition
"*<NAME>*"	*NAME* is the name of any list recorded by the user. For example, issuing the command "*DEMOS*," makes demos the currently active list (upon which all further operations are applied).
"*NAMES*"	Selects the names list as the currently active list.
"*RECORD*" / "*ADD*"	Records a new item and places it on the end of the current list. When inside the names list, a new name is recorded. When inside a list of notes, a new note is recorded.
"*DELETE*"	Deletes the current item in the list.
"*PLAY*" / "*SCAN*"	Plays/Scans the currently active list from the top.
"*NEXT*" / "*PREVIOUS*"	Skips to playing the next/previous item in the currently active list.
"*TOP*" / "*MIDDLE*" / "*BOTTOM*"	Plays starting from the top/middle/bottom item in the currently active list and continues forward through the list.
"*REPEAT*"	Repeats the current item in the list and continues forward through the list.
"*STOP*" / "*PAUSE*"	Pauses playback of the current list.
"*CONTINUE*"	Continues playback, starting at the current item in the list.
"*MORE*"	If the user is scanning a list, "*MORE*" replays the current item entirely.

Figure 4.11: VoiceNotes modeless command definitions. These commands are valid at all times.

4.3.3 Comparison of Moded and Modeless Designs

In order to select between the moded and modeless designs for the VoiceNotes interface, the advantages and disadvantages were considered for each design. The following discussion of both designs shows the rationale for selecting the modeless design for the final VoiceNotes interface.

4.3.3.1 Playing a List

In the moded design there are two methods available for playing a list: (1) issuing the "*PLAY*" command while in the top level or note mode and then specifying the name of the list to play; (2) listing the names ("*LIST*") and issuing the *PLAY* command when the desired list name is heard. The second option gives the user the ability to select a list with voice but does not require the name to be recognized. This might be useful if the user is having difficulty getting the system to recognize a particular list name. However, because the *PLAY* command is used in two different ways, this is likely to cause confusion.

In the modeless design, *PLAY* always causes the current list to be played. If the user has difficulty selecting a particular list with voice (due to recognition errors), a selection method is also provided with button input (see Chapter 6). Name words can also be retrained using the VoiceTrainer (see Chapter 5).

4.3.3.2 Recording a Voice Note

In the moded design, users can simply pick up the hand-held and say "*TAKE-A-NOTE*" in order to record a new voice note. Since the system will prompt them for the list name, the users do not have to know the current list. A note is always recorded before a list name is specified allowing the user to record a thought before worrying about where to place it. In order to add a new note to the current list, the user has to use the "*ADD*" command.

In the modeless design, the user has to remember only one command: "*RECORD.*" [8] Neither design solves the categorization problem. Even with the moded design, after recording a note, the user is still required to categorize it immediately (the user must answer the question: "What list?"). This problem was addressed with the addition of the notepad list (see section 4.2.2).

4.3.3.3 Recording New Names

Using the moded interface, new names can be added without any explicit command—if the user speaks a new list name in response to the prompt "What list?" a new list name is automatically added to the VoiceNotes vocabulary. However, determining whether or not a given utterance is a new word (i.e., the input utterance does not match any of the training templates for the current recognition vocabulary) is a difficult speech recognition problem (see section 5.4.4).

In the modeless design, the system does not need to determine when a new list name has been uttered by the user. In order to record a new name, the user must be inside the names list. In this way, recording a new list name is an explicit action.

4.3.3.4 Recognizing Names

Vocabulary subsetting helps to increase recognition accuracy by limiting the number of words that the recognizer has to compare to the input utterance (see section 5.4.1). With the moded design, since the response to "What list?" must be a list name, vocabulary subsetting can be performed following this prompt to increase the recognition accuracy.

Since a list name can be spoken at any time with the modeless interface design, it is not possible to perform vocabulary subsetting. The simplified interface of the modeless design and the ability it gives the user to select a list at any time, outweighs the advantage provided by vocabulary subsetting.

4.3.3.5 Navigation

Since there is no visual display, it is important for users to be able to maintain a mental model of their current position. In the moded design, when the end of the list is reached, the system jumps

[8] An "*ADD*" command is also provided, but has the same definition as "*RECORD*" (see section 5.2).

to the top level causing users to lose their position in the list structure. Unlike Interactive Voice Response systems (IVR) which usually have a main menu, the top level (similar to a main menu) is not conceptualized by the user. The user still thinks s/he is inside the current list and may try to issue commands such as "*NEXT*" and "*PREVIOUS*" that are no longer valid since the list has finished playing. The most significant problem with a moded interface, is that the system may be in one mode, while the user believes that s/he is in another.

Navigating between lists of voice notes is simplified with the modeless interface design. If there is always an active list, it is easier to keep track of one's position. In the modeless design, the user never moves anywhere without explicitly issuing a command. In order to move to another list, the user simply speaks the list's name. In order to move within a list, the user issues commands like "*NEXT*" and "*PREVIOUS*" without fear of "falling off the end of the list." When the last item is reached, if the user says "*NEXT*" the system responds "end of list." The user's position remains on the last item until a command is issued which changes this position. Unlike the moded interface, the user doesn't have to worry about the system suddenly changing the current position in the list. Once the last item on a list is played, the user can wait as long as desired before issuing a command to repeat this item, delete it, or play the previous item. The beginning and end of a list now act as "anchors" for navigational control instead of drop off points.

4.3.3.6 Consistent Command Set

In the moded design, the "*PLAY*" and "*SCAN*" commands operate differently when in name mode than in top level mode. In top level mode, an argument must be specified for the "*PLAY*" or "*SCAN*" command (e.g., "Play what list?"), while in name mode, the argument to the "*PLAY*" command is the current list item that is playing. In addition, not all commands are available at all times. For example, "*NEXT*" and "*PREVIOUS*" commands are not available from the top level mode.

With the modeless interface design, each command always has the same meaning, and the same commands are valid at all times. Since the modeless interface provides a consistent command set, the user will be able to learn and remember the commands more easily.

4.4 Phase 2: Modeless List Management Interface

During Phase 2 of the design process, a modeless interface was adopted. The modeless design provided a conceptual framework for continued iteration to the VoiceNotes interface. Once a general model had been established, the details of the list management interactions (recording, retrieving, navigating, etc.) could be analyzed and iterated upon.

4.4.1 Placement at Startup

When VoiceNotes is started, the user is placed in the notepad list. The addition of the notepad addressed many interface design issues. First, the notepad meets the design criteria that notes be

recorded in a timely manner. By starting in the notepad list, the user can begin recording notes immediately. If the user must interrupt the activity of recording a note, in order to first classify the note, the information may be lost. According to Miyata and Norman, "Because of a person's limited processing and memory capacity, one suspends work on current activity at the risk of losing track of the current activity by failing to resume the work where it was interrupted" ([Miyata 1986] p. 268).

The notepad helps to alleviate another problem: the user's difficulty in categorizing certain notes. Since the notepad constitutes a collection of uncategorized voice notes, notes can be added to this list when the user is unsure of the proper categorization. Malone found "evidence of cognitive difficulties in classifying information" in a study he performed of how people organize their desks ([Malone 1983] p. 107). The existence of the notepad list assists the user by providing a means of what Malone calls "deferred classification."

In order for the notepad list to be useful, however, users must be able to move notes from this list into another named list (see section 4.4.6).

4.4.2 Recording Names

After recording a few notes, the user may then wish to organize them according to topic. This is accomplished by creating a list name. List names are created by moving into the names list and recording a new name (Figure 4.12).

User:	"*NAMES*"	
Hand-held:	"Moving into *names*"	
	"*Notepad*"	
	"Done"	
User:	"*RECORD*"	
Hand-held:	"Recording name"	LED is turned on
User:	"*Shopping*"	LED is turned off
Hand-held:	"New name added"	

Figure 4.12: Speaking the "*NAMES*" command causes the names list to be selected and played back. In the above example, there is only one list name— "*Notepad.*" The user adds to this list by speaking the "*RECORD*" command or pressing the *RECORD* button.

The list name spoken by the user is recorded and trained at the same time (see section 5.2.2). The question arises whether or not the user should know that s/he is training the recognizer. From an interface design standpoint it is desirable not to require an explicit training process. However, since recording a name is exactly like recording a note, confusion may result. One possibility for resolving this problem is to use a different command for recording a name than for recording a note. The interface, however, would no longer be modeless, since recording into the names list would require the use of a different command than when recording into any other list. Instead of

changing the command, the feedback was changed. When recording a name, the system feedback is, "Recording name," and when recording a note the system feedback is "Recording note."

4.4.3 Recording Notes

Recording a note is just like recording a name. When the user records a new voice note, it is added to the end of the current list. The original feedback provided when recording a new note was problematic (Figure 4.13).

User:	"*SHOPPING*"
Hand-held:	"Moving into *shopping*"
User:	"*RECORD*"
Hand-held:	"Recording into...shopping"

Figure 4.13: In this example, the user is adding a voice note to the "*Shopping*" list.

The problem was that if there were any pauses around the word "*Shopping*," a user might begin to speak too early. The pause following the initial part of the feedback ("Recording into <pause> ") might cause the user to take this as an indication to begin speaking. The system does not begin recording until the prompt finishes playing. Therefore, the beginning of a user's note could be cut off under these circumstances. The original reason for the wording of the recording prompt, was to provide feedback of the list name where the new note would be placed. However, through informal observations, it was found that the primary confusion seemed to be in distinguishing between the names and the notes lists. Therefore, the prompts were changed to emphasize the type of item being recorded—"Recording note" versus "Recording name."

For additional feedback, when the system begins recording, the LED (located next to the microphone) on the hand-held device is turned on. When recording completes, the LED is turned off. If there is any doubt as to when to begin speaking, the user can look at the LED.

In addition to the recorded speech prompt (e.g., "Recording note") and the visual indication (LED), a third form of feedback was added—a beep tone. Users of voice mail systems and answering machines are accustomed to "wait for the beep" before they begin speaking. Since some more experienced users may find this extra feedback unnecessary, the ability was provided to turn the beep off.

4.4.4 Navigation

VoiceNotes provides the ability to navigate within and between lists using voice and/or button input. One problem occurred when navigating between lists—users did not know that they had moved to another list. The feedback did not give the user a sense of movement. When the user

selected a list (using voice or button input) the system responded by echoing the list's name (Figure 4.14).

User:	*"PLAY"*
Hand-held:	*"Notepad," "Calls," "Trips," "Demos"* "Done"
User:	*"NOTEPAD"*
Hand-held:	*"Notepad"*

Figure 4.14: When the user issued the command *"NOTEPAD"* (in order to move to the notepad list) the system responded by echoing the list name back to the user.

The feedback did not indicate that the notepad list was now current. There was no sense of having moved from the names list into the notepad list. The feedback was therefore changed to indicate this movement—"Moving into *Notepad.*"

Other problems occurred when navigating within a list. For example, the original definition of the *"PREVIOUS"* command was to play the previous item on the list and then to continue playing forward from that point (Figure 4.15).

User:	*"PLAY"*
Hand-held:	*"Notepad," "Calls," "Trips"*
User interrupts:	*"PREVIOUS"*
Hand-held:	*"Calls," "Trips," "Demos"* "Done"

Figure 4.15: When the *"PREVIOUS"* command was received, the previous item on the list (*"Calls"*) was played, and the system continued to play forward through the list (*"Trips," "Demos"*).

This was confusing since the user did not request "previous and continue playing" but merely asked for the previous item to be played. The commands *PREVIOUS, NEXT* (voice and button), *"FIRST," "MIDDLE," "LAST,"* and *"REPEAT"* were changed to provide only discrete movements. When responding to the *"FIRST"* command, for example, the system plays the first item on the list and waits for further input from the user. This also makes it easier to target an item (see section 5.2.3). Providing commands for discrete movement within each list gives the user fine control over the interaction. "If he can stop the flow, obtain repeats, and move forwards and backwards in the dialogue at will, he can effectively receive information at a rate that suits his own needs and capabilities" ([Waterworth 1984] p. 167).

PREVIOUS and *NEXT* movement commands are provided with both voice and button input. One problem arises when a list contains only one item. In an earlier design, when the user issued the

PREVIOUS command, the system prompted "beginning of list" and when the user issued the *NEXT* command, the system prompted "end of list." This feedback was modified to account for when there is only one item in a list. Under this condition, the system prompts "only one item in list" whenever the *PREVIOUS* or *NEXT* command is received.

In addition, the user's current position in a list must be retained to help maintain navigational control. The original definition of the *PLAY* command was to always play from the top of the list. A problem with this design occurred after playing a list—if the user issued the *STOP* command followed by the *PLAY* command, the system would begin playing from the top of the list and the user would lose his/her current position. The function of the *PLAY* command was therefore changed to start playing from the user's current position in the list. This position is only maintained while the user is in a particular list. Once the user moves to another list, this position is reset to the first item on the list.

4.4.5 Deleting/Undeleting

Users are given the ability to delete and undelete list names and voice notes. "*DELETE*" is an example of a target command—it applies to the current name or note in a list. Deleting a list name is different from deleting a note. Deleting a name removes the name and all associated voice notes. For this reason, the user is warned when deleting a list that is not empty (Figure 4.16).

Hand-held:	"*Notepad,*" "*Calls*"
User:	"*DELETE*"
Hand-held:	"Deleting *Calls* and all associated notes, do you really want to delete?"
User:	"*NO*"
Hand-held:	"Delete canceled"

Figure 4.16: When deleting a list that is not empty, the user is warned and must confirm the action.

When deleting a note, no confirmation is necessary. The system informs the user which note has been deleted by playing back the first 2.0 seconds of the note (Figure 4.17).

Hand-held:	"*Call mom to wish her happy birthday*"
User:	"*DELETE*"
Hand-held:	"Deleted..*Call mom to wish her*"

Figure 4.17: In this example, "*Call mom to wish her*" constitutes the first 2.0 seconds of the note being deleted.

In one design, when deleting a note, the system informed the user what had been deleted by playing the first 2.0 seconds of the note. The entire voice note was not played since this could cause the feedback to be too long. A second approach to this problem is to play the note back at a faster rate than the original recording. In this way, the entire voice note can be played without consuming too much time. In this design, the system plays the voice note being deleted at 1.5 times faster than the user's current speed setting (see section 6.1.6).

Regardless of whether a name or a note has been deleted, the user has the ability to undelete the last item that was deleted, even if the delete action was previously confirmed by the user (Figure 4.18). The list item is retrieved and placed back into its original position in the list. An item (name or note) can be undeleted at any point until the next item is deleted. The system only saves the last item that was deleted. The "*UNDELETE*" command was provided in case of recognition errors, target errors (see section 5.2.3), or in case the user changes his/her mind after deleting a particular item.

Hand-held:	"*Call mom to wish her happy birthday*"
User:	"*DELETE*"
Hand-held:	"Deleted...*Call mom to wish her happy birthday*"
User:	"*UNDELETE*"
Hand-held:	"Undeleted...*Call mom to wish her happy birthday*"

Figure 4.18: The "*UNDELETE*" command, retrieves the last item that was deleted. In the system feedback "Undeleted...*Call mom to wish her happy birthday,*" the voice note "*Call mom to wish her happy birthday*" is played back 1.5 times faster than the user's current speed setting.

4.4.6 Moving

Another important aspect of list management is the ability to move notes from one list to another. This gives the user the ability to re-categorize already existing voice notes. This capability is especially important for allowing notes to be moved from the notepad list (created by the system) into another list created by the user. In this way, first time VoiceNotes users can record notes into the notepad list and move them at a later point when they have added one or more of their own list names. In addition, users can defer categorization of notes by adding them to the notepad list, and moving them later on. The ability to move notes is also important for allowing the user to save segments of recorded background speech (see section 8.8).

The user is given the ability to move a single voice note or a group of notes between lists. In order to move several notes at once, the user must first select a group of notes by "marking" the first and last notes in the group to be moved. This is analogous to selecting a section of text on a visual display in order to cut, copy, or paste the text. In order to designate the first and last notes in the group, the "*BEGIN-MARK*" and "*END-MARK*" voice commands are used (Figure 4.19).

User:	"*NOTEPAD*"
Hand-held:	"*Call micro computer center regarding PowerBook cable*"
User:	"*BEGIN-MARK*"
Hand-held:	"Beginning marked as...*Call micro computer center regarding PowerBook cable*"
User:	"*NEXT*"
Hand-held:	"*Call Chris to set up a meeting for Monday*"
User:	"*END-MARK*"
Hand-held:	"End marked as...*Call Chris to set up a meeting for Monday*"
User:	"*MOVE*"
Hand-held:	"Move to what list?"
User:	"*CALLS*"
Hand-held:	"Notes moved to...*Calls*"

Figure 4.19: In this example, two notes are moved from the *Notepad* list to the end of the *Calls* list. As when deleting, feedback of the voice note being marked is played at 1.5 times faster than the user's current speed setting.

In response to the system prompt "Move to what list?" the user can speak the name of a list or say "*CANCEL*" in order to cancel the move action. The user can change a beginning or an ending mark at any time prior to issuing the "*MOVE*" command. Each time the user issues a "*BEGIN-MARK*" or "*END-MARK*" command, the beginning or ending point is reset to the current voice note. Each mark can be set independently (i.e., the user can change one mark without changing the other). Beginning and ending marks are erased once the user moves into another list.

In order to move a single voice note from one list to another, the user does not have to specify a beginning and ending point. If no marks have been specified, the move command can be used to move the current voice note to another list (Figure 4.20). Another way to move a single voice note is to specify the same beginning and ending point.

User:	"*NOTEPAD*"
Hand-held:	"*Call micro computer center regarding PowerBook cable*"
User:	"*MOVE*"
Hand-held:	"Moving...*Call micro computer center regarding PowerBook cable*"
Hand-held:	"Move to what list?"
User:	"*CALLS*"
Hand-held:	"Note moved to...*Calls*"

Figure 4.20: In this example a single voice note is moved from the *Notepad* list to the *Calls* list.

4.5 Feedback

Since there is no visual display on the hand-held device, audio output is the user's only means of receiving feedback.[9] Whenever a voice command is issued or a button is pressed, audio output indicates the current state of the interface to the user.

4.5.1 Dialogue Design Criteria

- Brevity. It is important to keep the system feedback as brief as possible. There are three reasons for this: (1) The shorter the feedback, the faster the interaction will be; (2) Since there is no visual display, the user will have to retain information in working memory. "Messages should not contain more information than can be readily held in working memory" ([Waterworth 1984] p. 167); (3) The feedback should reflect the kinds of utterances that are acceptable by the system. Since VoiceNotes uses isolated word recognition, the system output should be made terse in an attempt to encourage terse input commands from the user. "With isolated word recognition, the system must be designed to interrogate the user in a way that encourages as high a proportion of 'legal' responding utterances as possible" ([Waterworth 1984] p. 169).

- System's voice versus User's voice. The system's voice must be clearly distinguishable from the user's voice. This is especially important for the VoiceNotes application since much of the spoken output is the user's recorded speech. Many prompts are a combination of pre-recorded system feedback and the user's recorded speech. See sections 4.5.2 and 4.5.3.

- Different levels of feedback. The system should provide feedback that is appropriate for the user's particular experience level with the system. There should be different amounts of feedback for beginning, intermediate, and expert users. See section 4.5.4.

- Configurable. The user should be given the ability to select between different levels of feedback. In addition, where appropriate, the user should be able to "turn on" or off different types of feedback (Figure 4.21).

- Distinct. The system feedback for different commands must be distinct so that it is clear that the system correctly processed the user's command [Brennan 1992]. This can be accomplished without directly "echoing" the user's command. See section 4.5.5.

- System listening. The user needs to know when the system is listening [Brennan 1992]. This relates to the general design criteria for the VoiceNotes interface (section 4.1) that the system is "always listening." In this way there is never any question as to when the system can receive commands. Commands can be issued at any time unless the user explicitly issues the command "*STOP LISTENING*."

---

[9]There is also a single LED on the hand-held device that provides feedback when recording.

- Interruptible. The user should be able to interrupt the system's spoken feedback [Brennan 1992]. This is also a general requirement of the VoiceNotes interface. All output, including both system feedback and the user's voice notes, is interruptible at all times.

4.5.2 Distinguishing the System's Voice from the User's Voice

One important issue of audio output is distinguishing the system's "voice" from that of the user. Much of the speech output consists of the user's recorded voice notes and list names. Some prompts are a combination of the user's voice and the system's voice. For example, if the user selects a list with the name "*Shopping,*" the system feedback is "Moving into *Shopping*." This feedback is composed of two parts: "Moving into" which is a pre-recorded prompt, followed by a recording of the user's voice speaking the list name "*Shopping.*"

The VoiceNotes interface provides three options for voice output: pre-recorded digitized male, female or synthesized prompts. One way to distinguish the system's voice from the user's is by the sex of the speaker. If the user is female, male prompts can be used, and if the user is male, female prompts can be used. This option is recorded in a user configuration file that is read when the VoiceNotes application starts up (Figure 4.21). A second option is the use of synthetic speech.

```
# feedback level (1 - 3)
2
# feedback timeout (in seconds)
60
# prompts (m = male, f = female, s = synthesized)
f
# beep (y or n)
n
# lower limit time-compression ratio
1.5
# upper limit time-compression ratio
2.5
```

Figure 4.21: This is the default VoiceNotes user configuration (preferences) file. The user can select the type of output (male, female or synthesized feedback), the timeout between system feedback of the current list name, the level of feedback (1 = expert; 2 = intermediate, 3 = beginner), and whether or not a beep is played in recording prompts. The last two options are discussed in section 6.1.7.

4.5.3 Recorded vs. Synthetic Speech

There are many arguments in favor of using synthetic speech.[10] Synthetic speech provides easier maintenance of an application. When using recorded speech, each prompt must be digitized and edited. Whenever new prompts are needed, the original speaker must be found. Due to the unnatural quality of synthetic speech, it is also easily distinguishable from the user's voice.

However, since there are a limited number of prompts needed for the VoiceNotes application, the use of synthetic speech is not necessary and the liabilities outweigh the benefits. The perception of synthetic speech imposes greater demand on short-term memory than does recorded speech [Luce 1983]. According to Schmandt, "if it is more difficult to listen to synthetic speech, this may interfere with whatever else the listener may be doing, such as keeping track of position in a menu hierarchy, or remembering the information being spoken" ([Schmandt 1992], p. 100). The load on short-term memory may already be large due to the lack of a visual interface—users must maintain their position in the list structure while listening to recorded voice notes and prompts. Any further demand on short term memory is therefore undesirable.

> "As speech cannot be scanned, information must be held in working memory...A visual menu can be scanned repeatedly so that the memory load is reduced, and visual cues such as colour can provide pointers to the menu structure...The use of synthetic speech exacerbates the problems inherent in presenting information by voice" ([Waterworth 1984] p. 166).

In addition, since users may tend to have difficulty navigating without a visual interface, comprehension of the spoken feedback is essential for maintaining one's position in the lists. Any loss of intelligibility could be costly to user performance.

