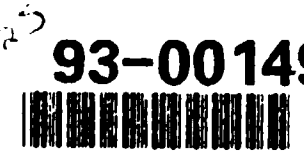
AFIT/GE/ENG/92D-34

ENHANCEMENT OF AUDIO LOCALIZATION CUE SYNTHESIS BY ADDING ENVIRONMENTAL AND VISUAL CUES

THESIS

Eric L. Scarborough
Captain, USAF

AFIT/GE/ENG/92D-34



AFIT/GE/ENG/92D-34

ENHANCEMENT OF AUDIO LOCALIZATION CUE SYNTHESIS BY ADDING ENVIRONMENTAL AND VISUAL CUES

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Electrical Engineering

Eric L. Scarborough, B.S.E.E.
Captain, USAF

December, 1992

Acknowledgements

The idea for this thesis topic came out of a series of lectures by Mr Rich McKinley during EENG 548, Human Factors. He presented the concepts behind human auditory localization, and the auditory localization cue synthesizer. Maj Steven Rogers and Dr Matthew Kabrisky agreed with my idea of bringing this technology to AFIT to incorporate 3-D sound with ongoing research and to try to enhance the system. There were times when I convinced myself to switch thesis topics altogether, however, Maj Rogers provided the enthusiasm and encouragement to help me see it through. I am also indebted to Lt Col Amburn for teaching me everything I know about computer graphics, and spending some of his time to keep me from wasting a lot of my time. Mark Ericson of AL/CFBA was my main source of ideas, knowledge and technical expertise for all aspects of the audio portion of the thesis.

I owe my loving wife, Jan, the most thanks, for the understanding, patience and for putting up with me during this endeavor. The best thing to happen to me during AFIT, was the birth of my second daughter, Leah. Thanks also to daughter number one, Cara, who kept me laughing and playing even though I "didn't have time". My thanks go also to my parents who provided the foundation for my interests and curiosity in life. Finally, I thank my Lord for providing me the opportunities to achieve more than I ever dreamed possible.

Eric L. Scarborough

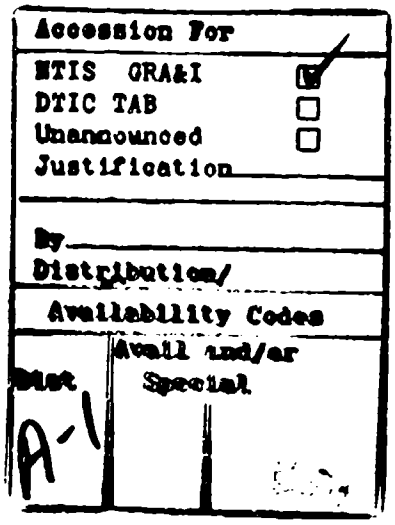


Table of Contents

List of Figures

Abstract

An audio localization cue synthesizer, the DIRectional Audio Display (DIRAD) was used to simulate auditory distance, room reflections, and to provide spatial audio for computer graphics images. The DIRAD processes input audio signals to generate spatially located sounds for headphone listening. The DIRAD can position audio sources around the head and these sounds are stable with respect to the listener's head position. An interactive, real-time simulation of auditory distance and room reflections was accomplished using the DIRAD in combination with a Silicon Graphics audio processor board installed in a Personal Iris 4D/35. Several demonstrations of auditory distance and the effects of early reflections are detailed, including a simulation of a direct sound source and three reflections that employed two DIRAD systems. Stored sound files were used to accompany three dimensional graphics images that were displayed on both a Silicon Graphics CRT and a three dimensional optical display device.

The use of the 4D/35 audio processor board proved to be an effective means of preprocessing audio for the DIRAD for these simulations. The combination of AFIT's Silicon Graphics workstations and the DIRAD proved to be a practical solution to the problem of combining virtual visual and audio cues.

ENHANCEMENT OF AUDIO LOCALIZATION CUE SYNTHESIS BY ADDING ENVIRONMENTAL AND VISUAL CUES

I. Introduction

1.1 Background

In recent years, several systems have been developed that can "position" sounds at virtual locations around a listener (30)(57:5-9)(15). These systems are known as audio localization cue synthesizers. Ideally, a person listening to an audio localization cue synthesizer over headphones will perceive the sound as coming from the desired direction and the sound should appear to be originating from a location outside of the listener's head.