Another factor affecting the intelligibility of synthetic speech is context. As the intelligibility of speech decreases, users rely more on the context of the speech [Pisoni 1985]. "The processes used to perceive and understand synthetic speech are heavily dependent on the contextual environment" ([Luce 1983] p. 19). When listening to isolated words users must "rely more on the acoustic-phonetic signal itself" due to the lack of "top-down contextual support" ([Luce 1983] p. 19). The terse output given by the VoiceNotes application could therefore be more difficult to comprehend if presented using synthetic speech.

The intelligibility of synthetic speech degrades more as a result of noise than does recorded speech [Pisoni 1982]. Since the hand-held must be usable from a variety of locations, some which may be noisy, noise is an important factor that must be taken into consideration.

4.5.4 Feedback Levels

The VoiceNotes application provides three levels of speech feedback to the user. Level 3 is for novice users and provides the greatest amount of feedback, level 2 is for intermediate users, and

[10]Synthetic speech as used in this section refers to text-to-speech.

level 1 is for expert users and is the least verbose. The purpose of providing multiple levels of feedback was to experiment with different amounts of audio output and to support a range of VoiceNotes users from the least to the most experienced (Figure 4.22).

User Command	Level 3 Feedback	Level 2 Feedback	Level 1 Feedback
PLAY	"Playing *Shopping*" "*Buy some lettuce*"...	if **timeout** has elapsed: "*Shopping*" "*Buy some lettuce*"... or else: "*Buy some lettuce*"...	"*Buy some lettuce*"...
NEXT	"Next" "*Salad dressing*"	"*Salad dressing*"	"*Salad dressing*"
RECORD	"Recording note" user speaks note "Note added to *Shopping*"	if **timeout** has elapsed: "Recording note" user speaks note "Note added to *Shopping*" or else: "Recording note" user speaks note "New note added"	"Recording" user speaks note "Done"
NAMES	"Moving into *Names*" "*Shopping*," "*Demos*"...	"Moving into *Names*" "*Shopping*," "*Demos*"...	"*Names*" "*Shopping*" "*Demos*"...

Figure 4.22: This figure shows examples of speech feedback for each level. **Timeout** refers to the amount of time that has elapsed since the last time the user was given feedback of the current list name. This timeout is set to 60 seconds by default but can be set to any amount in the user's configuration file.

4.5.4.1 Level 3

Level 3 "echoes" every command (voice or button), to assure novice users that their command has been accurately received. For example, if the user issues the *NEXT* command, the system echoes "Next" before playing the next item on the list. Each time the user plays or records into a list of notes, the list name is echoed. For example, when the user issues the *PLAY* command, the system responds "Playing *Shopping*."

4.5.4.2 Level 2

At level 2, commands such as *PLAY*, *NEXT*, and, "*FIRST*" are not echoed since the output of the given item will inform more experienced VoiceNotes users that their command was successfully received. For example, if the user says "*FIRST*," the system responds by playing the first item on the list. One problem is that if the user does not remember which item is first on the list, s/he may not realize if the recognizer made a substitution error and played the *NEXT* item instead.

In contrast to level 3, level 2 uses a timer to determine when to prompt the user with the current list name. Constant feedback of the list name can be quite tedious for a more experienced user,

especially when performing a sequence of operations within the same list. For example, if the user wants to record three consecutive voice notes into a list, it becomes a time consuming task if the user is prompted "Note added to *<list name>*" each time. Even though this feedback can be interrupted, the intermediate level user might tend to wait for the spoken output to finish before recording the next note. Therefore, the system gives feedback of the list name only if the user has not been prompted with the list name for a certain period of time.

Feedback of the list name is important when the user has been idle for a period of time. In these cases, the user probably does not recall his/her position in the list structure. For example, if a time period of 60 seconds or more elapses in which the user has not issued a command, the next time s/he issues the *PLAY* command the system will first play the name of the current list before playing the list items. In this way, if the user is interrupted while using VoiceNotes, knowledge of the current position can quickly be regained.

4.5.4.3 Level 1

In contrast, at level 1, the list name is never re-played once a list has been selected. In addition, when a list is selected, instead of the prompt "Moving into <list name>," only the list name is echoed to the user. The "Moving into" portion of the prompt is left off. In general, prompts at level 1 are kept as terse as possible. User commands are never echoed at this level.

For some commands, feedback is essential, even for an expert user. For example, feedback for the "*DELETE*" command informs the user if the system has deleted the correct item. While most systems provide evidence of misunderstandings using error messages, the importance of "positive evidence" [Clark 1991] is not always emphasized. "After all, part of the work of repairing a misunderstanding is in identifying that one has occurred" ([Brennan 1992] p. 2). If the feedback indicates that the system has deleted the wrong item, the user can then repair this error by issuing the "*UNDELETE*" command.

4.5.5 Distinct Feedback

It is important that feedback for each command be distinct. This can be accomplished without directly echoing the user's command. In one design iteration, the feedback for several commands was the same. In the example shown in Figure 4.23, the list name "Shopping" is both the current and the last item in the name list. This situation regularly occurs just after recording a new list name. After recording the list name "Shopping," if the user issues the *PLAY* command, the system plays the new recording, "*Shopping.*" Now if the user selects the list by speaking its name, the feedback is the same. Next, if the user asks for confirmation about his/her current position using the "*WHERE-AM-I*" command, the feedback is once again the same.

In total, the same feedback was given for three different commands. By providing more specific feedback for each of the commands, users are informed that their request has been correctly

processed even though the system has not "echoed" the command itself. In a later design, these prompts were changed to be more distinct (Design Iteration 2).

User Command	Design Iteration 1	Design Iteration 2
PLAY	"*Shopping*"	"*Shopping*"
"*SHOPPING*"	"*Shopping*"	"Moving into *Shopping*"
"*WHERE-AM-I*"	"*Shopping*"	"You are in *Shopping*"

Figure 4.23: In design iteration 1, it was possible, under certain circumstances, for the feedback to be the same for three different commands. Design iteration 2, resolved this problem by providing more distinct feedback for each of the commands.

5. Voice Input

This chapter discusses issues relating specifically to interaction with the VoiceNotes application through voice input. The first iteration of VoiceNotes was a voice-only design—all commands were issued using voice input. The goal at this point in the design was to experiment with a voice-only interface (all interactions are performed using voice input and all feedback is provided using voice output).

Voice Command	Definition
"*PLAY*" / "*CONTINUE*"	Plays each item in a list starting from the current item.
"*RECORD*" / "*ADD*"	Records an item onto the end of the current list.
"*STOP*" / "*PAUSE*"	Stops playing the current item or prompt and retains position in the current list.
"*NEXT*", "*PREVIOUS*"	Plays the next/previous item in the current list and stops to wait for user input.
"*REPEAT*"	Replays current item on the list and stops to wait for user input.
"*NAMES*", "*CAPTURE*", "*NOTEPAD*"	Moves into the *Names/Capture/Notepad* list and plays all of the items starting from the beginning of the list.
"*<NAME>*"	Moves into the list *<Name>* and plays all of the notes starting from the beginning of the list.
"*DELETE*"	Deletes the current item in the list; if the list item is a name, all of the notes associated with the name are also deleted.
"*UNDELETE*"	Retrieves the last item that was deleted and returns the item to its original place in the list structure.
"*SCAN*"	Plays the first 1.0 seconds of each item in a list starting from the current item.
"*FIRST*", "*MIDDLE*", "*LAST*"	Plays the first/middle/last item in the current list and stops.
"*STOP-LISTENING*"	Turns recognition off.
"*PAY-ATTENTION*"	Turns recognition on.
"*YES*"	Confirms deletion of a name that has notes associated with it.
"*NO*"	Cancels deletion of a name that has notes associated with it.
"*WHERE-AM-I*"	Plays the name of the current list (e.g., "You are in *Demos*").
"*BEGIN-MARK*"	Marks the current voice note as the beginning of a selection.
"*END-MARK*"	Marks the current voice note as the end of a selection.
"*MOVE*"	Moves the current or the marked voice notes to another list.
"*CANCEL*"	Cancels a move command.

Figure 5.1: VoiceNotes voice-input vocabulary.[11] An **item** can be either a **name** or a **note** depending on the type of list.

[11]Only a portion of these commands are supported by button input (see section 6.2).

5.1 Speech Recognition Technology

One important consideration in defining the user interaction for the VoiceNotes application is the type of speech technology that will be employed: isolated versus continuous word recognition and speaker dependent versus speaker independent recognition. The following sections discuss the reasons why a speaker dependent isolated word recognizer is appropriate for use with the VoiceNotes application given the current state of speech recognition technology.

5.1.1 Isolated vs. Continuous Word Recognition

A continuous speech recognition system provides a more natural conversational interface than an isolated word system, since it is unnatural to pause after each word spoken. However, if the commands are composed of single words or short phrases, the need for pauses can be minimized. Even with a continuous speech recognizer, the user must still speak within a pre-defined grammar. The user of a continuous speech recognition system may be mislead to believe that any conversational speech input is acceptable. Waterworth found it "difficult to find a simple way of encouraging the user to make acceptable responses" when using a connected speech recognizer ([Waterworth 1984] p.171). Using isolated word recognition, the distinction between conversational speech and acceptable input is more clearly defined. "To some extent, machines of intermediate sophistication can prove more problematic to the user than very simple systems. With the latter it is very clear what the limitations of the machines are whereas with the former the apparent 'intelligence' of the system may lead the user to have unrealistic expectations about its capabilities" ([Waterworth 1984] p. 175).

5.1.2 Speaker Dependent vs. Speaker Independent Recognition

Speaker dependent recognition systems have the disadvantage of requiring each individual user to train all of the words in the vocabulary. However, most speaker independent systems require a pre-defined grammar to be trained by collecting data from many speakers. Collected training data for a new grammar can often be a lengthy process and has the disadvantage of discouraging an iterative design process.

5.1.3 Selecting a Recognizer for VoiceNotes

The most important use of speech input in the VoiceNotes application is for the categorization and simple list selection method that it provides. The user's ability to personalize the application by selecting his/her own list names is essential. In addition, users must be able to create these categories in real-time. Speaker dependent isolated word recognizers such as the Voice Navigator provide the ability to add new words to the recognition vocabulary in real-time. Most speaker independent continuous recognition systems do not have this capability. Some speaker independent continuous word recognizers provide the ability to add a new word by providing a

phonetic spelling of the word for the system.[12] Since the hand-held has no keyboard, typing in new words would not be possible. Spelling or typing in new words would also be inconvenient given many of the situations in which the hand-held is intended to be used (e.g., hands and eyes busy).

The ability to add new words "on-the-fly" is therefore the main reason for using a speaker dependent recognizer for the VoiceNotes application. In addition, since the hand-held is intended to be a personal device, speaker independent recognition, while having many advantages, is not necessary. Isolated word recognition is also adequate for the simple input required by the VoiceNotes application, and more closely matches the button interface provided by the hand-held device (see Chapter 6).

5.2 VoiceNotes Vocabulary

The VoiceNotes vocabulary is made up of pre-defined command words (Figure 5.1) and user-defined names.

5.2.1 Command Words

"Language design refers to the creation of a spoken language suitable for human interaction with an application" [Rudnicky 1989]. The following are some of the considerations that were taken into account when selecting a command vocabulary for the VoiceNotes application:

- Intuitive. The words selected must be intuitive and easy to remember. Words like "play" and "record" are familiar to most people who use tape recorders. Originally, the command "*TAKE-A-NOTE*" was considered for recording a voice note. This proved to be too specific, however. The "*RECORD*" command could be used for recording names or recording notes since the word "*RECORD*" does not imply the type of item that is being recorded. Providing a single voice command for each type of action makes it easier for the user to recall the appropriate command word for a given task.

- Single word commands. Every action should be initiated with only a single command word. Originally, command phrases like "Play *<list name>*" were considered. However, users would be forced to pause between words since an isolated word recognizer is being used. Not only is it unnatural to pause between words, but it is especially difficult to determine just how long to pause before speaking the next word in order to avoid rejection errors. The modeless interface design solves this problem. The argument to commands such as *PLAY* and *RECORD* is the current list.

[12] It may be possible to allow the actual text to be entered by using the front-end of a text-to-speech synthesizer to convert it to its phonetic representation [Asadi, 1991].

- Acoustic similarity. It is important to make each command as "acoustically distinct" as possible. The selection of words that minimize acoustic similarity is often in conflict with the selection of words that are intuitive to the user. For example, the words "repeat" (/ripit/) and "delete"(/dilit/) are both 2 syllable words containing the phoneme /i/ in each syllable. The "repeat" command could be replaced with a semantic equivalent such as "again" but this seems a less intuitive choice. There is a tradeoff that must be made since it is not always possible to select words that are both intuitive and acoustically distinct.

- Vocabulary size. Vocabulary size can have an impact on both acoustic similarity and the user's ability to recall the correct command for a given task. The more words in the vocabulary, the more probable it is that some will be acoustically or semantically confusable. An attempt was made to keep the vocabulary as small as possible.

- Command synonyms. If the goal is to keep the vocabulary size as small as possible, it is questionable whether or not more than one command word should be provided for the same function. For example, there are two words in the vocabulary for recording—"*RECORD*" and "*ADD*." The "*ADD*" command was provided as a correlate to the "*DELETE*" command. Since there is a "*DELETE*" command, it would be natural for the user to assume that there is also an "*ADD*" command. Part of what makes a good interface is its predictability. Even though the goal is to limit the size of the vocabulary, providing the "*ADD*" command makes the vocabulary more "predictable" and possibly easier to remember. Other synonyms are "pause" (synonym for "*STOP*") and continue (synonym for "*PLAY*"). The user's goal at one time might be to "pause" the spoken output in order to perform another task. Even though the "*STOP*" command will also retain the user's position in the list, the "pause" command may correspond more closely to the user's goal under certain circumstances.

- Ability to provide equivalent button input. Another important consideration is the ability to provide equivalent button commands to correspond to the available voice commands. It is not necessary that every voice command have a button equivalent, but that the core functionality provided by the application be operable using button input (see section 6.1).

Vocabulary features were not considered when selecting the VoiceNotes vocabulary. Waterworth defines vocabulary features to be factors such as voicing, number of syllables, and manner of articulation [Waterworth 1984]. Waterworth found that "vocabulary features did significantly affect recognition, and these have the advantage that they can be selected to improve performance, without incurring additional expense" ([Waterworth 1984] p. 170). For example, one recommendation is to avoid command words beginning with an unstressed vowel or fricative when using a particular isolated word recognizer [Green 1983]. This would rule out the "*FIRST*" command, for example. It is only possible to employ such constraints "provided that subjectively acceptable and semantically equivalent alternatives can be found" ([Waterworth 1984], p. 170). This is just the problem—the best word from a semantic standpoint is not likely to meet a long list of such requirements. It is also unclear, whether or not there is a significant impact on

recognition accuracy. The factors more likely to have an impact on recognition accuracy are: noise, user experience and attitude, and proper training of the vocabulary (the user must speak clearly, and assertively, in a manner that is repeatable). Therefore, in selecting the VoiceNotes vocabulary, more attention was placed on determining the most intuitive words semantically than in matching a long list of acoustic parameters.

5.2.2 User-defined Name Words

A criteria for the design of VoiceNotes was to allow the users to define their own list name words. Names are personal to each user and provide the organizational structure for one's voice notes. The VoiceNotes application allows the user to build upon the baseline command vocabulary as the system is used.

The Voice Navigator stores vocabulary words in what is known as a "language file." VoiceNotes uses a simple flat grammar for its language (all words are active at all times[13]). Each word in the language has an associated word number. VoiceNotes allots the first 50 words in the vocabulary for commands (this is to allow for easy expansion; the current number of command words is 27), and the next 50 for user-defined name words. Therefore, the user can create up to 50 lists of voice notes.[14]

List names are defined by each individual user. When the user records a name, this single utterance of the name is used to create a voice template for the Voice Navigator. Voice templates are stored in the user's "voice file" with the corresponding word number from the language file. The process of training the recognizer has not been made explicit since only a single template is collected. Fairly accurate recognition is achieved using only a single utterance of training data.[15] It is possible that the use of more training utterances would provide an increase in recognition accuracy, however, requiring the user to speak each list name repeatedly is not desirable from a user interface standpoint. It is also possible to adapt[16] the voice template each subsequent time the list name is uttered by the user. However, this would require confirmation that the correct list name has been recognized.

5.2.3 Targeting Style Commands

Many of the VoiceNotes commands use a targeting style—they act on the current item (name or note) in a list. "*REPEAT*," "*NEXT*," "*PREVIOUS*," and "*DELETE*" are all examples of target commands. Muller and Daniel refer to this as "voice direct manipulation" [Muller 1990].

---

[13]There are some exceptions due to vocabulary subsetting (see section 5.4.1).

[14]This is an arbitrary number which can be expanded by updating the VoiceNotes language file. However, 50 lists seems to be a good practical limit for this application.

[15]An exact measurement is difficult to make except under controlled conditions (controlling for noise, speaker's voice quality, etc.).

[16]The Voice Navigator allows a voice model to be built from one or more templates. A model can be adapted by collecting more templates and averaging with the previously collected ones.

User:	*"PLAY"*
Hand-held:	*"Demo for Apple visitors on Wednesday"*
	"Demo for Speech Group after meeting"
User interrupts:	*"REPEAT"*
Hand-held:	*"Demo for Speech Group after meeting"*

Figure 5.2: In this example, *"REPEAT"* acts as a target command. In this case the target is the voice note *"Demo for Speech Group after meeting."*

If a list is continuously playing from one item to the next when a command is issued, there is the problem of determining the exact "target." In the above example, depending on when the system receives the *"REPEAT"* command, one of a possible three notes might be repeated—the one actually targeted by the user, the one previous to the desired target, or the one following. This is due to two factors: (1) user reaction time, and (2) system response time. The user's command generally follows the playback of the target item [Muller 1990]. Therefore, if the system requires the target command to be received during the playback of the targeted item, the wrong item could be selected (Figure 5.3). In addition, there may be some delay in receiving the voice input event from the Recognition Server (see section 9.2).

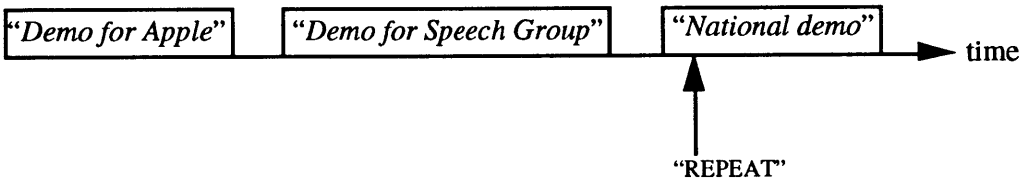


Figure 5.3: In the above example, the *"REPEAT"* command is received following the item that the user wants to repeat. If the system were to apply the target command to the current item that is playing, the voice note *"National demo"* would be erroneously repeated instead of the note, *"Demo for Speech Group."*

Muller and Daniel suggest using a "partially overlapping temporal window" to overcome this problem. In the VoiceNotes system, an item's **target window** extends beyond the item to the beginning of the next item. In addition, there is a 0.5 second period of silence following each item in a list (Figure 5.4).

The problem of selecting the correct target was also alleviated by giving the user discrete movement controls through the list. While the *"PLAY"* command causes the current list to play continuously from one item to the next, once a movement command is issued (e.g., *"NEXT,"* *"FIRST,"* or *"REPEAT"*) only the requested item is played and then the system stops to wait for the next user command. In this way, the user can locate the desired item and then issue the target command, ensuring 100% targeting accuracy.

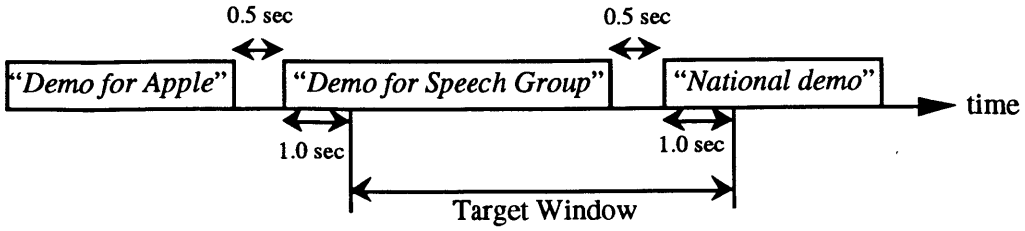


Figure 5.4: The VoiceNotes application uses a target window that extends to the next item in the list. This means that in order to "target" (e.g., repeat, delete, etc.) any note on a list, the user can speak the command in the time interval starting from the 1.0 second of the desired item to the 1.0 second of the next item in the list. There is a 0.5 second pause between each item. In the above illustration, if the "*REPEAT*" command is spoken within the time frame of the "target window" the voice note "*Demo for Speech Group*" will be repeated.

5.3 Voice-only Training

Prior to using VoiceNotes for the first time, users must train each of the command words in the vocabulary. Training is accomplished using a separate program called the VoiceTrainer (see also section 9.5). The training program is voice-only in that the user is prompted to speak each word using digitized speech output.

VoiceTrainer:	"Repeat after me"
VoiceTrainer:	"Play"
User:	"*Play*"
VoiceTrainer:	"Record"
User:	"*Record*"
etc.	

Figure 5.5: This shows a sample interaction for training the VoiceNotes vocabulary.

The first time that the user trains the vocabulary, there are only command words. This initial vocabulary is expanded each time the user records a new list name. Upon subsequent training sessions, the vocabulary to be trained (or re-trained) is now composed of both system-defined command words and user-defined name words. The initial motivation for using voice instead of text prompts was that the system does not know the "text" associated with each name word. All the system has is a digitized sound file of the user's voice speaking the name word. For each of the command words, the user is prompted using a pre-recording digitized speech prompt, while for each of the user's name words, the user is prompted with his/her own voice recording of the name.

Waterworth found that users, when prompted vocally in a voice training session, tended to imitate the way the words were pronounced [Waterworth 1984]. Thus, voice prompts could be used as a way of encouraging the kind of speech that is necessary when talking to the speech recognizer

(i.e., clear, declarative, and isolated). However, as Waterworth points out, this is only successful if the same pronunciation used during training is employed during actual use of the system.

Green and Waterworth discuss the idea of "hidden training" [Green 1983] [Waterworth 1984]. The goal is to elicit speech from the user that more closely matches the pronunciation that will be used during the actual task than when monotonously repeating words in a list. Green gives the example of training the words "up," "down," "left," and "right." The words are trained by playing a game in which the user has to move a marker through a maze. The claim is that this is similar to the actual task. Waterworth challenges this claim— "while this may avoid some of the irritation generated by the training session, the extent to which the hidden training truly resembles the task situation, from the user's point of view, is more doubtful" ([Waterworth 1984] p. 170).

VoiceNotes employs a kind of "hidden training" in that name words are trained when the user records a new list name. A question that needs further evaluation regards whether "hidden training" improves or degrades recognition accuracy (i.e., should users be aware that they are training the recognizer). On the positive side, "hidden training" is less tedious for the user and may elicit more natural speech. On the negative side, users may not speak in a manner that is best for voice recognition since they are unaware of the training process. In addition, VoiceNotes users create their own list names. If the training is "hidden," the user may not be aware of the considerations that should be taken into account when choosing a list name (e.g., acoustic similarity).

5.4 Speech Recognition Issues

5.4.1 Vocabulary Subsetting

Vocabulary subsetting helps to improve recognition accuracy by decreasing the corpus of words that the recognizer must match against. Although VoiceNotes uses a flat grammar, vocabulary subsetting is performed under certain conditions. Since the VoiceNotes interface design is modeless—all commands are (almost) always valid, there are only a few instances when subsetting can be used. One such instance is for user confirmation. When deleting a list name with associated notes, the user is warned and asked for confirmation. Following the prompt, "Do you really want to delete?" all words in the vocabulary are deactivated with the exception of "*YES*" and "*NO*." Once a response is received, all vocabulary words are activated once again. At all other times, the vocabulary is subset to contain all words except for "*YES*" and "*NO*" since these are only valid when requesting user confirmation.

Vocabulary subsetting is also performed when voice notes are moved from one list to another. When the "*MOVE*" command is received, the system susbets the vocabulary to contain only list names and the "*CANCEL*" command before prompting the user, "Move to what list?" This allows the system to obtain more accurate recognition of the list name, while allowing the user to cancel the move action.

Under certain conditions the vocabulary may be subset to contain only a single word. When the "*STOP LISTENING*" command is received, the vocabulary is subset to contain one command—"*PAY ATTENTION*." Upon receiving the command "*PAY ATTENTION*," all vocabulary words (except for "*YES*" and "*NO*") are once again activated. When the vocabulary is subset to contain a single word, the recognizer has only one template to match against input utterances. Under these conditions, the Voice Navigator often falsely reports the occurrence of the command "*PAY ATTENTION*" from speech that the user intended the recognizer to ignore.[17] For this reason, when subsetting the vocabulary to a single word, the rejection threshold is increased. This raises the criteria for reporting a correct recognition and acts to decrease the number of insertion errors. One resulting problem with raising the rejection threshold is that the user may have difficulty getting the recognizer to start listening again. Therefore, the rejection threshold cannot be raised to an extreme level.

VoiceNotes also uses subsetting in order to temporarily disable a single word in the recognition vocabulary (see section 5.5.6).

5.4.2 Recording vs. Recognizing

One difficulty is in determining the difference between spoken commands and voice notes. For example, when recording a voice note, the recognizer must be turned off. If the recognizer "listens" for input while the user is speaking a voice note, insertion errors[18] occur. Once the user says "*RECORD*," recognition is turned off until a pause is detected. The resulting problem is that if the "*RECORD*" command is erroneously recognized, or the user decides to cancel the recording, this cannot be done using voice commands. If the user were to remain silent after the "*RECORD*" command, a pause could be detected, and the recording discarded (see sections 5.5.2 and 5.5.3). However, most users will attempt to terminate the recording by shouting "stop." This does not terminate the recording, but records the speech instead. The result is a voice note containing "stop" and whatever other speech the user employs to try to terminate the recording.

An interesting research area for isolated speech recognition is therefore the ability to determine the difference between isolated "command" speech and connected natural speech. However, since the Voice Navigator does not have such a capability, it is not possible to interrupt recording of a voice note with voice input. An alternative has therefore been provided using button input (see Chapter 6).

---

[17]This is known as an insertion error or false acceptance. An insertion error occurs when the recognizer reports hearing a command (usually in response to background noise, a cough, lip smack, or background speech) when none was issued [O'Shaughnessy 1990].

[18]Note that insertion errors occur regardless of whether the voice note happens to contain any words that are in the recognition vocabulary.

5.4.3 Output vs. Input

Another difficulty is in distinguishing the spoken output from the spoken input. If the spoken output is "heard" by the recognizer, than insertion errors may result. This is a similar problem to the one described in the previous section. However, it has the added difficulty that the user's exact training utterance may be output by the system. For example, when the user issues the list name command "*DEMOS*," the system responds "Moving into *DEMOS*." The "*DEMOS*" portion of the system feedback is the recording of the user's voice that was used for training the recognizer. If the recognizer "hears" this spoken output, a "feedback loop" could occur (e.g., "Moving into *DEMOS*," "Moving into *DEMOS*," ...).

This problem can be solved through careful placement of the speaker and microphone on the hand-held unit that minimizes feedback of the spoken output into the microphone.

5.4.4 New Word Detection

The moded design of VoiceNotes proposed to determine when a new list name was spoken without any explicit command from the user. After recording a voice note the user would be prompted, "What list?" The response to this prompt would then be analyzed to determine whether it matched any of the current names in the vocabulary. If there were no "good fitting" match, it would be classified as a new name and a new template would be added to the user's voice file. The proposed approach was to examine the recognizer's top five guesses and scores to determine if a close enough match existed. There are many problems associated with such an approach. The most difficult problem is screening out noise and background speech. This input would be rejected by the recognizer and might then be inappropriately classified as a new word because it did not match closely to any of the voice templates.

The VoiceNotes modeless design circumvented the new word problem by making the recording of a new name an explicit process. Whenever the user is inside the names list and issues the "*RECORD*" command, a new name is added to the vocabulary. However, "new word" detection remains a compelling problem for future research.

5.5 List Management with Voice

5.5.1 Recording Names and Notes

The use of pause detection is necessary when recording a voice note or list name after issuing the voice "*RECORD*" command. The following sections describe why accurate pause detection is critical to the VoiceNotes interface and provides a technical description of the technique used for discriminating speech from silence.

5.5.2 Pause Detection: Importance to the Interface

Reliable and accurate pause detection is essential for the VoiceNotes interface. Due to the slow and serial nature of speech, it is desirable to play through a list of voice notes as quickly as possible. Periods of silence between voice notes need to be consistent and controllable by the application. If each voice note begins and ends with a period of silence, playing through a list of notes becomes sluggish and the control of timing between voice notes inaccurate. The goal of pause detection is to: (1) terminate recording if no speech is detected within the first n seconds of the recording, where n is the length of the initial pause; (2) terminate recording if a final pause is detected (after a period of speech, the user stops talking, indicating the end of a voice segment); (3) remove any initial or final silence from a recorded segment.

The removal of initial and final silence from a recorded segment is even more critical when recording a list name than when recording a voice note. At feedback levels 2 and 3, the list name is echoed to the user each time a list is selected (e.g., "Moving into *<list name>*"). This prompt is output by playing one segment of speech after the other (i.e., segment 1 = "Moving into," segment 2 = "*<list name>*"). If there is silence before the list name, the pause between the first and second segments causes the output to be perceived as two prompts instead of one.

Initial silences also cause problems with the "*SCAN*" command. If list items contain a large amount of initial silence before speech, the result of the "*SCAN*" command is to play silences rather than the content of the notes.

5.5.3 Pause Detection: Technical Description

VoiceNotes uses an adaptive pause detection algorithm.[19] When the command "*RECORD*" is spoken, speech is recorded until a pause is detected. The algorithm allows two parameters to be set by the application, the initial pause length and the final pause length. The initial pause length is the number of milliseconds of silence that must elapse before a pause is detected during the initial part of the recording. A pause initial value of four seconds is used, allowing the user to pause for up to four seconds before beginning to speak (after being prompted). If no speech is detected within this time period, the recording is deleted and the user is prompted "Nothing recorded." The final pause length is the number of milliseconds of silence that must be detected before terminating the recording. A final pause length of two seconds is used. The pause initial length is set to a larger value than the final pause since the user usually requires more time to begin speaking.

The average magnitude of the speech signal is calculated every 100 ms and passed to the pause detector. The algorithm dynamically keeps track of the minimum energy[20] value during a

[19]The pause detection algorithm used by VoiceNotes was written by Barry Arons of the Speech Research Group.
[20]For the sake of this discussion, the average magnitude will be referred to as the "energy" of the signal.

recording—the noise threshold. The noise threshold is initially set to a high value (greater than the highest expected noise value). If the energy value (in dB) for the current interval is within a given signal-to-noise ratio of the noise threshold, then that interval is determined to contain silence. For example, if the noise threshold is 4 dB and the SN ratio is 6 dB, an energy value of 10 dB or less would be considered silence. In a normal recording (after being prompted the user pauses shortly before beginning to speak) the noise threshold is expected to drop significantly within the first interval. As described in the previous section, in some cases, the user may begin speaking before recording has started causing the initial energy value to be high and the noise threshold to remain at a high value. This case is handled by looking for a significant drop in a later interval. If such a drop occurs, all of the previous intervals are determined to have contained speech.

Initial and final silence times are calculated by this algorithm. Once recording has completed, initial and final silences are removed from the sound file.[21]

This algorithm's adaptive noise threshold is particularly useful for the VoiceNotes application in that the hand-held device may be used under a variety of noise conditions. This pause detection algorithm allows the interface to operate successfully under different noise conditions by dynamically calculating the noise threshold.

One problem with the algorithm occurs if the noise conditions change during a recording. The algorithm adapts on a per recording basis; once the noise threshold is calculated, it is not recalibrated during the recording. If the noise level suddenly increases, the algorithm does not account for this. Since recording of voice notes and list names occur over very short time durations, this is not generally a problem. However, the impact is greater when capturing longer durations of background speech (see Chapter 8). The algorithm was therefore modified to handle possible changes in the noise level.

5.5.4 Recording Names

When recording names, pause detection information is obtained from both the pause detection algorithm and the Recognition Server. The information provided by the Recognition Server is used to terminate the recording, while the information returned by the pause detection algorithm is used to remove initial and final silence from the recording. Isolated word recognizers use end-point detection—a process that separates speech segments of an utterance from the background [Savoji 1989]. End-point detection differs from pause detection in several respects due to the high level of accuracy required for accurate isolated word recognition. While pause detection algorithms, such as the one described, can afford to leave several hundred milliseconds at the beginning and end of an utterance, end-point detectors cannot. When recognizing words with

[21]Two hunderd milliseconds are subtracted from these silence times to allow for a margin of error and to ensure that the speech doesn't get truncated at the beginning or end.

similar sounding beginnings or endings (e.g., the letters 'b, 'c,' and 'd') "the error rate may greatly increase because of end-point detection errors" ([Savoji 1989] p. 46). End-point detectors must also exclude artifacts such as lip-smacking or breath noise.

Since end-point detection is necessarily more accurate than pause detection, the information from the Recognition Server is used to terminate the recording of the name. The server sends an event to the VoiceNotes application when: the training utterance has been collected successfully, an error occurs during training, or a timeout specified by the application is reached. Training errors occur when the user attempts to record an utterance of more than 2.5 seconds.[22] Under this condition the user is prompted, "Names must be one or two words only, try again." If no training utterance has been received within a four second time period, the server returns a timeout and the user is prompted, "Nothing recorded." If a training template has been successfully collected, the application is notified and recording is terminated. The initial and final silence times returned by the pause detection algorithm are then subtracted from the sound file containing the new list name.

5.5.5 Navigation

Navigating between lists of voice notes is provided by the list name commands that are created by the user. In order to move into a particular list, the user simply speaks the list's name. Voice provides the greatest advantage in terms of list selection. Selecting a list by speaking its name is direct, fast, and intuitive.

Two commands are provided for continuously playing through a list: "*PLAY*" and "*SCAN.*" While "*PLAY*" causes each voice note to be played from beginning to end, "*SCAN*" outputs only the first 1.0 seconds of each voice note. The "*SCAN*" command attempts to provide the capability for browsing through a list. Once the user hears a voice note of particular interest or wants to stop scanning, issuing any command will terminate scanning. For example, the user can interrupt with the "*REPEAT*" command, causing the current item to be replayed from beginning to end.

Discrete movement within a list of notes or names is provided by commands such as "*NEXT,*" "*REPEAT,*" and "*FIRST.*" While "*NEXT*" and "*PREVIOUS*" are necessary for providing a complete voice interface, the use of such movement commands with voice can become tiresome as the lists become longer. For example, in order to move ahead three items, the user would have to utter "*NEXT*<pause> *NEXT*<pause> *NEXT.*" Voice seems more appropriate for "coarse" rather than "fine" movements through the list. Voice provides the ability to easily jump to different points within a list (see section 7.1).

One question arose regarding the words selected for two of these commands. Originally the words "*TOP*" and "*BOTTOM*" were used instead of "*FIRST*" and "*LAST,*" but they impose a spatial

[22]The Voice Navigator will only accept isolated utterances of up to 2.5 seconds in length.

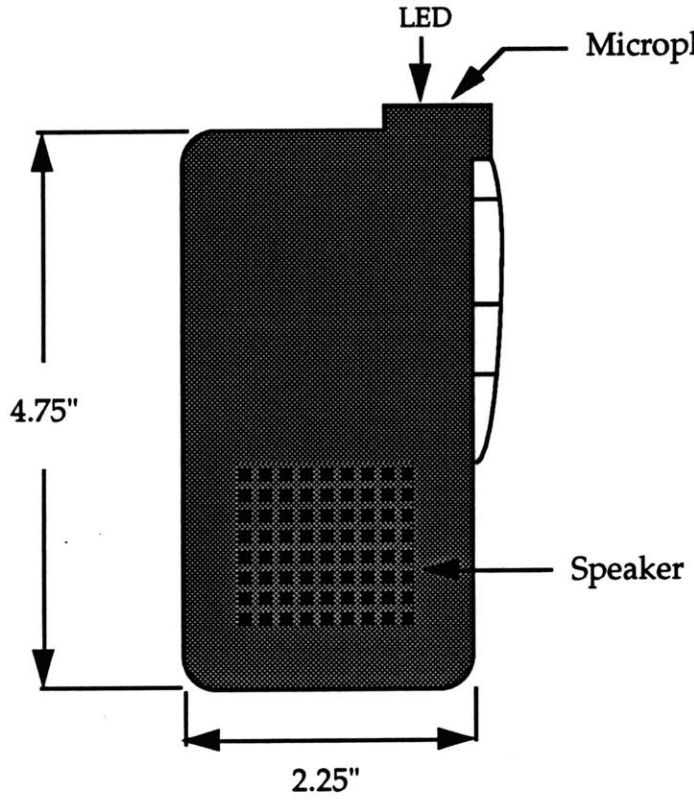
metaphor on the user. The words "*FIRST*" and "*LAST*" are temporal and imply the beginning and end of the list. A temporal reference is used rather than a spatial metaphor that might differ from the user's mental model.

5.5.6 Deleting/Undeleting

The ability to interactively "turn on" and "turn off" words in the recognition vocabulary in real-time is important for the VoiceNotes application. Whenever a list name is deleted, its associated template must be removed from the user's voice file. However, the template cannot be removed until another item is deleted by the user. Each time an item is deleted, it is saved by the system so that the user may "*UNDELETE*" this item at any time until the next item is deleted. The problem is that if the template is not removed immediately, there is a chance that the currently "deleted" list name will be falsely reported by the recognizer. Therefore, until the template can be removed from the voice file, it is deactivated (the vocabulary is subset to exclude the deleted word number). Once the next item is deleted, the template is removed and the word number reactivated. Once the template is removed, its associated word number is free to be used for new list names created by the user.

6. Button Input

This chapter discusses issues relating specifically to interaction with the VoiceNotes application through button input. Button input is implemented using prototype 1 of the hand-held device (Figures 6.1-6.3). The goal is to allow button control of as many of the basic VoiceNotes list management functions as possible given the design of prototype 1. Based on an evaluation of the button mechanics and layout of prototype 1, improvements for a second hand-held prototype are suggested and discussed.



Front View

Figure 6.1: Front view of hand-held prototype 1.

6.1 List Management with Buttons (Hand-held Prototype 1)

Hand-held prototype 1 uses the body of an Olympus S912 microcassette recorder. As such, the button interface design is limited by the number and type of buttons provided on this unit. The main set of original button functions on the microcassette recorder are: play, record, stop/eject, rewind, fast forward, pause on/off, and volume. These buttons are mapped to the VoiceNotes

functions as shown in Figure 6.4. Note that there are not enough buttons on the prototype 1 hand-held unit to support the functions delete, undelete, and move. These functions are "voice-only" in prototype 1, but will be supported with both voice and button input in prototype 2.

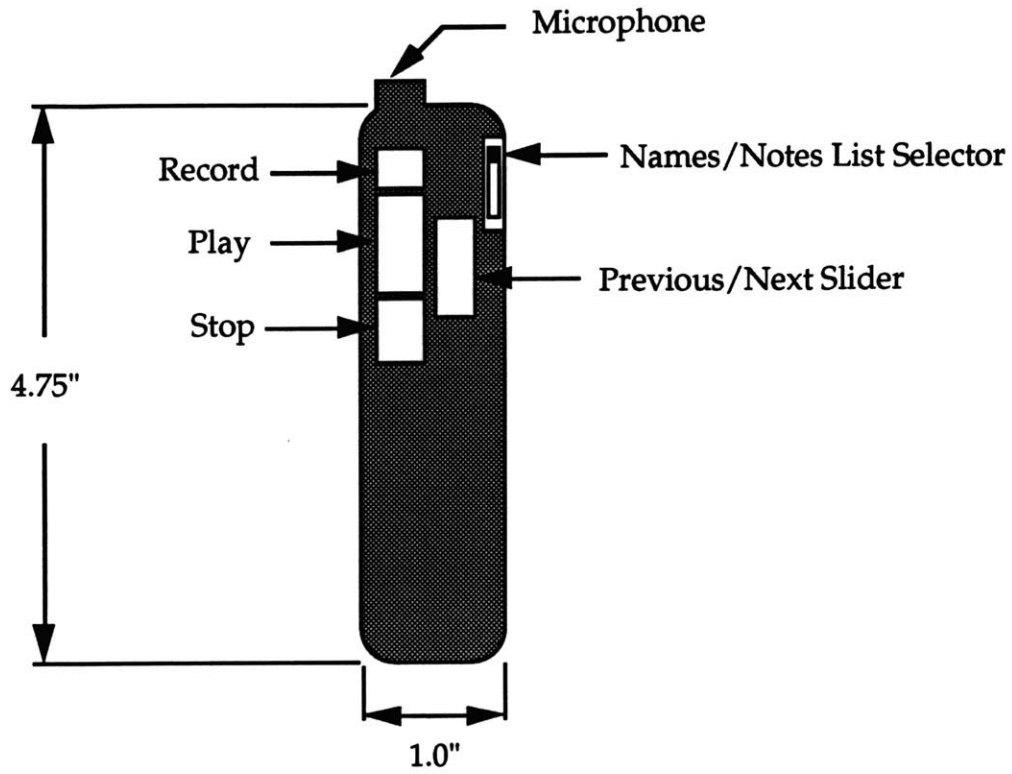


Figure 6.2: Side view of hand-held prototype 1.

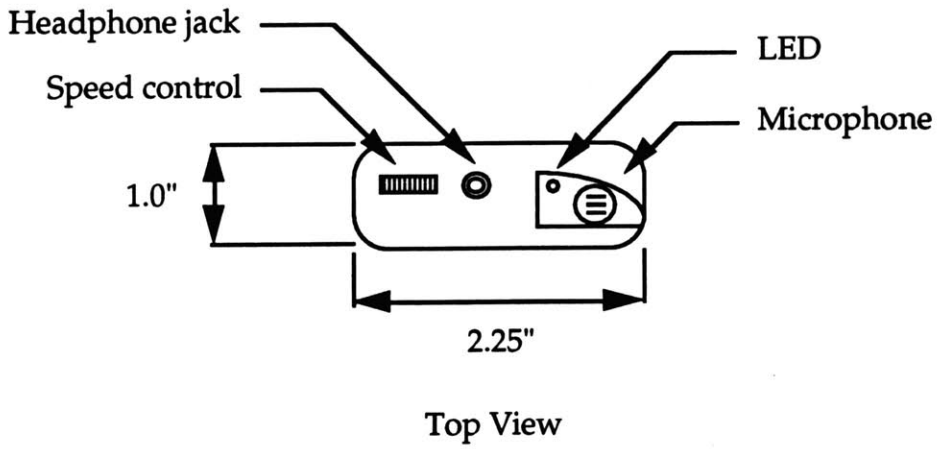


Figure 6.3: Top view of hand-held prototype 1.

Olympus Button	VoiceNotes Function
Play	*PLAY*
Record+Play	*RECORD*
Stop/Eject	*STOP*
Rewind/Fast Forward	*PREVIOUS/NEXT*
Pause On/Off	*NAMES/NOTES* list selector
Volume	Speed control

Figure 6.4: Button mappings of original Olympus buttons to VoiceNotes functions.

6.1.1 Prototype 1: Button Mechanics

The prototype 1 hand-held unit has mechanically interlocking buttons typical of a standard tape recorder. In order to engage the *RECORD* button, *PLAY* and *RECORD* must be pressed simultaneously. When *PLAY* and *RECORD* are pressed, they lock into position; pressing the *STOP* button releases them. Rewind and fast forward (mapped to *PREVIOUS* and *NEXT*) are controlled by a slider mechanism. When the slider is pushed up, rewind is engaged; when pushed down, fast forward is engaged. The slider operates differently depending on whether the *PLAY* button is engaged. If *PLAY* is not engaged, the slider locks into position when pushed up or down and is released by pressing the *STOP* button. When *PLAY* is engaged, the slider becomes spring loaded and always returns to center when not held in position. Pause on/off (used for list selection) is a two position switch that locks into place.

The button mechanics of prototype 1 proved to be inappropriate for supporting both voice and button interaction with the VoiceNotes application (see section 6.2.1). Because the buttons lock into position, it is possible for the software and the hardware to get out of synchronization. Although the button mechanics on this unit need improvement, prototype 1 allows the exploration of button interaction with VoiceNotes as part of an iterative design process.

6.1.2 Recording Notes and Names: Button User Interface

From the user's standpoint, recording notes and names using button input is exactly the same as when using voice input, with two notable exceptions: (1) terminating the recording, and (2) canceling the recording.

When the user issues the "*RECORD*" voice command, recording is terminated when the user stops speaking. In contrast, when the user begins recording a note by pressing *RECORD* on the hand-held unit, recording is terminated by pressing the *STOP* button. In the first case, the system determines termination, in the latter, termination is controlled by the user.

In order to cancel a recording initiated by the voice "*RECORD*" command, the user must remain silent until the system detects an initial pause (i.e., the user cannot issue the "*STOP*" voice command). When recording is initiated by pressing *RECORD*, it can be canceled by pressing the

STOP button, as long as the user has not already begun speaking. As with termination, the user controls cancellation when buttons are used, the system determines cancellation when voice is used.

The next two sections discuss technical issues related to recording notes and names using button control of the interaction.

6.1.3 Recording Notes: Button "Click" and Silence Removal

When recording is initiated by button input, the pause detection algorithm is used for determining initial and final silence durations in the utterance, but not for terminating the recording.

When recording is terminated with the stop button, final silence removal is problematic. The "click" sound made by the release of the stop button gets recorded at the end of the sound file. This means that if there is any silence prior to the button press, it will not be removed (Figure 6.5). Further processing is therefore required in order to remove both the button click and any final silence from the sound file. Since the button press has a fairly constant duration, it is removed by truncating the final 200 msec of sound data.