The Air Force is interested in the development of this technology for several reasons. Audio localization cue synthesizers could be used in the cockpit of an aircraft to provide another means of getting information to a pilot, an information channel that has yet to be utilized. For instance, a pilot could keep track of another aircraft's position relative to his own position simply by radio communications. Another example would be a B-52 pilot who could readily identify which crew member was speaking at any given time based on the spatial location of the intercom audio signal. Air Force simulators will also benefit from the added effects of directional audio. Directional audio could be used in simulators to alert a user of objects that are not in his field of view, provide him with orientation information, or simply to enhance the simulation experience.

The human ability to determine the direction and distance of a sound is known as auditory localization. Many cues are used by human listeners in order to determine a sound's direction and distance. Several of these cues are encoded onto input signals by an audio localization cue synthesizer to simulate direct path sounds from any direction. Although the auditory experience provided by an audio localization cue synthesizer can be quite real, this thesis will enhance the simulation by incorporating some of the other cues that we use to localize sounds.

The localization synthesizers used for this research do not provide a way to model the effect that the listening environment adds to sound. Environmental cues include the effects that the listening environment imposes on distant sources, early reflections from the ground, walls and ceilings, and reverberation, or late reflections. Environmental cues have been found to play a role in the perception that the sound sources are originating at some distance away from the listener's head. The challenge of synthesizing directional audio with environmental effects is an active area of research (58). Interactive simulations of room reflections and distance cues will be developed in this thesis.

This thesis will also develop a means to add visual cues that will be spatially correlated with the synthesized audio. The incorporation of visual cues can be accomplished in many ways like positioning lights or other physical objects at the location of the synthesized audio. The method that will be used for this work will be the incorporation of computer graphic images at the sound source location. This method will allow for various types of visual objects to be represented, and allow for easy positioning and movement of the objects. Synthetic environments will be developed that will allow for computer generated images to be positioned and moved anywhere in the space surrounding the viewer. Thus, one of the goals of this thesis is to create an interactive environment in which images and sounds can be positioned together at various locations in the space around the user.

1.2 Definitions

This section provides some working definitions of terms that will be used frequently throughout this thesis.

Binaural sound, or two ear sound, refers to having a separate audio signal presented to each ear. This is how audio information arrives to our ears naturally. The spectral shape of each signal and differences in the two signals are used to give a listener information about the direction and distance of sounds as well as information of the environment from which the sound originated.

Binaural recording is a method of making recordings that capture the temporal and spectral cues that exist in binaural sound. This is usually accomplished by making recordings from microphones at the entrance to the ear canals of a manikin, and listening to this recording over headphones. When binaural recordings are heard, the desired perception is that the sounds are originating outside of the listener's head. More realistic manikin features, in terms of outer ear shape, head size and torso size and proportions, and the recording environment can impact the "externalized" effect of the recording. Binaural recordings can also be made by placing microphones in the ears of humans, or by recording the outputs from an auditory localization cue synthesizer.

A *synthetic environment* is a graphical computer simulation which can be interacted with using intuitive, natural means. Traditionally, synthetic environments have relied on visual information as the only means of providing information to the user; recently however, more sensory inputs like sound and tactile feedback are being incorporated to enhance the effects of actually being "in" the virtual world.

1.3 Problem

The audio localization cue synthesizer, the DIRectional Audio Display (DIRAD), that was used for this thesis does not have the means of simulating dynamic

environmental effects like reflections and distance. Also, the methods and procedures to combine visual cues from synthetic environment systems and audio cues from an audio localization cue synthesizer do not presently exist at AFIT.

Reflections have recently been simulated with the Convolvotron, (another audio localization cue synthesizer), in order to enhance the perception of the sound originating external to the listener's head (16). Synthesized audio from the Convolvotron has been used in conjunction with graphics images (50)(32), and the DIRAD has been used with visual images in high fidelity flight simulators (2).