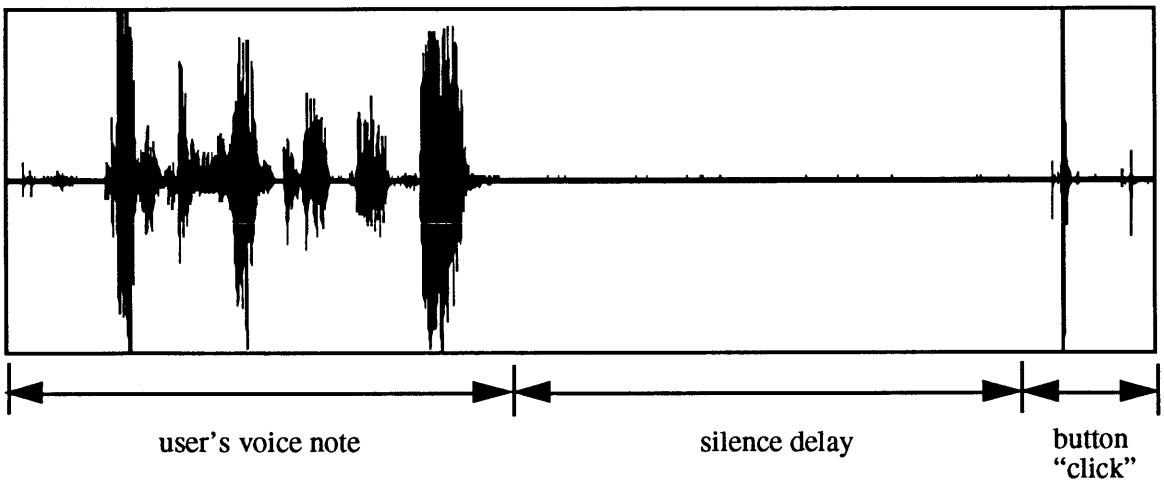


Figure 6.5: This is an example of a voice note that was terminated by pressing the *STOP* button. There is a delay between the end of speech and the button press terminating the recording (silence delay).

If the amount of data remaining after button click removal is less than 400 msec, it is assumed that recording was canceled by the user, since it is unlikely that an entire voice note would be less than this length. If there is sufficient sound data remaining following the removal of the click, any silence occurring prior to the time that the user presses the stop button (silence delay) is removed. Processing from the end of the sound data to the beginning, the average magnitude is calculated every 100 msec and passed to the pause detection algorithm that returns whether speech or silence is contained in each interval. Once speech is detected, the process is terminated, and the cumulative amount of silence is removed from the end of the sound file.

6.1.4 Recording Names: Terminating and Canceling

Recording of a name is terminated using information supplied by the Recognition Server. This is true whether recording is initiated using voice or button input. The end points of the utterance are determined by the recognizer, therefore, termination cannot be controlled by the user.

When recording a name using button input, the user most likely expects to terminate the recording by pressing the *STOP* button (as when recording a note). This is especially true of prototype 1, given that the *PLAY* and *RECORD* buttons are not released until the *STOP* button is pressed. One solution may be to wait for the user to press the *STOP* button before prompting "New name added." However, this may falsely lead the user to believe that recording is still taking place. Another solution is to continue recording the name (until the stop button is pressed) even after the training utterance has been collected. In this case, the recording of the name may not match the utterance collected by the recognizer. For example, the user could record the name "Demos this week" while the recognizer collected only "Demos." Therefore, poor recognition accuracy may result if recording is not terminated as soon as a template has been collected.

Currently, once a voice template has been collected, recording is terminated immediately, and the user is prompted, "New name added." Using prototype 1, the user must then press the *STOP* button in order to release the *RECORD* and *PLAY* buttons. This problem of "synchronizing" the termination of recording with the termination of training may be alleviated by eliminating mechanically interlocked buttons.

If the user presses the *STOP* button before a template has been collected, the recording is terminated and the user is prompted, "Record canceled." The user can terminate the recording at any point until speech has been detected by the recognizer.[23] This differs from voice initiated name recordings that are canceled if no speech has been detected within four seconds.

6.1.5 Navigation with Buttons

The user is given the ability to navigate both within and between lists using button input. Navigating within a list is accomplished using the *PREVIOUS/NEXT* slider (Figure 6.2). Pushing the slider up activates *PREVIOUS*, while pushing the slider down activates *NEXT*. *PREVIOUS* and *NEXT* are intended to be discrete actions—a single voice note (the next or previous one in the list) is played and then the system stops to wait for button or voice input from the user. One problem caused by the button mechanics of prototype 1 is that when *PLAY* is not engaged, the slider remains in a previous or next position until *STOP* is pressed. This implies a continuous action (continually moving up or down the list). For example, when the slider is engaged in the previous position, the user may expect the list to continuously play in reverse order until *STOP* is pressed.

[23]There is, however, an upper limit of 9 seconds. If the *STOP* button has not been pressed and no speech has been detected after 9 seconds, recording is canceled by the system.

Movement between lists (list selection) is accomplished using a *NAMES/NOTES* list selector button (Figure 6.2). In order to move into the names list, the user slides the button up into the names position. This differs from the voice "*NAMES*" command in one respect—the names list is selected but the names do not automatically begin to play. This design decision was made based on the mechanics of prototype 1 and will be changed in prototype 2 to match the voice interface. In prototype 1, pressing the *STOP* button does not have an effect unless either the *PLAY*, *RECORD*, or *PREVIOUS/NEXT* slider is engaged.[24] Therefore, if the list automatically begins to play without the *PLAY* button engaged, the user cannot stop playback with the *STOP* button. Once the list has been selected, the user can press the *PLAY* button to hear the names.

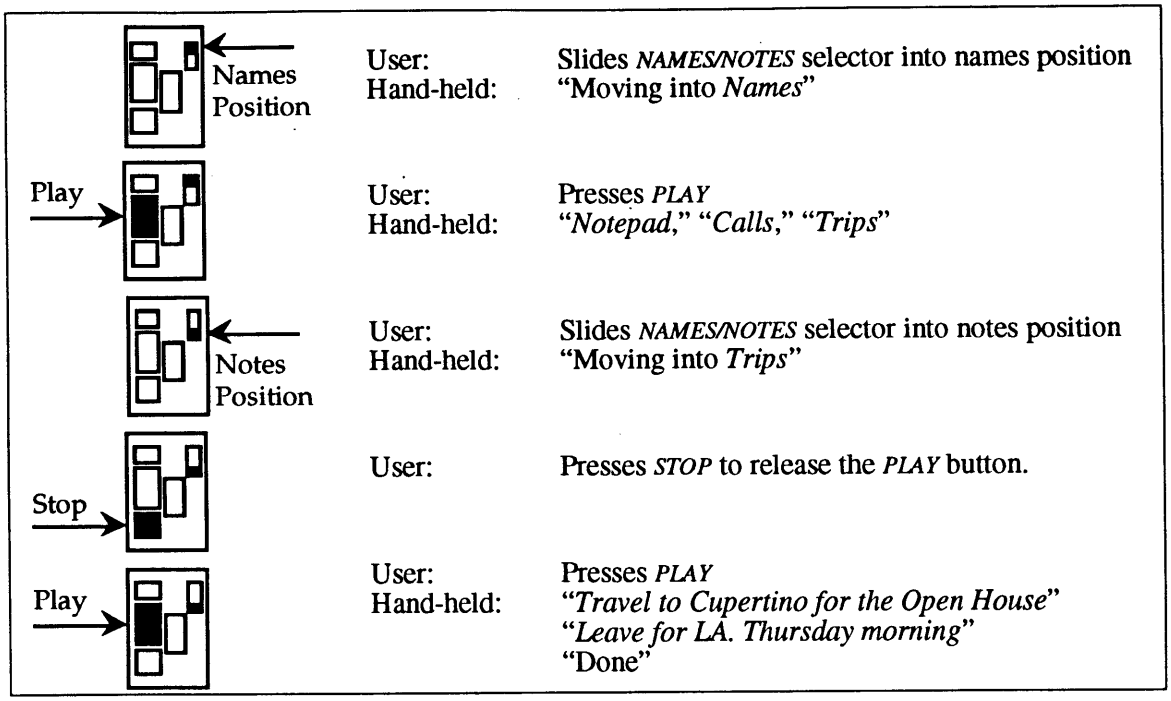


Figure 6.6: When the desired list name is heard, the user slides the list selector down into the notes position. In this example, the *PLAY* button is still engaged from playing the names list. Therefore, the user needs to press *STOP* to release the *PLAY* button before it can be pressed again to play the "Trips" voice notes.

[24]Pressing the *STOP* button does not cause a two byte message to be sent to the Macinotsh unless it disengages the *PLAY*, *PREVIOUS/NEXT*, or *RECORD* button.

6.1.6 Speed Control

Button control is provided for changing the speed of playback. This is a "button-only" action—no voice commands are provided for changing the speed. The goal of time-compressing the spoken output is to play it back in less time than it took to record. Since speech is slow and serial, this gives the user the opportunity to obtain the same information at a faster rate.

In prototype 1, the rate of playback is adjusted using a potentiometer control located on top of the hand-held unit (Figure 6.3). When the VoiceNotes application begins, the rate is set to normal speed. The speed of playback can be increased up to five times the speed of the original recording (a compression of 20 percent). However, research suggests that compression of more than 50% presents too little of the signal in too little time for a sufficient number of words to be accurately perceived [Heiman 1986].

It is important to provide the user with interactive control of the rate of playback. The rate does not have to be set prior to playing a list of voice notes, but may be changed interactively as the list is playing. This is important as the speed control may be used for two different purposes. One use of the speed control is to allow users to select a fixed rate of playback that they find comfortable for normal listening. However, the user may also adjust the speed of playback in order to browse a list, speeding up during some portions and slowing down when a note of particular interest is located. The latter is a more interactive use that requires the capability to change the rate of speed as the list is playing.

6.1.7 Linear vs. Non-linear

The first implementation of the speed control provided a linear mapping of movement to changes in the rate of playback. However, the granularity of control required for changing the rate of playback may differ over different intervals of speed.

Two approaches were taken for the design of a non-linear speed control. One rationale is that as the speed increases, the granularity of control over the change in speed should also increase. This type of non-linear control can be approximated with a logarithmic function (Figure 6.7). The log curve, however, does not provide enough granularity between a time compression ratio of 1.0 and 2.0, since the slope of the curve is too steep between these values. Therefore, a better approximation can be achieved using a piecewise linear function (Figure 6.8).

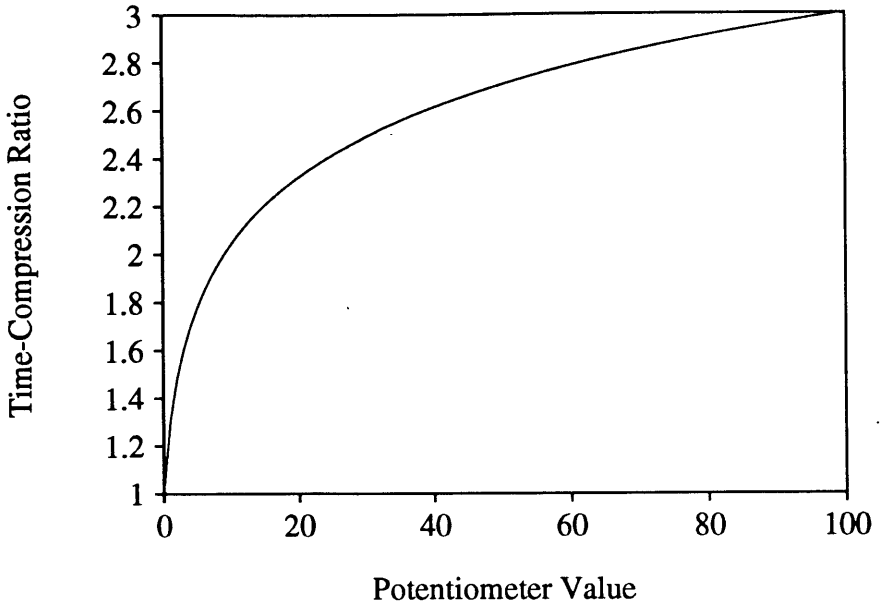


Figure 6.7: A logarithmic function for mapping the movement of the speed control (values returned by the potentiometer) to time-compression ratios.

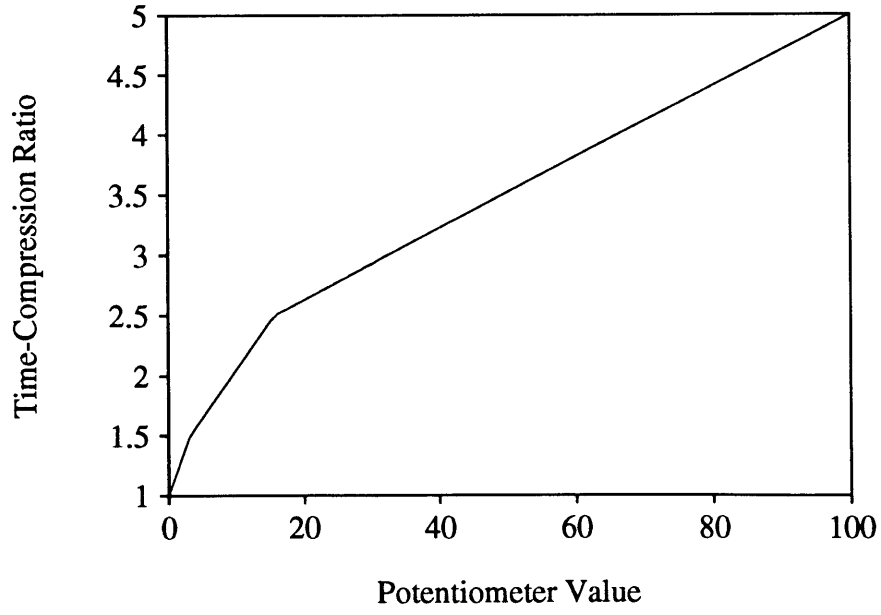


Figure 6.8: Approach 1: A piecewise linear function that increases in granularity (slope decreases) as the time-compression ratio increases.

A second approach for the design of a non-linear speed control involves determining the **range** of speeds (ratios) in which the finest control is needed. This type of non-linear control can also be approximated using a piecewise linear function (Figure 6.9). The **lower limit** of the range is the value at which the change in speed provides a significant time savings. For example, although the user may perceive a change in speed from 1.0 to 1.2, from the user's perspective, this change may not significantly decrease the amount of time required to listen to the speech output. The change in speed until this lower limit is reached may be considered a "ramp up" interval in which fine granularity is not needed. The **upper limit** of the range is the fastest comfortable listening value for each user. Above this value, fine control may not be required.

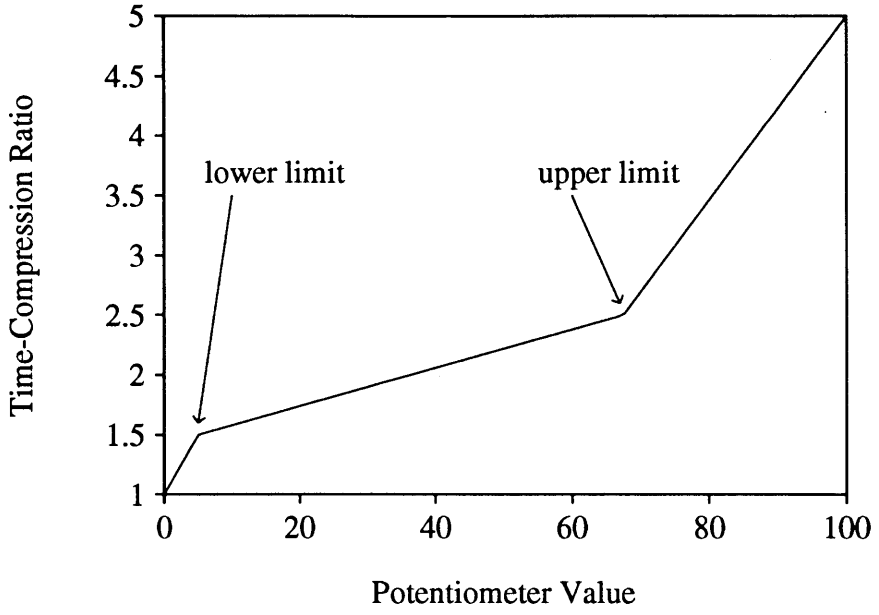


Figure 6.9: Approach 2: A piecewise linear function that provides the finest movement control over a given range (marked by the lower limit and upper limit).

Currently, VoiceNotes uses the second approach (Figure 6.9). However, the upper and lower limits, marking a range in which the finest control over speed is provided, will vary for users of different experience levels and amounts of exposure to time-compressed speech. Therefore, the lower and upper limits that compose each user's time-compression "range of interest" can be set in the user's configuration file (see section 4.5.2, figure 4.21).

Further evaluation is needed in order to determine the optimal design for a non-linear speed control (see also section 10.3.2).

6.1.8 Technique Used for Time-Compression

There are many techniques for changing the rate of playback of speech without changing the pitch.[25] The time-compression method used by VoiceNotes[26] is a time domain technique referred to as **isochronous sampling** [Fairbanks 1954]. The sampling technique used is isochronous in that segments of speech are removed at regular intervals, as opposed to **pitch-synchronous selective sampling** in which redundant pitch periods are selectively removed during voiced portions of the speech signal [Portnoff 1978].

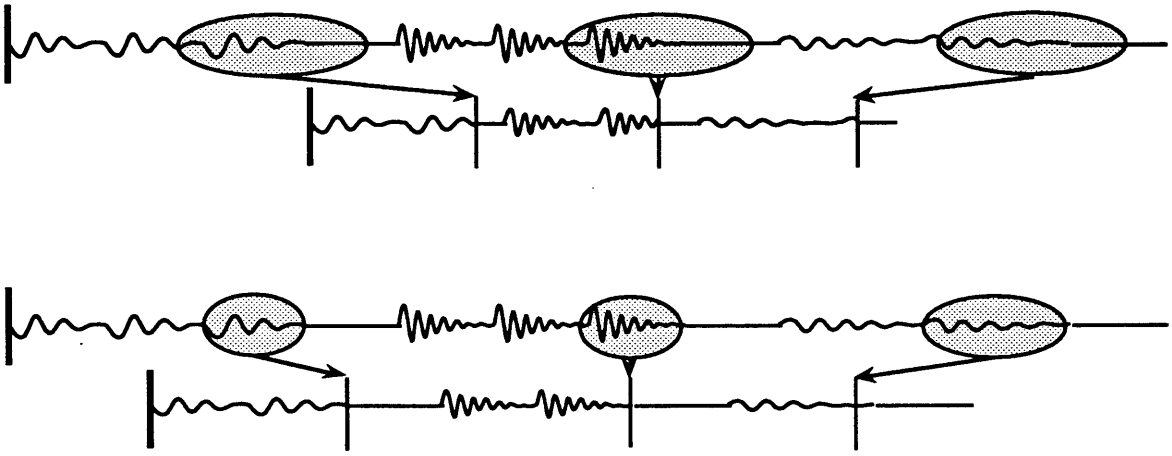


Figure 6.10: Time-compression techniques. Top: Isochronous sampling. Bottom: Pitch-synchronous selective sampling.[27] The isochronous sampling method is used by VoiceNotes.

Simply removing portions of the signal at regular intervals and abutting the remaining speech segments results in a "burbling" distortion at interval boundaries [Portnoff 1978] [Portnoff 1981]. In VoiceNotes, this is partially alleviated by performing a linear cross-fade at interval boundaries—the last 10 ms from the speech segment prior to the discarded interval are blended with the first 10 ms of the segment following the discarded interval.

6.2 Button Interface Design (Hand-held Prototype 2)

This section describes the button interface design and layout for a second hand-held prototype.[28] The goal is to improve the button interface provided by prototype 1 and to provide a complete set of button interactions with the VoiceNotes application. All basic VoiceNotes list management capabilities (including delete and undelete) will be accessible by button input in prototype 2.

---

[25]For a summary of time-compression techniques and important references in this area, see [Arons 1992].

[26]The time-compression code used by VoiceNotes was written by Barry Arons of the Speech Research Group.

[27]Note that this is a conceptual representation of pitch-synchronous selective sampling and does not attempt to depict Portnoff's implementation.

[28]This prototype is currently in the design phase, and has not been built yet.

Ultimately the VoiceNotes application will be operable by buttons alone, voice alone, or a combination of voice and button input.

6.2.1 Improved Button Mechanics

The two most significant changes to the button mechanics are: (1) the elimination of buttons that stay in position, and (2) the elimination of interlocking between buttons. It is important that the buttons not remain in a fixed position after being pressed. In the example shown in Figure 6.11, the user presses the *PLAY* button and the list is played until completion without interruption. The list has finished playing, but given the button mechanics of prototype 1, the *PLAY* button remains in position. The state of the hand-held hardware is now out of synchronization with the state of the VoiceNotes software.

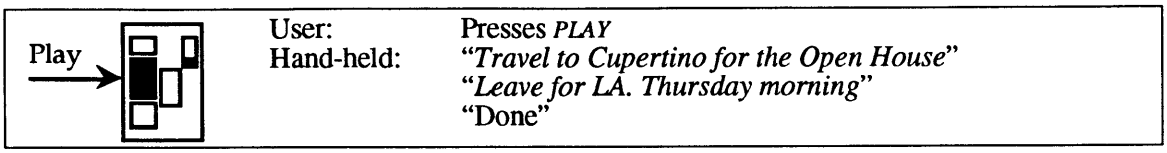


Figure 6.11: In this example, the *PLAY* button remains engaged even after the list has finished playing.

This also has implications for a hybrid voice and button interface. For example, if the user presses the *PLAY* button and says "*STOP*," the *PLAY* button remains engaged even though playback has stopped. This problem becomes even more critical for list selection. The list selector may be in the notes position when the user is actually in the names list. Not only does this give the wrong visual indication to the user, but it forces the user to have to synchronize the hardware with the software by moving the selector into the correct position the next time it is needed. It is also undesirable for the *RECORD* button to remain in position, since recording can be terminated for a number of reasons other than the user pressing the *STOP* button.

The next important design change is the elimination of button interlocking. These interlocks are an artifact of the original analog microcassette recorder body used in prototype 1. For example, for digital recording, there is no reason to require that the *PLAY* and *RECORD* buttons be pressed simultaneously to initiate recording. Another problem is that the *STOP* button is currently dependent on the *PLAY*, *RECORD*, and *PREVIOUS/NEXT* buttons; the *STOP* button only sends a two byte code when it disengages one of these buttons. Given a hybrid voice and button interaction, if the user says "*PLAY*," pressing *STOP* will not have an effect since the *PLAY* button is not engaged. Therefore, in prototype 2 it is important that each button operate in an independent manner.

In addition, it is important that the buttons operate in a consistent manner. In prototype 1, due to interlocking, the *PREVIOUS/NEXT* slider operates differently depending on whether the *PLAY* button is engaged.

On the basis of the above criteria for improvements over prototype 1, and the requirement of supporting all basic VoiceNotes functions, buttons are suggested for prototype 2 (Figure 6.12).

Button Function	Type of Button
Play	momentary push button
Record	push-to-talk
Stop	momentary push button
Previous/Next	spring loaded slider
Names/Notes List Selector	2 position rocker with spring back
Speed	continuous (potentiometer)
Volume/Off	continuous; off at bottom (potentiometer)
Delete, Undelete	2 position rocker with spring back or 2 momentary push buttons

Figure 6.12: A list of button functions that will be supported in prototype 2, with a suggested type of button for each function.

6.2.2 Improved Button Layout

In addition to improving the button mechanics of prototype 1, the button layout can also be improved. In selecting a button layout for prototype 2, the following are some factors need to be considered:

1. Frequency of use of each button
2. Buttons that have related functionality, or may be used in conjunction with one another
3. Buttons that need to be protected from accidental activation.

PLAY, *RECORD*, *STOP*, and *NEXT/PREVIOUS* will probably be the most frequently used button controls for the VoiceNotes application. Therefore, these controls should be the most accessible by the user.

However, it is also important to place buttons that may be used in conjunction with one another within close proximity. For example, in prototype 1, the speed control is on the top of the device, while the buttons for playing and navigating among voice notes are on the side of the device. Therefore, it is awkward to adjust the speed of playback while navigating; two hands are required, one for operating the *PREVIOUS/NEXT* slider and the other for changing the speed. In prototype 2, these controls should be placed in closer proximity to one another.

It is important that buttons like *DELETE* and *UNDELETE* are not accidentally activated. One solution to this problem is to place these buttons behind a protective cover that can be "flipped" open when the buttons are needed. This may be acceptable, since *DELETE* and *UNDELETE* are not frequently used functions. Figure 6.13 provides a conceptual drawing of such a design. The device is similar in appearance to a cellular telephone. The "flip-down" partition of the device not only protects the buttons behind it, but also allows the user to speak into the hand-held device as one does a telephone. In public situations, the user may feel more comfortable speaking into a "telephone-like" device than into a hand-held computer. In addition to privacy, this also provides noise-cancellation.

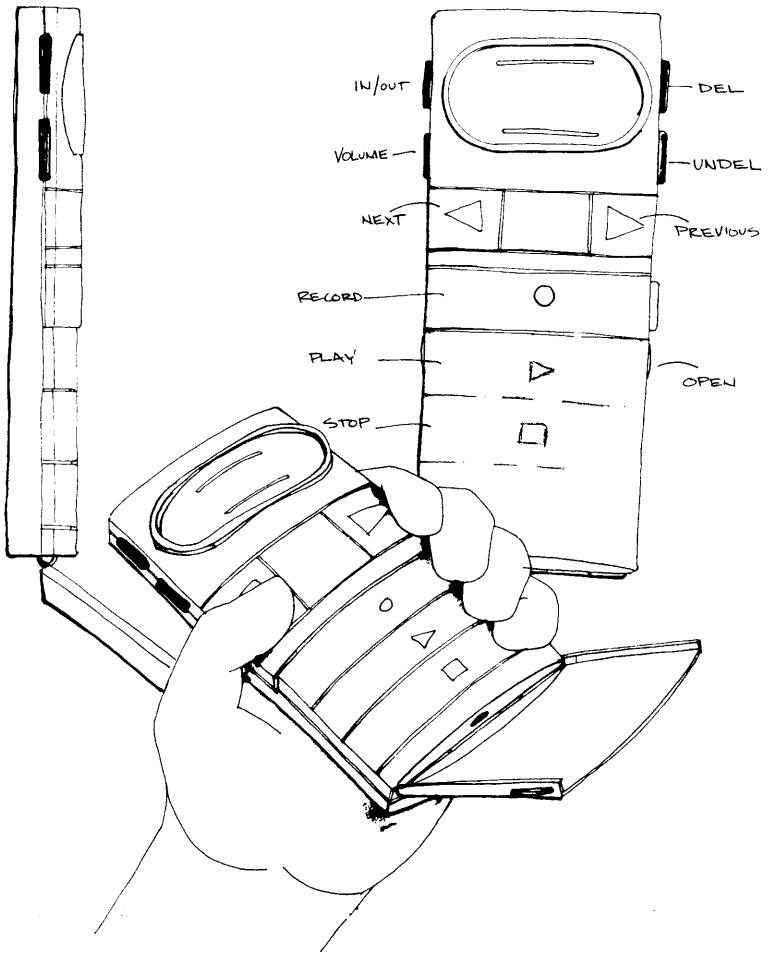


Figure 6.13: Conceptual design of hand-held prototype 2. (Drawing ©Apple Computer, Inc. 1992)

One problem with the design shown in Figure 6.13, is that the user will not be able to talk or listen and press the buttons on the front of the device at the same time. This problem can be solved by placing the buttons on the side of the device. In the current hand-held prototype, most buttons are on the side of the device. This is desirable since the buttons can be easily reached using one hand, but are not accidentally activated. An alternate design, with buttons on both sides of the device, is shown in Figure 6.14. Ultimately a hybrid design, with some buttons on the side of the device and others on the front, may be desirable.

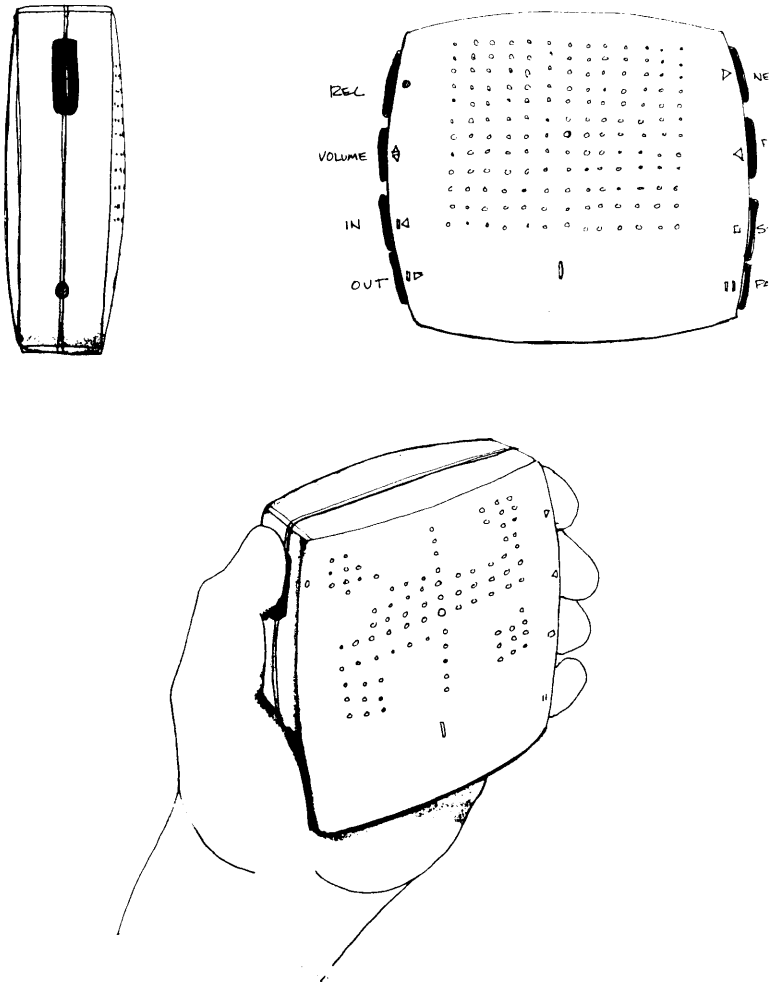


Figure 6.14: Conceptual design of hand-held prototype 2. (Drawing ©Apple Computer, Inc. 1992)

7. VoiceNotes Hybrid Interface: Voice and Buttons

While Chapter 5 discusses voice input and Chapter 6 discusses button input, the final VoiceNotes interface combines both types of interaction.

A hybrid interface provides the combination of two complimentary input modalities, voice and buttons. This allows the user to take advantage of the different capabilities provided by each input modality while allowing the limitations of one type of input to be overcome by the other.

A hybrid interface is advantageous in that it provides a flexible interface that can be operated using voice alone, buttons alone, or a combination of voice and button input. This flexibility is important, since the user's selection of how to interact with the application at any given time may depend on: (1) the task (in this case, the specific list management task); (2) the context (e.g., social situation, acoustic environment); (3) individual user preference. The following sections discuss the advantages of a hybrid voice and button interface given these three factors.

7.1 Task-Based Advantages

For performing specific list management tasks, voice and buttons are complimentary input modalities. While voice provides random access between lists, buttons provide fine movement control within a list. Therefore, the task of locating a particular voice note might involve the use of both voice and buttons (e.g., the list is selected using voice, and the particular voice note is located using buttons). In addition, limitations of voice input can be overcome using button input. For example, buttons support fine adjustments (e.g., speed control) which are awkward to control with voice. In turn, limitations of button input can be overcome using voice input. For example, the number of buttons is limited by physical constraints, while voice can support a much larger command set.

The following sections detail the complimentary advantages of voice and buttons for the VoiceNotes list management interface.

7.1.1 List Selection by Name with Voice

Voice provides the ability to select a list by name. Users are given control over both the number and the names of the lists. While voice can provide a one-to-one correspondence between each list and the command for accessing it (the list name), buttons cannot. Currently, a maximum of 50 lists can be created. In order to achieve a one-to-one correspondence between lists and buttons, 50 list selection buttons would be required. In addition, the meaning of each button

would probably be cryptic. The buttons could be numbered, like those on a speed dialer, but then the user must remember the correspondence of each button number to the associated list.

Since a one-to-one correspondence between lists and buttons is not practical, a single two position list selection button is used instead (described in section 6.1.5). This requires the user to listen to the list names in order to select a list. Therefore, list selection by button is likely to be slower than list selection by voice. Figure 6.6 (section 6.1.5) shows an example of list selection by button. This task can be accomplished using only a single voice command ("*TRIPS*"). Voice commands make it possible to jump directly from one list to another, while button commands require an intermediate step (the user must first move into the names list).

7.1.2 Coarse Movement with Voice

Voice provides the ability for "coarse" movement between and within lists of voice notes. The user can jump from one list to another simply by speaking a list's name. Voice also provides coarse movement within a list, by allowing the user to jump directly to the first, middle, or last item in a list. This functionality could be provided by button input, however, it would not be as "goal directed." The user would have to first locate the first, middle, or last command button on the device. If the voice commands are intuitively worded, less translation has to be made between the user's goal and the steps for accomplishing it than with other methods of input.

7.1.3 Fine Movement with Buttons

While voice input provides coarse movement control within a list of voice notes, button input provides fine control. The *PREVIOUS/NEXT* slider allows the user to easily move forward or backward through a list (this is normally accomplished using the thumb to move the slider up and down). This type of fine movement is awkward to achieve using voice input. For example, in order to move forward three items in a list, the user must speak "*NEXT, NEXT, NEXT,*" making sure to pause sufficiently between each command. This task can be accomplished with much less effort using button input.

7.1.4 Larger Command Set with Voice

Voice can support a larger command set than buttons. The number of button commands is limited by physical space considerations. In order to put more buttons on the hand-held device, a larger device and/or smaller buttons would be required. This is the problem with many portable computers and devices; they provide the user with a bewildering array of small and cryptic buttons. Therefore, this limited space must be reserved for those functions best suited for button input. Currently, there are roughly 27 voice commands and 8 button commands.

7.1.5 Command Synonyms with Voice

Command synonyms are another advantage of voice input. Several words or phrases can be used to activate the same functionality. While the "*RECORD*" command may be intuitive to some users, others may prefer the "*ADD*" command. In addition, since speaker dependent recognition is used, users can select their own commands simply by speaking the desired word when prompted during training (e.g., when prompted to say "*FIRST*" the user could say "*TOP*"). In contrast, if a particular button is not intuitive to a user, there is no way to re-configure it.

7.1.6 Fine Control with Buttons

While voice can support a larger number of functions than buttons, buttons can provide the user with finer control over a single function, such as speed. Button input is more appropriate than voice for making fine adjustments to control volume and speed of playback. Buttons can provide continuous control over the speed of playback, voice can only provide discrete adjustments. For example, to control speed using voice would require command sequences such as "*FASTER, FASTER*" or "*SLOWER, SLOWER.*" The only way to provide a continuous voice control would be moded (e.g., the user could say "*FASTER*" and the system could continually increase the speed until the user commands "*STOP*"). In addition, it is likely to be more difficult to correct "overshoot" errors using voice than when using buttons (i.e., the resulting speed is slightly slower or slightly faster than desired). Since voice control will not be as fine as button control, it is also more likely that an overshoot error would occur.

7.1.7 Interruption with Buttons

Users may find it easier to interrupt spoken output using button input rather than voice input. One reason is that there may be some reluctance to interrupt the spoken output with spoken input. Social protocol tells us to wait until someone is finished speaking before interrupting them. However, this may not apply for the VoiceNotes application since most of the spoken output is the user's own voice. It is also difficult to "talk over" the spoken output when attempting to interrupt the system. If users cannot clearly hear their own voice speaking a command, they might change their voice quality, causing recognition errors (e.g., by speaking louder). Spoken output is less likely to interfere with the user's ability to press a button.

7.1.8 Terminating and Canceling Actions with Buttons

As discussed in section 6.1, button input provides the user with the ability to cancel recording, while voice does not. In addition, when using button input to initiate recording of a voice note, the user controls termination. When using voice input, the system determines termination.

Button input could also be useful for canceling actions due to recognition errors. When a critical substitution error occurs (e.g., the system falsely recognizes the "*RECORD*" command), the user may speak in an agitated tone of voice causing the recognizer to reject the utterances spoken in an

attempt to cancel an action. In addition, the user is unlikely to speak in isolated words, and may use words that are outside the vocabulary under these circumstances. A stop or cancel button on the hand-held device could be used in such situations.

7.2 Context-Based Advantages

In addition to task-based advantages, a hybrid interface also supports interaction given a variety of contexts. Context refers to the acoustic environment, social situation, and task-context (i.e., other tasks that are being performed) in which the interaction is being performed. The following sections provide some example of different contexts that can be supported when both voice and button input are provided for interacting with the hand-held device.

7.2.1 Acoustic Context

Button input allows the hand-held to be operated under noisy conditions. For example, a user may wish to review a list of voice notes containing an agenda during a lunch meeting in a noisy restaurant. Attempts to use voice input may be frustrated due to recognition errors caused by background noise. Since the agenda list can be selected, played, and navigated using button input, the VoiceNotes application is still usable even given the noisy environment of the restaurant.

7.2.2 Social Context

Button input supports the use of VoiceNotes given social contexts in which it is inappropriate or awkward to speak aloud to one's hand-held computer. For example, perhaps a user desires to review a list of voice notes when riding in the elevator of his/her office building. While headphones can be worn for private listening, speaking commands out loud could be extremely awkward.

In contrast, voice input may be appropriate for other social interactions in which one or more people are cooperatively using the application. For example, several users might be reviewing some speech that has been captured during a brainstorming session. Since only one person will be controlling the device, voice commands may be helpful for "announcing" the actions that are being executed for the other participants.

7.2.3 Task-Context

The type of interaction used (voice or button input) is also dependent on other tasks that may be performed in parallel with the task of recording and reviewing VoiceNotes. For example, the VoiceNotes application may prove particularly useful while riding in one's car. This is normally "wasted" time for people who have to commute to work each day. When driving, one's hands and eyes are busy, making it impossible (or extremely difficult) to operate a traditional computer.

Given this context, voice input allows users to operate the VoiceNotes application without requiring them to take their eyes off the road to press a button or view a visual display.

7.3 User-Preference-Based Advantages

While some users will prefer voice input, others will prefer button input, regardless of the aforementioned advantages for each input modality. Some users may feel awkward talking to their computer in any kind of public situation (e.g., even while walking down the street). In contrast, other users (e.g., executives who are accustomed to dictating information) may prefer speaking their commands at all possible times. By providing both types of input, users are given the ability to decide which input modality to use, not only on the basis of the given context, or task performance differences, but based on their personal preference.

7.4 Design Considerations for a Hybrid Interface

Designing a hybrid interface does not merely involve providing two separate input modalities. When combining voice and button input, careful consideration must be made as to how the two modalities can be integrated and work in conjunction with one another. The following lists some of the factors that were considered when combining voice and button input for use with the VoiceNotes application:

- Consistency. It is important that voice and button interactions be consistent. If the user performs an action using voice commands, button commands, or a combination of the two, the feedback should be consistent. For example, if the user initiates recording using the *RECORD* button or the "*RECORD*" voice command, the feedback is the same (e.g., "Recording note"). However, since button input is more reliable than voice input, less verbose feedback may be needed. Due to the occurrence of recognition errors, feedback in response to voice input is more critical than for button input.

  Some differences between voice and button interactions are necessary due to the differences between the two modalities. For example, since there cannot be a button for every list name, list selection using buttons must differ from list selection using voice. Using voice, a list is selected by speaking its name. Using buttons, a list is selected using the *NAMES/NOTES* list selection button.

- Voice and buttons. The user should be able to perform actions using a combination of voice and button input. For example, in order to review a list of voice notes, the user can say "*PLAY*," and then use the *PREVIOUS/NEXT* slider to move within the list, or press *PLAY* and then say "*PREVIOUS*" and "*NEXT*" to move within the list.

- Actions initiated by voice, terminated by button. Actions initiated by a voice command should be able to be terminated using a button command. For example, the user should be able to say "*PLAY*" or "*RECORD*" and press *STOP* to terminate the action.

- Actions initiated by button, terminated by voice. Actions initiated by a voice command should be able to be terminated using a button command. For example, the user should be able to press *PLAY* or *RECORD* and say "*STOP*" to terminate the action.

- Simple command set. In order to be able to support a basic set of functions (record, play, stop, previous, next, delete, undelete) using both voice and buttons, a simple command set is needed. While voice can support a large number of commands, buttons cannot. For example, it is desirable to have only a single record command even though there are different types of items that can be recorded (names, notes). The number of commands can be minimized by separating the action from the object (in this case the action is record and the object is a name or a note).

- Stateless buttons. If the VoiceNotes interface used only button input, it might be desirable to use buttons that stay in place once pressed. This gives the user a visual indication of the current state of the system. However, in order to allow hybrid interactions to take place without causing the hardware to become out of synchronization with the software, the buttons must be "stateless" (once pressed, buttons should not remain in position). For example, if the user presses *PLAY* and says "*STOP*," given the button mechanics of prototype 1, the *PLAY* button remains engaged even though the audio output has stopped. An example of a stateless button is a momentary contact; when the user presses the button, it does not stay in position unless the user continues to hold it in place.

8. Capturing Background Speech

> "It has been estimated that office workers spend between 15% and 20% of their time in verbal communication. If some of this already existing speech could be captured and acted upon automatically, financial savings and worker productivity might be achieved" ([Noyes 1989] p. 475).

In section 3.4, a taxonomy of voice notes is presented. The first distinction made is between intentional and unintentional or "on-the-fly" voice notes. Chapters 3-7 discuss the use of the VoiceNotes application for recording, retrieving, and, managing intentional voice notes. This chapter presents the capture and retrieval of "on-the-fly" voice notes and their integration into the VoiceNotes list structure.

8.1 What Did You Just Say?

Speech is transient by nature—a spoken utterance leaves no trace of its contents. During a conversation, brainstorming session, or when bumping into a colleague in the hallway, important information may be exchanged that cannot be retrieved in its original form. Often, it is only upon hindsight that one realizes that something important has been said. A portable hand-held computer that is with a person at all times can capture this information.

A good example of the need to capture background speech is during a conversation between two people who are co-authoring a paper. One person suggests the wording of a sentence that turns out to be "just right." The other person may make a vain attempt to quickly write down the suggested wording or may not recognize the value of the suggestion until it is too late. As such, the question posed is "What did you just say?" or "Can you repeat that?" The automatic capturing feature of the VoiceNotes application constantly records the last several minutes of background speech, allowing the user to replay the speech and extract segments as voice notes.

8.2 How is Capturing Accomplished?

Ambient speech is recorded into a digital "tape-loop" (Figure 8.1). Speech is recorded into a circular list of in-memory buffers. Once the memory is exhausted, buffers are re-used, overwriting the oldest previously recorded speech. Start and stop points are reset in order to maintain logical beginning and ending points through the list. As shown in Figure 8.1, recorded background speech might fill buffers 1-6. The next time background speech is recorded, the new speech data might overwrite buffers 1-2. On playback, the remaining portion of the original background speech data (buffers 3-6) will play followed by the new speech data (buffers 7 and 8).

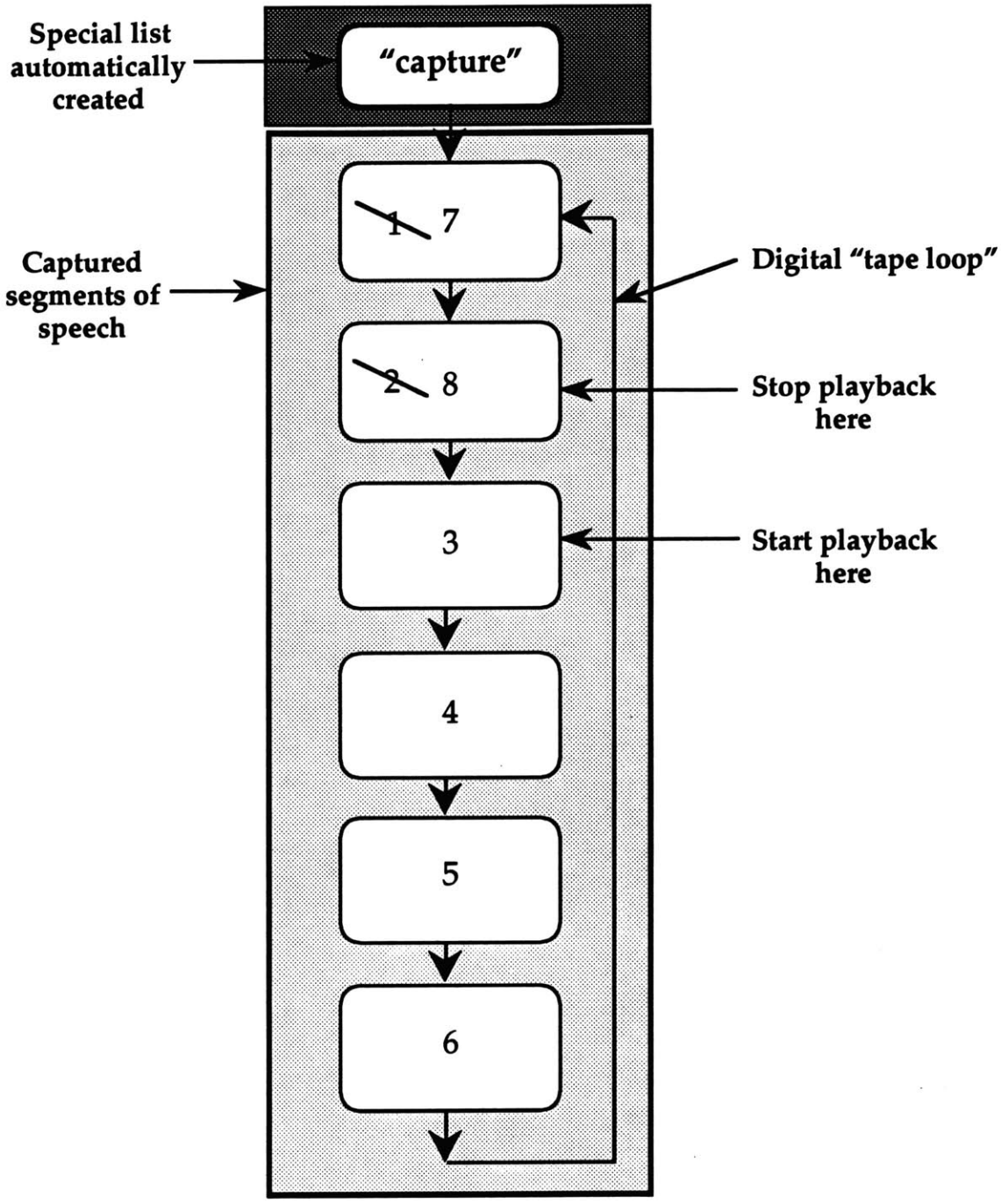


Figure 8.1: Speech is captured into a circular linked list of buffers. Once the memory is exhausted, buffers are reused and start and stop points are reset.

8.3 Memory Allocation

VoiceNotes allocates as much available memory as possible for recording background speech. While recording of intentional voice notes uses a 22 kHz[29] sampling rate, background speech is recorded at 7 kHz[30] in order to allow a larger amount of speech to be captured in memory. When using the Macintosh operating system, applications have a current size[31] associated with them, which places an upper bound on how large the application's heap can grow. Temporary memory is also available but is only intended for occasional short-term use [Apple 1991].

Currently, the maximum amount of memory allocated to the VoiceNotes application is 5 MB. This amount is calculated based on the following assumptions: (1) 8 MB RAM on the machine; (2) virtual memory is not used; (3) the Recognition Server uses 384K; (4) System Software uses a maximum of 2.6 MB, of which a maximum amount of 320K is reserved for the Voice Navigator's voice driver[32]; (5) no other applications are running. Of the 5 MB allocated to VoiceNotes, 4.5 MB is designated for capturing background speech, and the remaining 0.5 MB for the rest of the application. This corresponds to roughly 10.5 minutes of background speech at 7 kHz (three times the amount of speech that can be captured at 22 kHz).

Another possibility is to use data compression. At a 22 kHz sampling rate using 3:1 compression, recorded speech data will consume the same amount of memory as when recording at 7 kHz and recording quality may be significantly increased.

8.4 How is Background Recording Invoked?

The idea of recording background audio is to capture "spontaneous speech." Therefore, the user should not be required to explicitly invoke background recording (as is the case with intentional voice notes). Background recording is invoked under each of the following conditions: (1) at the start up of the application; (2) when the user has been idle for more than 30 seconds; (3) when the user issues the command "*STOP LISTENING*," speech recognition is deactivated and background recording is activated; (4) the user issues the *RECORD* command when "Capture" is the currently selected list (see section 8.5). In conditions 1-3, background recording is invoked automatically by the system. The user is also given the ability to manually start background recording (condition 4). The feedback is the same in each of the conditions: the system prompts "capturing speech," and the LED on the hand-held unit is turned on.

---

[29]The actual value is 22254.5454 Hz.

[30]The actual value is 7418.1818 Hz.

[31]The current size is set by default to the sugessted size specified in the application's 'SIZE' resource, but may be reset by the user.

[32]However, the voice driver can allocate 64K blocks in the system heap as needed, instead of using a large fixed allocation size [ASI 1991].

In section 5.4.2 the problem of differentiating background speech from spoken commands to the recognizer is discussed. This is a problem in each of the background recording conditions (except when the user commands "*STOP LISTENING*"). When recording background speech, if speech recognition is not deactivated, the recognizer may detect the background speech and falsely report the occurrence of a voice command. However, since the system is supposed to be "always listening," for input commands from the user, it is not desirable to turn off recognition during background recording. If the recognizer is always activated, the user can interrupt the background recording process at any time with a voice command.[33]

One proposed solution is the use of a mercury switch to detect when the hand-held is in an upright versus a horizontal position (as when laying flat on a table). When the hand-held is moved from an upright to a horizontal position, recognition would be automatically deactivated and background recording would begin (as if the user had explicitly issued a "*STOP LISTENING*" command). This approach has the following limitation: the user will not be able to issue voice commands unless the hand-held is in an upright position.[34] In addition, when in an upright position (such as when inside the front pocket of a shirt), there is still the possibility of background recording being interrupted due to an insertion error.

It was found that speech could be recorded while the recognizer is listening (without causing insertion errors), as long as the user does not speak directly into the microphone. In the current implementation, background recording is activated as described in conditions 1-4 without regard to the hand-held's current position. In condition 2 (background recording is activated after 30 seconds of idle time), there is a possible 30 second delay in beginning to record background speech. Combining this approach with position information from the mercury switch could eliminate this delay under certain conditions. Instead of using the position information to turn recognition on and off, the position information could be used to invoke background recording. Whenever the hand-held is in a horizontal position, background recording could be immediately activated, eliminating any delay.

However, a problem with the current implementation is that it assumes that the user will not speak directly into the microphone unless issuing a voice command or speaking an intentional voice note. A more robust solution for recording background speech while recognizing might be attained with the use of two microphones (see section 8.9). Thus, a future hybrid approach might involve the use of two microphones in addition to a mercury switch.

---

[33]Button commands are always active and can be used whether or not the recognizer is "listening."

[34]However, the characteristics of the current hand-held microphone require "close-talking" for accurate voice recognition.

8.5 The "Capture" List

Just as intentional voice notes are organized into a list structure, so are unintentional or "on-the-fly" voice notes. Background speech is recorded into segments (Figure 8.1 shows a list containing 6 segments of speech). Each segment is automatically entered into a list of voice notes with the list name "Capture." Although the underlying programmatic structure is a circular linked list, the user interface is a linear list with beginning and ending points, the same as all of the user's other lists of voice notes. The first and last voice notes in the list are determined as shown in Figure 8.1. In this list, segment 3 represents the beginning of the list, and segment 8 is the last item on the list.

The "Capture" list is accessed and managed the same as an intentionally recorded list of voice notes. One difference, however, is that issuing the *RECORD* command initiates the recording of background speech as opposed to the recording of an intentional voice note. There is no limit to the recording duration; recording can continue indefinitely. Since the recognizer has not been disabled during background capturing of speech, recording can be terminated by any voice or button command.

An important difference between the automatically created "Capture" list, and intentional lists of voice notes created by the user, is that it is temporary; the data in the list is constantly being overwritten. For this reason, background recording is not automatically invoked (even after 30 seconds of idle time) as long as "Capture" is the currently selected list. For example, the user may be reviewing the "Capture" list, searching for a particular set of speech segments to save (see section 8.8), when interrupted by another activity. If the interruption lasts for longer than 30 seconds, the set of speech segments that the user was in the process of saving could then be overwritten with new speech data. This problem is resolved by requiring manual initiation of background recording when "Capture" is the current list in use.

8.6 Version 1: Fixed Segment Size

Even though the ultimate goal is to segment background speech based on pauses (see section 8.7), in the first version of background recording, a fixed segment size was used. The goal was to first concentrate on the integration of automatically captured speech data into the existing structure for intentionally recorded voice notes.

Each voice note recorded automatically into the "Capture" list was 20 seconds in length. Since each segment was treated as a separate voice note, the list playback sounded "choppy." Each 20 second speech segment was played followed by 0.5 seconds of silence (as is the case with intentional voice notes). In addition, no discrimination was made between speech and silence. Memory management was simple when using fixed segment sizes. Twenty second segments were allocated until the memory was exhausted (4.5 MB). Once memory was exhausted, previously allocated segments were reused in the manner described in section 8.2.

This version of background recording proved useful for testing purposes. It was expected that an arbitrary segment size of 20 seconds would not be effective. However, the recording of silence proved even more deleterious to the utility of the background recordings. If the device was "left alone" for only a short period of time, any useful speech data would be overwritten with silence. Since there are often large periods of silence or background noise, if this audio data is not filtered, the result may often be a "Capture" list containing only silence.

In addition, navigation was negatively affected. When navigating from one speech segment to the next (e.g., using the *PREVIOUS/NEXT* slider), the large portion of silence made it difficult to determine the location in the recorded speech.

8.7 Version 2: Segmentation Based on Pauses

In version 2 of background recording, audio data is segmented based on pauses—periods of silence above a designated length. This is accomplished using the pause detection algorithm described in section 5.5.3. Each temporary buffer of recorded data returned by the Macintosh Sound Manager[35] is analyzed to determine which portions contain speech and which portions contain silence or background noise.

The segmentation process is as follows:

1. When recording begins, a new segment is started.
2. Average magnitude is calculated over 100 ms windows and passed to the pause detection algorithm.
3. Silence intervals are discarded until the first interval of speech is detected.
4. Speech data is copied into the current segment until a "significant" pause (see section 8.7.4) has been detected. Segments of less than a minimum required length (400 ms) are discarded.
5. Once a pause has been detected, a new segment is started.
6. Go to step 2.

This process continues until capturing of background speech is interrupted by the next user command. When recording is halted, the final buffers of sound data are analyzed and the current segment is terminated. The next time background recording begins, a new segment is started (the process returns to step 1).

[35] Ten second buffers are used.

Memory management is more complicated with variable length segments than with fixed length segments. Referring to the above segmentation process, no memory is allocated until speech is detected (step 4). A new segment, the size of the current interval of speech data (10 seconds or less) is allocated. Each time step 4 is reached, the segment size is increased by another 74180 bytes (10 seconds) until a "significant" pause is reached. If there isn't enough memory available to allocate or increase a segment, segments at the head of the list are deallocated until a contiguous block of the size needed can be allocated. Since segments are constantly being allocated, deallocated, and resized, relocatable memory (handles) is used. If non-relocatable memory (pointers) is allocated, the application heap may become fragmented. Using handles allows the heap to be compacted, creating contiguous blocks of memory space.

8.7.1 Defining Pauses

Maclay and Osgood identified four types of hesitation phenomena found in spontaneous speech—filled and unfilled pauses, repeats and false starts [Maclay 1959]. Filled pauses contain utterances such as "ah," "m," and "er," while unfilled pauses correspond to periods of silence. In terms of segmenting conversations, only unfilled pauses are considered. The location of unfilled pauses can be determined by analyzing the energy in the waveform over time, while the location of filled pauses is a more difficult problem.

Boomer and Dittman defined two types of unfilled pauses—**hesitation pauses** and **juncture pauses** [Boomer 1962]. Juncture pauses are defined as occurring at phonemic clause or syntactic boundaries; all other pauses are defined as hesitation pauses. Boomer and Dittman performed a study comparing the discriminability of hesitation and juncture pauses of lengths 100, 200, and 500 msec.[36] The results showed that 75% of the listeners could discriminate hesitation pauses of 200 msec or longer, while less than 50% could discriminate juncture pauses of 500 msec or less (only 25% of subjects could discriminate juncture pauses of 200 msec). Therefore, Boomer and Dittman defined discriminability thresholds: 200 msec for hesitation pauses, and 500-1000 msec for juncture pauses.

Boomer and Dittman cite an earlier study comparing actual pauses to those judged by human listeners. The study found that only those pauses of 1.0 seconds and longer could be accurately discriminated by the human judges [Boomer 1962]. A "lower limit" for pause duration was therefore defined as 1.0 seconds in correspondence with the discriminations made by the human listeners. However, a distinction was not made between those pauses at syntactic boundaries and other pauses (juncture versus hesitation).

Grammatical pauses, defined similarly to juncture pauses, are often referred to in the literature. Grammatical pauses are defined as "those which occur between clauses and are used for structural and semantic long-term planning in speech production" ([Reich 1980], p. 380). Pauses that are

[36]These pauses were inserted into a tape recording of a spontaneous discussion among four men.

within-clause boundaries are defined as **nongrammatical** and are used for "last-minute word selection" [Reich 1980]. While grammatical pauses are said to facilitate recall, ungrammatical pauses may interfere with the processing of speech. The results of a study performed by Reich showed that 78% of sentences with pauses at grammatical locations were recalled correctly, while correct recall was only 59% for sentences with pauses at ungrammatical locations. In addition, if a pause (330 msec) was placed in a nongrammatical location, subject's response time in recalling the sentence was increased by more than 2 seconds. This indicates that segment boundaries placed at nongrammatical or non-juncture positions could result in degraded speech perception.

Butterworth, Hine, and Brady distinguish between three types of unfilled pauses: grammatical, hesitation, and transition [Butterworth 1977]. Transitions are defined as between-speaker pauses used as verbal turn-taking cues.

8.7.2 Spontaneous vs. Read Speech

Minifie performed a study comparing durational characteristics of speech for various modes of speaking [Minifie 1974]. Several conditions of oral reading were compared against each other and against an "impromptu" or spontaneous speaking condition. The results showed that the "impromptu" speech had a "slower speaking rate, lower speaking time ratios, longer speech-interval durations, and longer silence-interval durations" than did the read speech ([Minifie 1974], p. 715).

In a comparison of read and spontaneous sentences, Soclof and Zue found that pauses in the spontaneous speech were an average of 2.5 times longer than pauses in the read speech [Soclof 1990]. However, this speech was produced in the context of a human-computer dialogue, not a human-human conversation.

Agnello summarized several studies regarding the lengths of pauses in spontaneous and read speech [Agnello 1974]. One study found that the majority of pauses in spontaneous, conversational speech exceeded 0.5 seconds and reported an average speech unit (periods of speech separated by at least 0.5 seconds of silence) of 2.3 seconds. However, the average speech unit was found to change as a function of the minimum pause length. For example, increasing the minimum pause length to 0.80 seconds caused the average speech unit to be increased to 3.52 seconds. Another study reported mean pause lengths for spontaneous speech ranging from 0.75 to 5.0 seconds. In studies of read speech summarized by Agnello, average pause lengths ranging from 0.40 to 0.50 seconds were reported.

8.7.3 Face-to-Face vs. Telephone Conversations

Brady studied the on-off speech patterns of 16 experimental telephone conversations [Brady 1968]. For each speaker, the lengths of alternating "talkspurts" and pauses were calculated.[37] The average talkspurt length was calculated to be between 0.902 and 1.311 seconds in length; average pause length was between 1.664 and 1.721 seconds.[38] Brady also calculated "mutual silence" durations (the length of time in which neither speaker a or b was talking) and "alternation silences" (following a mutual silence, the opposite speaker begins to talk). These two measures are probably the most applicable for segmenting background speech. Average mutual silence was found to be in the range of 0.425-0.495 seconds,[39] while average alternation silences ranged from 0.345-0.456 seconds.

Butterworth, Hine, and Brady examined conversational speech in three conditions: vision, non-vision, and telephone [Butterworth 1977]. In the vision condition, subjects participated in a face-to-face conversation, while in the non-vision condition, subjects were separated by a screen. Six pairs of conversations were studied; each pair consisted of one male and one female participant. The results showed significant differences between the telephone and vision conditions. The "vision" conversations consisted of more overall pausing, more within speaker pausing, and, more frequent and longer grammatical pauses. The mean length of grammatical pauses in the telephone conditions (0.76 seconds) differed significantly from the mean length in the vision conditions (1.3 seconds). This was also true of hesitation pauses (a mean of length 0.65 seconds for telephone, and 1.1 seconds for vision).

Silence is less tolerable during a telephone conversation than when conversing face-to-face. Some subjects in Butterworth's study reported that "telephone silences were less acceptable, and they were oppressed by this" ([Butterworth 1977] p. 93). Research findings, cited by Rochester, noted that placing an opaque screen between the conversants causes them to utter more filled pauses and shorter duration silent pauses than when speaking face-to-face [Rochester 1973]. Pauses are filled in order to maintain control over a conversation, as if to say, "I'm still in control—don't interrupt me" ([Goldman 1961] p. 19). This control is necessarily verbal when speaking over the telephone, while control can be maintained using visual cues during a face-to-face conversation.

8.7.4 "Significant" Pause Length

The intention of recording background speech is to "capture" spontaneous face-to-face conversations between two (or more) conversants. The goal is to separate the conversation into

---

[37] A pause had to be at least 200 msec.

[38] Talkspurt and pause length are calculated for each speaker individually. Pause length, for example, includes the time that one speaker is silent while the other speaker is talking.

[39] Average talkspurt, pause, mutual silence, and alternation silence durations were calculated based on three different thresholds for dinstinguishing speech from silence.

segments according to approximate syntactic boundaries or significant "junctures" in the speech. In this way, when users replay the recorded information, they will have the ability to move between logical chunks in the conversation and be able to locate the speech data of interest.

Based on Brady's research findings (the average mutual silence in the experimental telephone conversations ranged from 425 to 495 msec), a pause length of 500 msec was used to separate captured background speech into segments [Brady 1968]. After experimenting with this value, it seemed to be too short and had a tendency to break the speech up into rather small, insignificant chunks. Rather than facilitating the comprehension of the background speech, it seemed to have a deleterious effect. This seems to coincide with Reich's description of the negative perceptual affects of ungrammatically placed pauses [Reich 1980]. The problem is that Brady's findings are based on telephone rather than face-to-face conversations. Pause lengths in the 500 msec range were also cited in studies summarized by Agnello, but these studies were based on read rather than spontaneous speech [Agnello 1974].

Due to the poor results achieved when segmenting background speech based on a 500 msec pause length, this value was increased to 1000 msec. In support of a longer pause length, research cited by Agnello found that pauses in spontaneous conversational speech were at least 500 msec. Boomer and Dittman's study indicates that juncture pauses must be greater than 500 msec in order to be discriminable above a 50% accuracy level. Boomer and Dittman also cite a prior study in which pauses needed to be at least 1000 msec in order to be discriminated accurately by human listeners. Perhaps the strongest evidence for using a longer pause value for determining segments in spontaneous face-to-face speech is given by Butterworth. While the mean pause length calculated for the telephone conditions was 760 msec, this value increased by more than 1.5 times for the vision conditions (mean = 1300 msec).

Figure 8.2 shows a small portion of a conversation that has been segmented using a pause length of 1000 msec.[40] The captured speech is not always segmented at grammatical junctures; pause duration alone may not be enough for distinguishing between unfilled grammatical and ungrammatical pauses (see section 10.3.3).

[40]This data is taken from an actual session using the VoiceNotes application. The actual pause length between each segment may be less than 1000 msec due to possible overshoot in the pause detection algorithm.

Segment 1, Speaker A:	"I should get one of these for my computer."
Segment 2, Speaker B:	"A CD, would you really need it that much?"
Segment 3, Speaker A:	"Well, if I had any CD-ROMS I would."
Segment 4, Speaker A:	"Anyway, I have enough CD-ROMS that I could, I could..."
Segment 5, Speaker A:	"...seriously see using lots of stuff...I mean there's all the ETO, if we had the ETO disk..."
Segment 6, Speaker B:	"What comes on there?"
Segment 7, Speaker A:	"The central, that's like everything, all the developer tools, like all the..."

Figure 8.2: Portion of a "captured" conversation. Note that although the segments happen to fall at speaker boundaries, no attempt is being made to separate the speakers. Notice also that segments are not always at grammatical junctures (Segment 4 to Segment 5).

8.7.5 Segment Length

In segmenting the captured background speech, it may be desirable to define minimum and maximum segment lengths.

A minimum segment length can be used in one of two ways. Segments below a certain length can be discarded or continued. In the "continued" case, if the segment length is too small, then subsequent recorded data can be added onto the end of the current segment until the next "significant" pause is reached. The former method, discarding segments below a minimum length (400 ms), is used by VoiceNotes. One advantage is that the button "click" (that gets saved when recording is terminated), is discarded.

Although a minimum segment length has been defined, VoiceNotes does not use a maximum segment length. Segments are defined as being both preceded and followed by 1000 msec of silence. This seems a more accurate measurement than defining an arbitrary maximum length that could result in poor segmentation in some cases (e.g., a phrase longer than the maximum length will be broken at an arbitrary point). Although researchers have tried to estimate the average "talkspurt" length for an individual speaker in a conversation, the overall "talkspurt" length for two or more conversants is unclear. Brady determined an average "double talk" duration, but this accounts only for the duration in which the two conversants were speaking simultaneously, and does not account for back-to-back "talk-spurts" without an intervening pause (i.e., speaker A talkspurt, speaker B talkspurt, speaker A talkspurt, etc.). In addition, Agnello cites findings that the average speech unit length changes as the definition of the minimum pause length changes [Agnello 1974].

8.7.6 Filtering Voice Commands

In the current implementation of VoiceNotes, background recording takes place when there is no other user activity. Therefore, in general, the user's voice commands do not get captured when recording background speech. However, if a voice command interrupts background recording (e.g., "*PLAY*"), this speech data gets recorded and entered into the last speech segment on the "Capture" list. It is questionable whether this speech should be filtered from the other background speech and discarded, or considered part of the background speech. Currently, VoiceNotes does not filter these voice commands. The inclusion of this speech data may add contextual information (i.e., the reason recording was terminated).

If the goal is to filter such a voice command, one approach might be to throw away the last segment of background speech. However, the voice command may be at the end of a segment containing other ambient speech (rather than in a segment by itself). Therefore, the last segment cannot be discarded without risk of losing valuable speech data. While there are many other possible approaches for filtering the voice command, the effort seems unnecessary. For example, if stereo recording was available, "background speech" could be recorded on one channel, while the other channel is being used for recording intentional voice notes. In this case, the "background speech" will include voice commands, intentional voice notes, and captured conversations (i.e., a history of speech data).

8.8 Saving Segments

The previous sections have discussed the capture and segmentation of background speech. The "Capture" list ("digital-tape loop") serves as a temporary place holder for these speech segments. Users may wish to extract speech segments from the capture list (unintentional voice notes) and add them into one of their permanent lists of voice notes (intentional voice notes). This is accomplished using the "*MOVE*" command. "*MOVE*" provides a general mechanism for transferring voice notes between lists. Therefore, rather than assigning a new command (e.g., save) to this function, the "*MOVE*" command is used. Users are given the ability to move segments from the capture list to any other list of voice notes.

8.9 Use of Two Microphones

Different characteristics are needed for recording intentional voice notes and recognizing voice commands than for capturing background speech. Hand-held prototype 1 uses only a single microphone. While the microphone is omni-directional, since it has been mounted inside the hand-held unit, it acts more like a directional microphone. Therefore, it is not suitable for capturing background speech.

In a future hand-held prototype, two microphones will be used:

1. Omni-directional for recording background speech

2. Directional (possibly noise-canceling) for recording intentional voice notes, and recognizing voice commands.

If only one recording channel is available, software control can be used for switching recording input from one microphone to the other. However, if two channels are available, background speech could be recorded at all times (not only during periods of user inactivity). In addition, if speech could be recorded by both microphones simultaneously, this data could be used to screen out insertion errors. This could be accomplished by comparing the energy levels recorded by each microphone. During periods in which the energy of the speech recorded using the omni-directional microphone is greater than the energy recorded using the directional microphone, any speech commands reported by the recognizer could be ignored (i.e., assumed to be insertion errors).

9. Software Design

9.1 Overview

The VoiceNotes application communicates with a separate recognition server process. The Recognition Server manages the speech recognition hardware and asynchronously sends events to the VoiceNotes application when words are recognized or training utterances are collected (Figures 9.1). The VoiceNotes process also sends and receives data serially from the hand-held unit, and uses the Macintosh Sound Manager in order to digitally record[41] and playback sound.

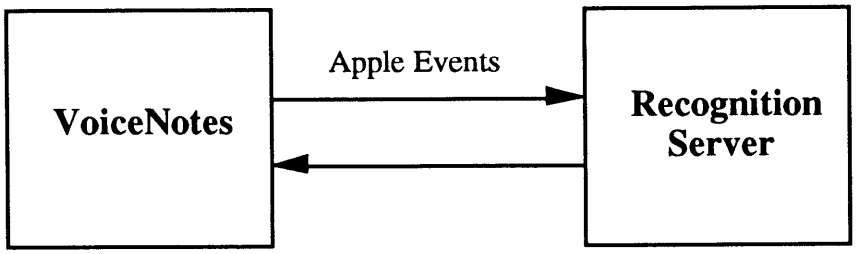


Figure 9.1: VoiceNotes and the Recognition Server are separate processes that communicate using Apple Events.

9.2 Recognition Server

The Recognition Server is a process that runs in the background sending and receiving data from the VoiceNotes application using apple events (Figures 9.2-9.4). The primary reason for the server is to provide asynchronous I/O between the VoiceNotes application and the Voice Navigator, since the Voice Navigator library only provides synchronous communication with the device. Utterances spoken by the user are collected in a queue and obtained using a synchronous Voice Navigator library call, SDRecognize(). When SDRecognize() is called, it does not return until the next utterance is collected. The only way to perform other actions while waiting for input is to use a key-watch function. The key-watch function is called periodically while waiting for input from the user. Inside a key-watch function an application can check for events such as mouse and keyboard presses. The key-watch function is designed to allow the application to quit recognition upon receiving a given user input event. Given this structure, development of a highly interactive application that communicates with many devices would be severely limited. The Recognition Server allows recognition events to be handled the same as mouse and keyboard events.

[41] When using a Macintosh that does not have built-in sound input (e.g., MacIIfx, MacIIci), the MacRecorder hardware and driver are used.

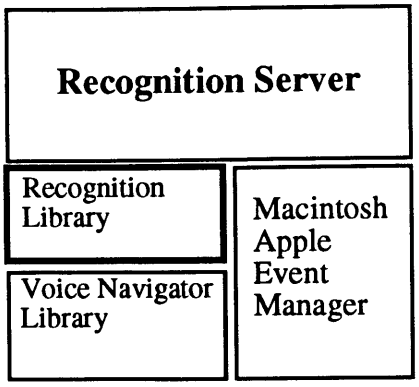


Figure 9.2: Libraries used by the Recognition Server.

VoiceNotes receives recognition apple events (Figure 9.3) on its main event loop. Once a connection between VoiceNotes and the server is established and recognition is started, the server sits in a loop waiting for spoken input. Each time a word is recognized, the server sends an event to VoiceNotes (the client) containing the word number (or a rejection error). The Recognition Server uses a key-watch function to check for pause or quit recognition events sent by the client. The server can be run on either the local or a remote machine. Currently, VoiceNotes and the Recognition Server are run on the same machine.

Apple Event ID	Usage
TRAIN_RESULT_EVENT	notifies the client that a word was successfully or unsuccessfully trained (in response to ASYNC_TRAIN_EVENT)
RECOGNITION_EVENT	notifies the client that a word was recognized and reports the word number or a rejection error

Figure 9.3: List of events sent by the Recognition Server to the client, VoiceNotes.

9.2.1 Recognition Library

The Recognition Server uses a recognition library to perform all recognition functions. The recognition library provides a high-level functional interface for speech recognition. In this case it has been implemented for the Voice Navigator, but the same functional interface is general enough to be used with any speaker dependent isolated word recognizer.[42] The purpose of the library is to shield the client from the low-level details of communicating with a given recognizer and therefore allow easier development of speech interfaces (Figure 9.2 lists the library function associated with each recognition event). The recognition library has been implemented on top of a standard "C" library provided for the Voice Navigator.

[42]This functional interface is used with a Texas Instruments speech recognizer as well.

Apple Event ID	Library Function	Usage
REGISTER_EVENT	s_set_up_sound	sets up a connection with the server and initializes the recognizer
USER_EVENT	s_user	passes user name to server
FILENAME_EVENT	s_filename	passes application name to server
DISK_TO_RECOGNIZER_EVENT	s_disk_to_recognizer	loads the application's language file and the user's voice file
RECOGNIZER_TO_DISK_EVENT	s_recognizer_to_disk	saves language and voice file for the specified application
RECOGNIZE_ONCE_EVENT	s_recognize_once	starts recognition, reports the next word recognized, and quits recognition
START_RECOGNITION_EVENT	s_start_recognition	starts recognition
QUIT_RECOGNITION_EVENT	s_quit_recognition	quits recognition
PAUSE_RECOGNITION_EVENT	s_pause_recognition	pauses recognition
CONTINUE_RECOGNITION_EVENT	s_continue_recognition	continues recognition
START_TRAINING_EVENT	s_start_training	initializes training
TRAIN_EVENT or ASYNC_TRAIN_EVENT	s_train	collects a template for the specified word number
QUIT_TRAINING_EVENT	s_quit_training	quits training
DELETE_WORD_EVENT	s_delete_word	deletes the template associated with the specified word number
SUBSET_EVENT	s_subset	subsets vocabulary as specified by client
ACTIVATE_WORD_EVENT	s_activate	activates or deactivates a single word number in the vocabulary
THRESHOLD_EVENT	s_set_threshold	sets recognition rejection threshold
DONE_EVENT	s_unlink	disconnects from the server and recognizer

Figure 9.4: List of events sent by the client to the Recognition Server. Each event is associated with a particular recognition library function that gets called when the event is received by the server.

9.3 VoiceNotes

The VoiceNotes application receives input from three sources: the sound input device, the speech recognizer, and the hand-held device. VoiceNotes always checks for user input (speech or buttons) while idle, and while playing or recording speech. This creates a highly interactive application; the user can interrupt the current action at any time. Whenever voice or button input is received, a single dispatch function is called which then calls the appropriate action handler. Whenever the system is idle for a certain period of time, the application begins to capture background speech. Background recording is terminated whenever the next user input command is received.

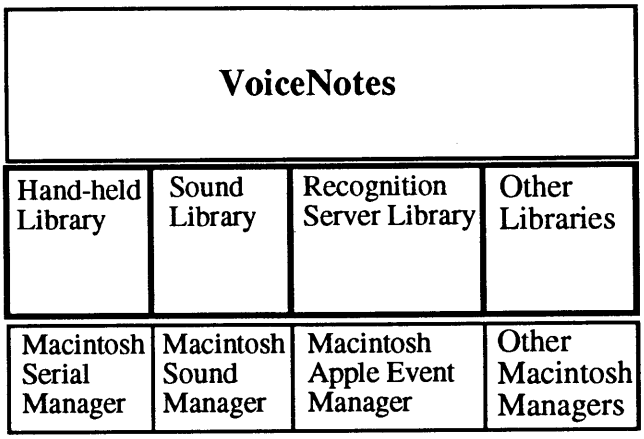


Figure 9.5: Libraries used by the VoiceNotes application. Other libraries include: a voice notes database library for data management, a linked list library, and a library for capturing background speech. Other Macintosh managers used are the File Manager and the Memory Manager.

9.3.1 Recognition Server Library

VoiceNotes uses a recognition server library in order to send apple events to the Recognition Server. The recognition server library provides the same functional interface as the recognition library. For example, when VoiceNotes calls the recognition server library function s_start_recognition(), this causes a START_RECOGNITION event to be sent to the Recognition Server which then calls the actual recognition library function, s_start_recognition(). This provides an interface programmatically similar to remote procedure calls (RPC), but implemented in a fully asynchronous manner. A client can choose to link in either the recognition server library for use with the Recognition Server or the recognition library for a direct interface to the recognition functions.

9.3.2 Data Management

An ASCII voice notes data file is created for each user when the VoiceNotes application is used for the first time (Figure 9.6). The system keeps track of each list name and its associated notes

by storing the names of the sound files in the user's data file. For each list name, the associated vocabulary word number is also stored. The data is read from the file into an in-memory linked list structure at the start up of the application and written back into the file upon quitting. Only the names of the sound files are stored, *not* the paths. This allows more flexible file management as described in the next section.

```
notepad 23              #the notepad list
name0note1
name0note2
name0note3
*
name1 51                #list name sound file and word number
name1note1              #voice note sound filename
name1note2
*
name3 52
name3note1
name3note2
name3note3
```

Figure 9.6: An excerpt from a VoiceNotes data file. Each list name is stored with its associated word number in the VoiceNotes vocabulary (e.g., name1 is associated with word number 51).

9.3.3 File Management

All VoiceNotes data is stored in a "Speech" folder (Figure 9.7). This folder can be stored under the default volume (i.e., Machine Name:Speech), or can be mounted from a remote server machine. When VoiceNotes starts up, it checks for a Speech folder on the local machine; if none exists, it checks for a mounted Speech volume. This path (either Machine Name:Speech or Speech) is then used for accessing all data files. The Speech folder contains a "master" folder, a "prompts" folder, and a folder for each VoiceNotes user (e.g., "lisa," "eric"). The master folder contains the VoiceNotes language file used by the Voice Navigator, and a default user configuration file. The prompts folder contains three folders: "female," "male," and "synthesized" containing all pre-recorded VoiceNotes prompts. Each user folder contains a voice file (containing stored templates for recognition), voice notes database, optional configuration file, and all list names and notes sound files.

This file management structure is somewhat unconventional in its use of separate folders for multiple users since the Macintosh is a single user machine. However, in our environment, each user does not have his/her own Macintosh. This set-up also provides a simple mechanism for demonstrating the application. In addition, the ultimate goal is to store the sound data directly on the hand-held device.

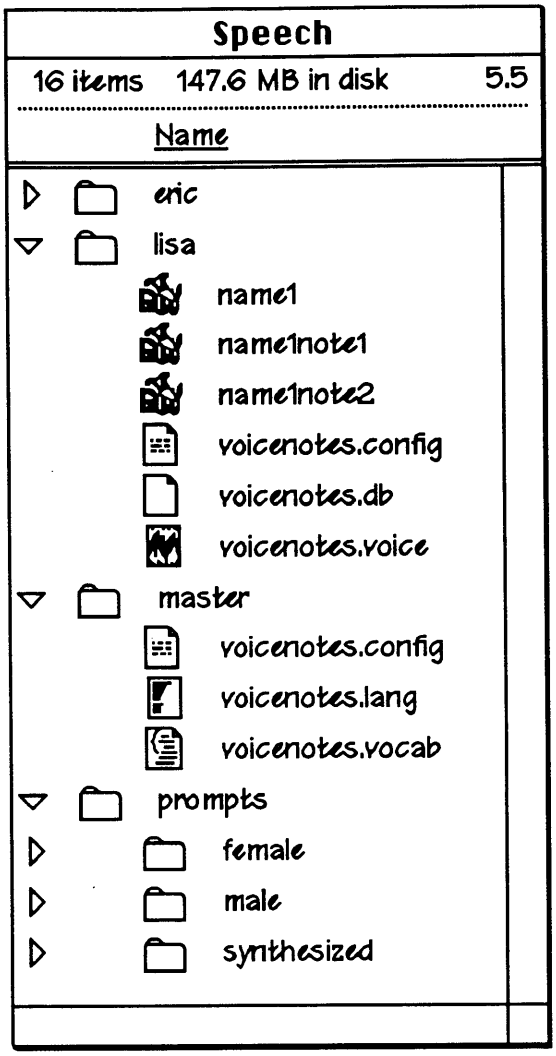


Figure 9.7: VoiceNotes file management structure.

9.3.4 Sound Library

The sound library was written on top of the Macintosh Sound Manager to provide a high-level functional interface for recording and playing back sounds. The sound library supports the following: playing and recording at 5, 7, 11, and 22 kHz, playing and recording into memory or AIFF sound files, pause detection, silence compression,[43] and time-compression during playback. The sound library can be used with either the MacRecorder sound driver or the Macintosh built-in sound driver.

The sound library does not currently provide asynchronous playing and recording of sounds. While a sound is playing or recording, the client must sit in a loop calling a sound library work

[43]Silence compression is a feature provided by the Macintosh built-in sound driver but is not supported by the MacRecorder driver.

procedure until completion has been indicated. In the future, asynchronous playing and recording capabilities will be provided.

9.3.5 Hand-held Library

The hand-held library provides an interface for sending and receiving data from the hand-held device. There are four main functions: one for opening a connection to the device, one for polling the device, one for turning the LED on and off, and one for closing the connection. A connection is established with the device through the modem port of the Macintosh. A program on the hand-held is activated by sending the command "MAIN" to the device. Once a connection has been established, VoiceNotes polls the hand-held for input using the hand-held library's polling function.

It is necessary to "filter" the input received from the hand-held device (e.g., due to switch bouncing). The hand-held polling function filters information received from the device by using a finite state machine where only the previous state is stored. For example, if the user presses the play+record buttons on the hand-held device, it is possible to receive two record messages instead of just one. This is filtered by ignoring any new state that is the same as the previous one. Another problem occurs when the user slides the fast forward/rewind button into the fast forward position and then releases it—the hand-held sometimes sends both a fast forward and a rewind message. Under these conditions the polling function filters out the extraneous rewind message. The potentiometer on the hand-held (used for speed control) can oscillate between two values, causing the hand-held device to send a continuous stream of data. This data is filtered by comparing the current value of the potentiometer with the previous one. If the new value is within a certain range of the previous value, it is ignored.

Since it is possible for the hardware to become out of synchronization with the software, VoiceNotes must perform some additional "application-specific" filtering of input returned by the hand-held library's polling function. For example, the position of the list selection button often differs from the user's actual list position. The user may be in the names list, when the list selector is in the notes position. Therefore, if the selector is moved into the names position when the names list is already current, the action is ignored. In addition, the *STOP* button may remain depressed even after playback has ended. In these cases, when the *STOP* button is pressed, the action is ignored.

9.4 Servers vs. Libraries

It may be desirable for the sound and hand-held libraries to be converted to servers based on apple events (similar to the Recognition Server). This allows asynchronous communication between the VoiceNotes application and the sound, serial, and voice recognition devices. All input events (e.g., button input, speech input, play or record completion notification) could then be processed by the application's main event loop.

While the use of separate server processes is a desirable goal, it is difficult to manage multiple processes in the Macintosh environment. Each process is responsible for managing its own use of processor time. For example, when VoiceNotes is the foreground application it must make sure to call the Macintosh Toolbox routine WaitNextEvent() often enough to give background processes (such as the Recognition Server) sufficient processing time.[44] In turn, the background process must call WaitNextEvent() in order to relinquish the processor to the foreground process. This can be difficult if a process must synchronously call a remote procedure. For example, when calling the synchronous function SDRecognize(), the Recognition Server must call WaitNextEvent() from inside the key-watch function in order to give up processing time while waiting for the next user utterance to be reported.

9.5 VoiceTrainer

Before running the VoiceNotes application for the first time, the user must train the VoiceNotes command vocabulary. The first time that the user runs the VoiceTrainer, a folder and voice file are automatically created for the user. The VoiceTrainer application links in the recognition library directly, and does not use the Recognition Server.

The VoiceTrainer provides the flexibility to train the entire VoiceNotes vocabulary or individual words. When training the entire vocabulary, the interface is "voice-only"—the user is prompted with voice output and responds with voice input. However, when training individual words, the user must enter the word numbers using the keyboard into a console window.

Currently the VoiceTrainer is a separate application from VoiceNotes. In the future, it may be desirable to integrate training into the VoiceNotes application, and allow the user to train individual command words.

[44] If WaitNextEvent() is not called by the current process, other applications will not receive processing time.

10. Conclusions

This thesis has explored a voice and button interface to a hand-held computer through the development of the VoiceNotes application. Many lessons were learned and questions raised about an interface to a hand-held computer that has no visual display, and uses audio as the only data type.

10.1 Differentiating VoiceNotes from other Voice-I/O Systems

- Random access. VoiceNotes addresses one of the most significant problems with current personal recording technology—the lack of random access to recorded audio data. Voice recognition is an effective means for providing users with the ability to categorize and organize segments of audio data.

- Spontaneity. Unlike the use of voice at a desktop computer (e.g., voice annotation of a document), VoiceNotes supports the capture of spontaneous thoughts, ideas, and conversations. This necessitates the use of a portable device, since such "spontaneous" thoughts and interactions will occur in a variety of situations and locations.

- User-authored speech data. VoiceNotes is unique from voice-I/O hypermedia systems in that it allows users to author, navigate, and organize their own speech data, not data that has been edited and structured by a system designer.

10.2 Lessons Learned and Questions Raised

10.2.1 Voice Interfaces

- Terminology. The terminology used by a non-visual application is critical for providing the user with a "mental model" of the application's structure. Since there is no visual representation of the list structure in VoiceNotes, users must rely on the wording of commands and feedback to guide them through the use of the application.

  There are many visual metaphors in graphical user interfaces (e.g., folders, clipboard), but there are no auditory equivalents. For example, a collection of papers can be placed in a "folder" but what is the term for a collection of audio segments? Should speech interfaces borrow terminology used by visual interfaces or create new metaphors? Although work has been done in the area of auditory icons [Gaver 1989], audio terminology has not yet been established.

- Modeless interface. Since each user's mental model of the system is likely to differ from that of the designer, moded designs should be avoided. The first version of the VoiceNotes application was moded and error prone since only certain actions were valid in each mode. Speech commands cannot be "grayed-out" the way items on a visual menu can be. While the vocabulary can be subset, the user may still attempt to issue commands outside the current subset, leading to rejection errors. For these reasons, VoiceNotes adopts a modeless interface, and attempts to keep all commands active at all times.

- Voice direct manipulation. Direct manipulation in visual interfaces generally involves the user's engagement with static objects. In contrast, voice direct manipulation involves the manipulation of time-driven objects ("moving targets"). Both user reaction time and system response time must be considered when using voice direct manipulation or target commands. VoiceNotes addresses this issue by providing a "target window" that extends past the current item.

- Interruption of spoken output. VoiceNotes allows the user to interrupt the spoken output of the system with spoken input or button input. While the importance of interruption is often stressed [Arons 1991] [Brennan 1992], what factors influence the user's likelihood and ability to interrupt using voice input versus button input? For example, users may find it awkward to "talk over" the system output. In addition, users may have a tendency to wait for a pause in the spoken output before interrupting with a voice command. For these reasons, will users find it easier to interrupt the spoken output using button input rather than voice input?

- Simple yet powerful. The goal of VoiceNotes is to provide a simple, yet powerful interface. Therefore, an attempt has been made to support only the most essential functionality, while eliminating minor features that might dilute the power of the interface.

  However, is the interface powerful enough? VoiceNotes currently provides only a one-level list hierarchy. As people become more accustomed to using voice-I/O interfaces, they may desire the ability to perform more complex operations.

- Combining speech and buttons. The ultimate interface to a computer is often described as "conversational"—the user converses with the computer as with another human. Human conversations, however, not only employ speech, but use many additional methods of communication. Therefore, the "ultimate" conversational interface may not employ speech alone. VoiceNotes explores the combined use of speech and button input for interacting with a hand-held computer. This results in a more powerful and interactive interface than can be provided with either input modality in isolation.

10.2.2 Navigation

- Placement at startup. Startup conditions are critical for a non-visual interface. If users cannot identify their position on start up of the application, they will be "lost" from this point forward.

Therefore, it is important to provide a navigational "home-base." The notepad list serves as this navigational starting point for the VoiceNotes application. This will be especially important as the navigational space becomes larger and more complex.

- Navigational cues. Navigational feedback should give the user a sense of movement or change. In one version of VoiceNotes, when moving between lists, only the new list name was echoed. There was no indication that the user's position in the list structure had changed. The speech feedback was modified to more clearly indicate this change in position (e.g., "Moving into...*Notepad*").

  However, how can different navigational spaces be represented? For example, should the background audio quality change as one moves between lists? Non-speech audio may be better than speech output for providing a "sense" of change or movement when navigating. Perhaps a combination of speech and non-speech feedback can result in more powerful navigational cues.

- Navigational anchors. It is important to represent the limits of the navigational space. For VoiceNotes, the limits are the beginning and end of each list. If users attempt to navigate beyond these limits, they are prompted "beginning of list" or "end of list" and their current position is maintained at the first or last item in the list. The beginning and end of a list therefore act as "anchors" for navigational control instead of "drop off" points.

  While VoiceNotes represents navigational limits, it does not currently convey navigational affordances. For example, on entering a list, perhaps the user should be informed of the number of notes it contains. In addition, how can the "global" affordances (i.e., the total span of information) of a speech database be represented to the user as the information space grows larger and more complex?

- Position in "navigational space." The system should maintain the user's current position in the navigational space (i.e., current list item) in order for the user to maintain navigational control. In one iteration of VoiceNotes, each time the user issued the *PLAY* command, the system responded by playing from the first item on the list. Given this design, it was difficult for the user to start and stop playing while maintaining their current position in the list. The final design of VoiceNotes maintains the user's current position in the list structure; when the *PLAY* command is received, the system responds by playing from the current item in the list instead of the first item.

10.2.3 Feedback

- Distinct feedback. Feedback should be brief, yet distinct. In an attempt to be brief, the feedback for several commands was too terse to clearly indicate that the system had correctly recognized the user's request. The feedback was updated to be more informative while maintaining brevity.

- Feedback levels. Provide feedback based on the user's experience level with the system. Since speech is slow and serial, it is recommended that feedback be as brief as possible [Arons 1991]. However, more explicit feedback is required for novice users. VoiceNotes provides different levels of feedback for different levels of user experience by allowing a feedback level to be set in the user's configuration file.

10.2.4 Speech Recognition

- Voice-only training. The VoiceTrainer uses voice to prompt the user to speak each word in the VoiceNotes vocabulary. Does voice, as opposed to text prompting of vocabulary words during training, lead to better recognition accuracy? Do users mimic the pronunciation of the commands used by the speaker of the voice prompt? If so, it may be possible to help direct users to speak in the clear and declarative manner that is necessary to achieve good recognition accuracy.

- Isolated vs. continuous speech. One problem encountered in the development of VoiceNotes, was the recognizer's inability to distinguish between isolated "command" speech directed at the recognizer and continuous natural speech used, for example, when recording a voice note. Currently, VoiceNotes relies on the speaker's distance from the microphone in order to distinguish when to record speech versus when to recognize speech. Other systems require the user to address the system by name, or "push to talk" each time a command is issued. Many different methods have been used in an attempt to deal with this issue, but an optimal solution has not yet been found.

- New word detection. Currently there is an explicit process for adding new words (list names) to the VoiceNotes vocabulary. It is desirable, however, for the recognizer to be able to determine when a new word (i.e., a word outside the currently trained vocabulary) is spoken by the user. For example, in response to the system prompt, "Move to what list?" the user must supply an already existing list name. It would be convenient if the system could determine automatically if an old or new list name is spoken by the user.

  A related issue is the ability to add new words to the recognition vocabulary "on-the-fly." It is desirable to use a speech recognizer that supports a fixed vocabulary of command words while allowing users to add new words of their selection in real time. Currently, recognizers support either speaker independence or dependence, not both. From an application development standpoint, a hybrid system would be desirable. For example, VoiceNotes could use speaker independent recognition for commands words, while using speaker dependent recognition for list names.

10.2.5 Time-Compression

- Interactive control. Time-compression of speech is essential for systems in which stored speech is a primary data type. VoiceNotes provides the user with the ability to change the speed of

playback interactively while listening to the spoken output. In addition, VoiceNotes also uses time-compression in order to provide system feedback in a more time-efficient manner. Comprehension of time-compressed speech increases with practice and users tend to adapt quickly [Voor 1965].

10.3 Future Plans

10.3.1 VoiceNotes Interface

- Key word spotting. Word spotting[45] is an interesting alternative to complete speech-to-text transcription. For example, word spotting could be used by VoiceNotes to allow the user to search for key words and phrases across all the lists of VoiceNotes. In addition, "virtual" lists could be created this way; the user could play a list of voice notes containing a particular key word. Word spotting could allow users to maintain a larger number of voice notes by providing an additional mechanism for accessing the recorded speech data.

  Key word spotting is becoming a viable speech recognition technology. A speaker dependent key word spotter developed by Xerox PARC is used in conjunction with an audio editor [Wilcox 1991][Wilcox 1992]. By using speaker dependent recognition, an unlimited number of key words can be matched. A speaker dependent key word spotter, such as this one, would be applicable for use with VoiceNotes since the a majority of the speech data has been recorded by a single user.

- Non-speech audio feedback. Just as the combination of speech and button input provides the user with a rich set of interactions, the combination of speech and non-speech audio output could also provide a powerful combination for user feedback. VoiceNotes could incorporate non-speech audio cues into its current feedback levels, allowing non-speech audio to be used in place of or in combination with the speech feedback. Non-speech audio could provide enhanced feedback for novice users, when used in addition to speech output, while providing more efficient feedback for experts, when used in place of speech output. In addition, non-speech audio may be more effective than speech for providing navigational cues.

10.3.2 Time-Compression

- Scanning. While VoiceNotes allows the user to control the speed of playback, there is still the question of how time-compression can be used for scanning or skimming speech data as one can scan text. This is especially important for reviewing larger quantities of speech (e.g., captured background speech). How can time-compression be used to allow the user to scan forwards or backwards through speech at increased speeds, in order to listen for key words and phrases? This is a topic that needs further consideration.

[45] Word spotting is the process of locating a key word or phrase in a stream of fluent speech [Wilcox 1992].

- User perception of speed. Another research question relates to the user's perception of speed. What is the just noticeable difference for the speed of playback (i.e., how much change in speed is necessary for the user to perceive a difference), and how is this affected as the speed increases? While fine-grained speed control is desirable, the granularity of control should be designed to match the user's perceptual capabilities. In addition, the type of control, linear or non-linear, should match the user's perception as well.

- Speed and movement. Should speed control be combined with movement control (e.g., next/previous), or separate? A combined speed and movement control might provide similar scanning capabilities for audio that the "jog" and "shuttle" functions provide for video.

10.3.3 Segmentation

- Identifying grammatical pauses: VoiceNotes captures and segments background speech based on pause duration. The goal is to identify grammatical unfilled pauses, as opposed to ungrammatical ones, in order to segment speech at major syntactic boundaries. However, "unfilled pauses [grammatical versus ungrammatical] cannot be reliably thus separated based on silence duration alone" [O'Shaughnessy 1992]. Grammatical pauses could be more reliably distinguished from ungrammatical ones by examining not only the duration of pauses, but the fundamental frequency just prior to the pause as well [O'Shaughnessy 1992].

10.3.4 Testing

- User testing. User testing is useful for evaluating the design of the VoiceNotes interface. This can be accomplished by observing the user's performance of the specific tasks for creating, navigating, and managing VoiceNotes. This type of evaluation generally takes place in a controlled environment (i.e., lab) where users are videotaped while completing an assigned set of tasks. Several types of data can be collected: (1) informal observations made by the experimenter regarding aspects of the interface that caused user's confusion; (2) user ratings of the ease of performing each of the given tasks; (3) empirical performance data such as task completion time and the number of errors made; (4) the user's speculation about features of the interface that they think might be most or least useful, and suggested improvements.

- User studies. In addition to user testing, it is important to evaluate the user's actual experience using the VoiceNotes application and the hand-held computer outside a controlled laboratory environment, over a longer period of time. In order to learn more about the user's interaction with the hand-held computer, it will be necessary to build an "untethered" hand-held prototype that can be given to several users for evaluation purposes. The type of information that can be collected from such an evaluation is very different from the data that can be obtained from a user test.

  The following are some questions that could be considered through a user study in which several people are given hand-held prototypes to use over a given period of time:

Will users find VoiceNotes/the hand-held computer useful?

Where will users most often use VoiceNotes/the hand-held computer (e.g., car, office, home)?

In what contexts will users find VoiceNotes/the hand-held computer most useful (e.g., meetings, lunches, by oneself)?

How many lists of voice notes will users create on the average?

How many voice notes will users store in each list?

How long will the average voice note be and how much speech will users need to store?

How many lists/notes can be created before the lists become unmanageable given the current interface design?

Is a one-level hierarchy enough for managing voice notes or will users want the ability to create more levels?

How useful will users find the capturing of background speech?

How often will users save captured background speech?

What other applications will users want for the hand-held computer?

10.3.5 Hand-held Prototypes

- Untethered. Future hand-held prototypes will not have to be "tethered" to a back-end processor. Truly portable hand-held prototypes will be developed that use wireless technology to communicate with a back-end processor and/or have a built-in processor and memory for stand-alone use.

- Hand-held vs. hands-free. The focus has been on creating a voice-controlled "hand-held" computer. However, this does not take advantage of the ability for voice to provide hands-free interaction. Therefore, a future "hand-held" prototype might not require any hands to use; perhaps a wireless microphone, clipped onto the user's shirt, sends data to a hand-held processing unit contained in a belt pack (Figure 10.1), or a wrist-worn device (Figure 10.2).

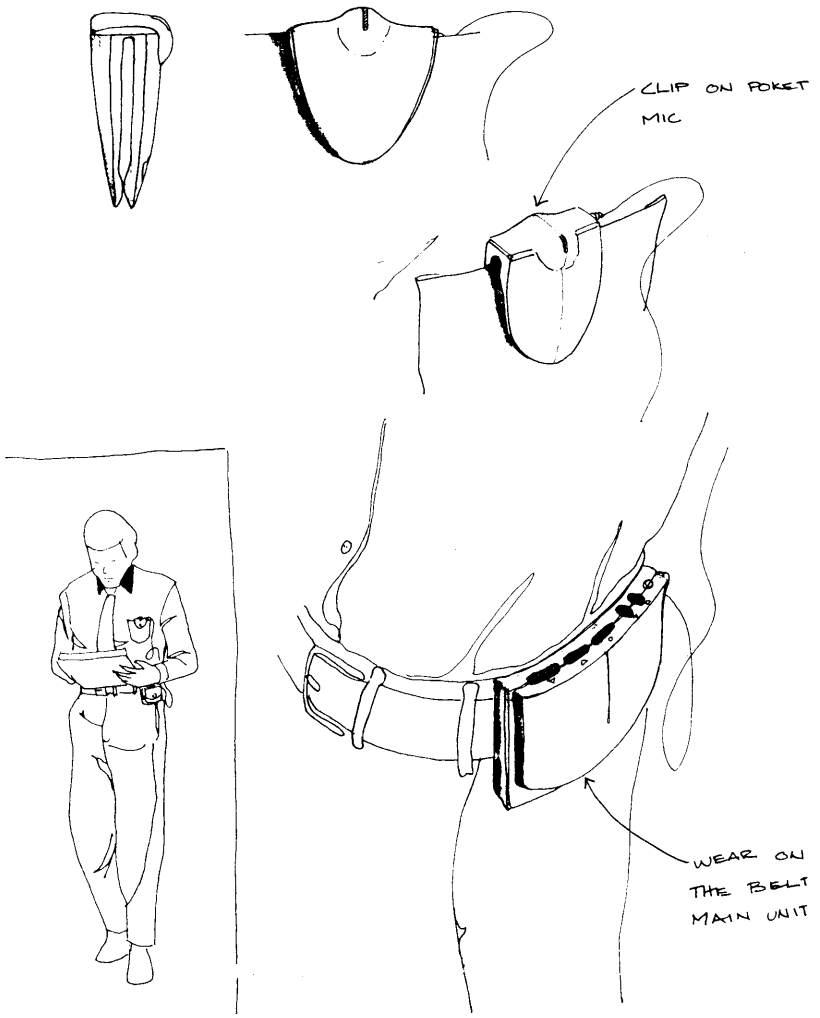


Figure 10.1: Conceptual drawing of a future "hand-held" prototype. (Drawing ©Apple Computer, Inc. 1992)

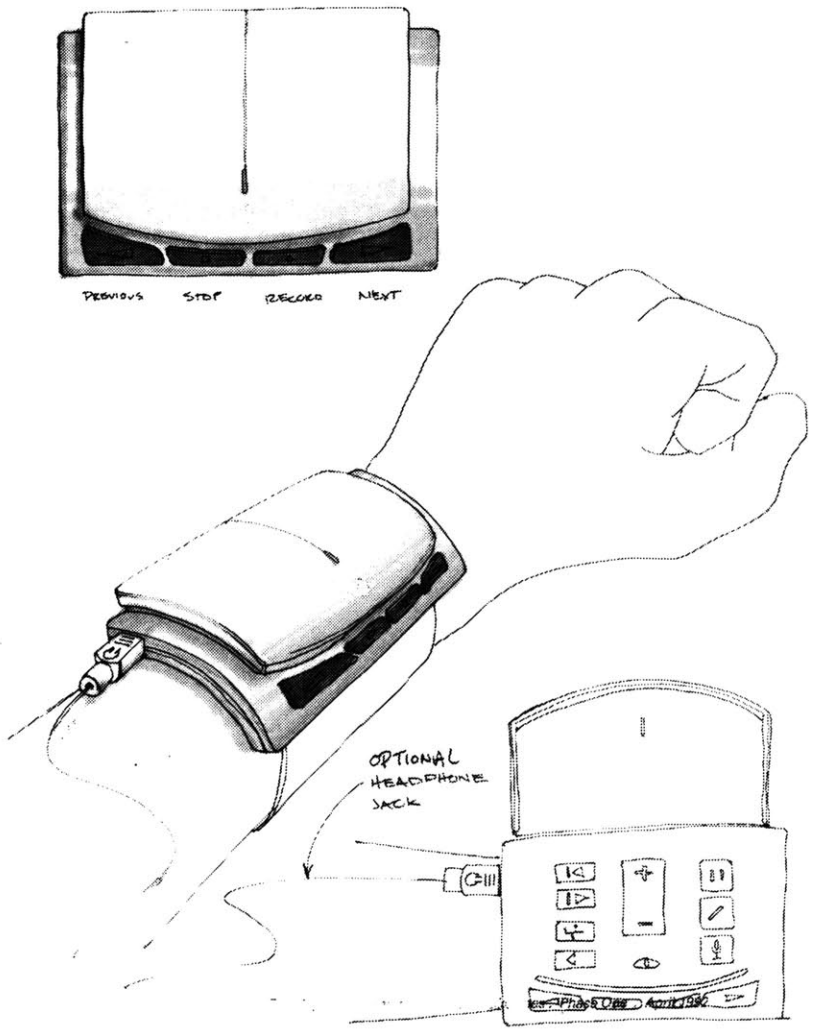


Figure 10.2: Conceptual drawing of a future "hand-held" prototype. (Drawing ©Apple Computer, Inc. 1992)

10.3.6 Other Hand-held Applications

This thesis has focused on information that is directly recorded and retrieved using a hand-held device. Recorded information can come from additional sources, such as: (1) data transferred from one's desktop machine; (2) data transferred from another user's hand-held; (3) data automatically transferred from an information providing service (e.g., news, radio).

Given a variety of information sources, new uses of a hand-held device can be envisioned:

- Voice mail. Users could transfer voice mail messages, received by their office voice mail system, to their hand-held computer. These messages can then be reviewed at the user's convenience (e.g., in the car on the way home from work). In addition, users could record voice mail messages on their hand-held computer. These messages could be delivered as soon as the hand-held device is brought within communicating distance of the user's desktop machine.

- Shared voice lists. This thesis has focused on isolated use of the VoiceNotes application by a single user, but cooperative uses can also be envisioned. Two users might share a list of voice notes, for example, regarding a paper that they are writing. These notes could be transferred back and forth between each of the user's hand-held devices. One user might also record a list of voice notes for another. For example, driving directions are given as an example of one of the possible uses of the VoiceNotes application. However, directions are generally recorded by one person for another. Given the ability of one hand-held device to communicate with another, one user could record a list of voice notes for another.

Finally, use of a hand-held device should also be integrated with use of a desktop machine, such that data recorded on a hand-held device can be easily transferred, reviewed, and organized on one's desktop machine.

Acknowledgments

Thanks to the following people for their support and contributions to this thesis:

Chris Schmandt, my advisor, for his insight and ideas about speech interfaces, for interface design input, and for giving me the opportunity to work on this project.

Eric Hulteen, sponsor and reader, for devoting his time to this project, for interface design input, for teaching me about moded and modeless interfaces, for graphic design contributions, and for constantly advocating the user.

Victor Zue, for taking the time to be a reader, and for providing insight regarding the speech recognition problems of new word detection, and key word spotting.

Barry Arons, for our brainstorming sessions of possible thesis topics, for coding advice, for coding contributions (pause detection, time-compression), for interface design advice, for valuable comments on drafts of the thesis, and for taking the photographs in this document.

Andrew Kass for his work on the sound library, and for figuring out the details of the Macintosh Sound Manager.

Derek Atkins for developing the hardware for prototype 1 (hand-held unit and black-box), and writing the code that runs on the hand-held.

Nicholas Chan for his work on the hand-held library and for the many hours spent testing the hand-held hardware.

Tom Bentley and Lawrence Lam of Apple Computer, Inc. for providing the concept renderings of hand-held devices included in this document. The drawings were done by Mike Nuttall and Doug Satzger of IDEO Product Development.

Corine Bickley for her assistance in outlining this document, for encouragement, and for her excitement about VoiceNotes.

Marlene Stifelman for carefully proof-reading this document and providing valuable comments.

The Human Interface Group/ATG at Apple Computer, Inc. for supporting this project.

References

[Apple 1991] Memory management. *Inside Macintosh, Volume VI*, chapter 28, pages 3-45. Apple Computer, Inc., 1991.

[Ades 1986] S. Ades and D. Swinehart. Voice annotation and editing in a workstation environment. *In Proceedings of* AVIOS '86, pages 13-28. American Voice I/O Society, 1986.

[Agnello 1974] J. G. Agnello. Review of the literature on studies of pauses. In S. Duker, editor, *Time-Compressed Speech*, pages 566-572. Scarecrow, 1974.

[Arons 1991] B. Arons. Hyperspeech: navigating in speech-only hypermedia. *In Proceedings of* Hypertext '91, pages 133-146. ACM, 1991.

[Arons 1992] A review of time-compressed speech. MIT Media Laboratory, Speech Research Group, 1992.

[Asadi 1991] A. Asadi, R. Schwartz and J. Makhoul. Automatic modeling for adding new words to a large-vocabulary continuous speech recognition system. *In Proceedings of* International Conference on Acoustics, Speech, and Signal Processing, volume 1, pages 305-308. IEEE, 1991.

[ASI 1991] Voice Navigator manual. Articulate Systems, Inc., 1991.

[Bananafish 1991] ThoughtPattern Handbook. Bananafish Software, Inc., 1991.

[Blazie 1991] Blazie Engineering product catalogue: products for blind and visually impaired people. Blazie, pages 4-7, 1991.

[Boomer 1962] D. S. Boomer and A. T. Dittmann. Hesitation pauses and juncture pauses in speech. *Language and Speech*, 5:215-220, 1962.

[Brady 1968] P. T. Brady. A statistical analysis of on-off patterns in 16 conversations. *Bell Systems Technical Journal*, 47(1):73-91, 1968.

[Brennan 1992] An adaptive feedback model for speech interaction. Unpublished draft, 1992.

[Butterworth 1977] B. B. Butterworth, R. R. Hine and K. D. Brady. Speech and interaction in sound-only communication channels. *Semiotica*, 20(1-2):81-99, 1977.

[Clark 1991] H. H. Clark and S. E. Brennan. Grounding in communication. In J. Levine, L. B. Resnick and S. D. Teasley, editors, *Perspectives on socially shared cognition*, pages 127-149. APA, 1991.

[Cypher 1986] A. Cypher. The structure of users' activities. In D. A. Norman and S. W. Draper, editors, *User Centered System Design*, chapter 12, pages 243-263. Lawrence Erlbaum Associates, 1986.

[Degen 1992] L. Degen, R. Mander and G. Salomon. Working with audio: integrating personal tape recorders and desktop computers. *In Proceedings of* CHI '92, pages 413-418. ACM, 1992.

[Fairbanks 1954] G. Fairbanks, W. L. Everitt and R. P. Jaeger. Method for time or frequency compression-expansion of speech. *Transaction of the Institute of Radio Engineers, Professional Group on Audio*, AU-2:7-12, 1954.

[Gaver 1989] W. W. Gaver. The SonicFinder: an interface that uses auditory icons. *Human-Computer Interaction*, 4(1):67-94, 1989.

[Goldman 1961] F. Goldman-Eisler. A comparative study of two hesitation phenomena. *Language and Speech*, 4:18-26, 1961.

[Green 1983] T. R. G. Green, S. J. Payne, D. L. Morrison and A. Shaw. Friendly interfacing to simple speech recognizers. *Behaviour and Information Technology*, 2(1):23-28, 1983.

[Grice 1975] H. P. Grice. Logic and conversation. In Cole and Morgan, editor, *Syntax and Semantics: Speech Acts*, pages 41-58. Academic Press, 1975.

[Hayes 1983] P. J. Hayes and D. R. Reddy. Steps toward graceful interaction in spoken and written man-machine communication. *International Journal of Man-Machine Studies*, 19:231-284, 1983.

[Heiman 1986] G. W. Heiman, R. J. Leo and G. Leighbody. Word intelligibility decrements and the comprehension of time-compressed speech. *Perception and Psychophysics*, 40(6):407-411, 1986.

[Luce 1983] P. A. Luce, T. C. Feustel and D. B. Pisoni. Capacity demands in short-term memory for synthetic and natural speech. *Human Factors*, 25(1):17-32, 1983.

[Machina 1992] Pencorder by Machina: instructions. Machina, 1992.

[Maclay 1959] H. Maclay and C. E. Osgood. Hesitation phenomena in spontaneous English speech. *Word*, 15:19-44, 1959.

[Malone 1983] T. W. Malone. How do people organize their desks? Implications for the design of office information systems. *ACM Transactions on Office Information Systems*, 1(1):99-112, 1983.

[Minifie 1974] F. D. Minifie. Durational aspects of connected speech samples. In S. Duker, editor, *Time-Compressed Speech*, pages 709-715. Scarecrow, 1974.

[Miyata 1986] Y. Miyata and D. A. Norman. Psychological issues in support of multiple activities. In D. A. Norman and S. W. Draper, editors, *User Centered System Design*, chapter 13, pages 265-284. Lawrence Erlbaum Associates, 1986.

[Muller 1990] M. J. Muller and J. E. Daniel. Toward a definition of voice documents. *In Proceedings of* COIS '90, pages 174-182. ACM, 1990.

[Noyes 1989] J. M. Noyes and C. R. Frankish. A review of speech recognition applications in the office. *Behaviour and Information Technology*, 8(6):475-486, 1989.

[O'Shaughnessy 1990] D. O'Shaughnessy. Speech Recognition. Speech Communication, chapter 10, pages 413-477. Addison-Wesley, 1990.

[O'Shaughnessy 1992] D. O'Shaughnessy. Recognition of hesitations in spontaneous speech. *In Proceedings of* International Conference on Acoustics, Speech, and Signal Processing, volume 2, pages 521-524. IEEE, 1992.

[Pastel 1991] DayMaker manual. Pastel Development Corporation, 1991.

[Pisoni 1982] D. B. Pisoni and E. Koen. Some comparisons of intelligibility of synthetic and natural speech at different speech-to-noise ratios. *Journal of the Acoustical Society of America Supplement*, 71(1):S94, 1982.

[Pisoni 1985] D. B. Pisoni, H. C. Nusbaum and B. G. Greene. Constraints on the perception of synthetic speech generated by rule. *Behavior Research Methods, Instruments & Computers*, 17(2):235-242, 1985.

[Portnoff 1978] M. R. Portnoff. Time-scale modification of speech based on short-time fourier analysis. Massachusetts Institute of Technology, 1978.

[Portnoff 1981] M. R. Portnoff. Time-scale modification of speech based on short-time fourier analysis. *IEEE Transactions on Acoustics, Speech, and Signal Processing, ASSP*, 29(3):374-390, 1981.

[Reich 1980] S. S. Reich. Significance of pauses for speech perception. *Journal of Psycholinguistic Research*, 9(4):379-389, 1980.

[Rochester 1973] S. R. Rochester. The significance of pauses in spontaneous speech. *Journal of Psycholinguistic Research*, 2(1):51-81, 1973.

[Rudnicky 1989] A. Rudnicky. The design of voice-driven interfaces. *In Proceedings of* Speech and Natural Language Workshop, pages 120-124. DARPA, 1989.

[Savoji 1989] M. H. Savoji. A robust algorithm for accurate endpointing of speech signals. *Speech Communication*, 8(1):45-60, 1989.

[Schmandt 1992] C. Schmandt. Speech synthesis. *Conversational Computer Systems*, chapter 5, In Publication, 1992.

[Soclof 1990] M. Soclof and V. W. Zue. Collection and analysis of spontaneous and read corpora for spoken language system development. *In Proceedings of* International Conference on Spoken Language Processing, 1990.

[Stifelman 1991] L. J. Stifelman. Not just another voice mail system. *In Proceedings of* AVIOS '91, pages 21-26. American Voice I/O Society, 1991.

[Voor 1965] J. B. Voor and J. M. Miller. The effect of practice upon the comprehension of time-compressed speech. *Speech Monographs*, 32(452-455)1965.

[Waterworth 1984] J. A. Waterworth. Interaction with machines by voice: A telecommunications perspective. *Behaviour and Information Technology*, 3(2):163-177, 1984.

[Weiser 1991] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):94-102, 1991.

[Wilcox 1992] L. Wilcox, I. Smith and M. Bush. Wordspotting for voice editing and audio indexing. *In Proceedings of* CHI '92, pages 655-656. ACM, 1992.

[Wilcox 1991] L. D. Wilcox and M. A. Bush. HMM-based wordspotting for voice editing and indexing. *In Proceedings of* Eurospeech '91, pages 25-28. ESCA, 1991.