1.4 Research Objectives

This thesis will develop a means of preprocessing audio signals that will be used as inputs to an audio localization cue synthesizer, to allow for the simulation of auditory distance and surface reflections. Also, this thesis will develop the procedures required to integrate an audio localization cue synthesizer to various synthetic environments in order to spatially correlate visual cues and directional audio.

1.5 Scope

In this thesis, a series of control programs will be developed that will allow for an audio localization cue synthesizer to be used with AFIT's Silicon Graphics workstations. These control programs will provide interaction with the DIRAD system, generation of a distance cue (using amplitude scaling), and simulated room reflections. This thesis will develop a means to incorporate spatially located audio with visual images from AFIT synthetic environment software. Finally, binaural recordings will be made using an acoustic research manikin in a natural environment and two state of the art audio localization cue synthesizers.

1.6 Approach

The following general approach will be used to complete the objectives of this effort. The problem as stated above consists of two distinct parts, 1) develop a means of preprocessing synthesizer inputs so that environmental effects can be incorporated, and 2) provide visual cues to accompany the synthesized audio. These tasks will be carried out in parallel.

Binaural recordings will be made from the synthesizers and from an acoustic research manikin in natural environments. The simulation of the environmental effects of distance and room reflections will be carried out using an audio processor board installed in a Silicon Graphics workstation, in conjunction with the audio localization cue synthesizer.

Concurrent with the above work, the localization cue synthesizer will be used to provide directional audio for objects in a computer graphics based synthetic environment. Existing AFIT synthetic environment software will be modified to create the graphics and interface to the localization cue synthesizer.

1.7 Thesis Outline

The next chapter is a literature review consisting of four main sections. The process of human auditory localization is covered first, following that is a background on binaural recordings, then a review of audio localization cue synthesizer research and development, and finally a brief review of synthetic environments.

Chapter III presents the methods used to perform the work described in this thesis. This chapter is developed according to the three main projects that were accomplished. The experiments are presented in the general order in which they were carried out, comparative recordings, simulation of distance and reflections, and incorporation of visual cues.

In Chapter IV, the results of this thesis are presented. A discussion of the subjective comparison of the binaural recordings is presented first, then the results of the distance and reflection simulations, and finally, a presentation of the results of combining the directional audio with visual images.

To conclude, Chapter V will summarize this research and present recommendations for follow-on research in the area of enhancing synthesized directional audio and its use in AFIT synthetic environments.

II. Literature Review

2.1 Introduction

This literature review will provide a current assessment of four main areas. The first section is an overview of the research pertaining to human auditory localization, stressing the important cues that allow us to localize sounds. Following that is a review of the technique of binaural recording. The two audio localization cue synthesizers that will be used in this thesis will be presented next, and finally, a review of synthetic environment research is presented.

2.2 Human Auditory Localization

2.2.1 Introduction This thesis will provide enhancements to an existing audio localization cue synthesizer. It is therefore important to understand the process of human auditory localization.

Mackous and Middlebrooks give insight into the capabilities of the human localization process:

> ...the auditory system is capable of combining information across classes of cues and across frequencies to synthesize a unitary spatial image (29).

Localizing a sound in space is a very complex process and involves the use of many features or cues. A very thorough review of the process of human auditory localization is given in Blauert's 1983 book *Spatial Hearing: The Psychophysics of Human Sound Localization* (4). Two recent AFIT theses have also presented reviews of localization theory (30)(11).

Some introductory remarks are made next to define a set of planes that are used in the literature to describe positions in space relative to the listener. Then a brief discussion on the accuracy of human localization and how this is measured is

presented. The rest of this section will focus on some of the cues that allow humans to perform audio localization. These include interaural time difference, the effects of the head and outer ear, head motion, environmental and visual influences.

2.2.2 Defining Planes In order to discuss the various positions around a listener three defining planes are used. The planes are the horizontal, median, and frontal. The horizontal plane bisects the head at ear level parallel to the ground. The median plane bisects the head between the eyes from front to back, and the frontal plane bisects the head from top to bottom through the ears. (see Figure 1 adapted from Blauert (4:14, Figure 1.4)) The azimuth angle (φ) is 0° directly in front of the listener. Elevation angle (δ) is 0° at ear level, −90° below and +90° directly overhead.

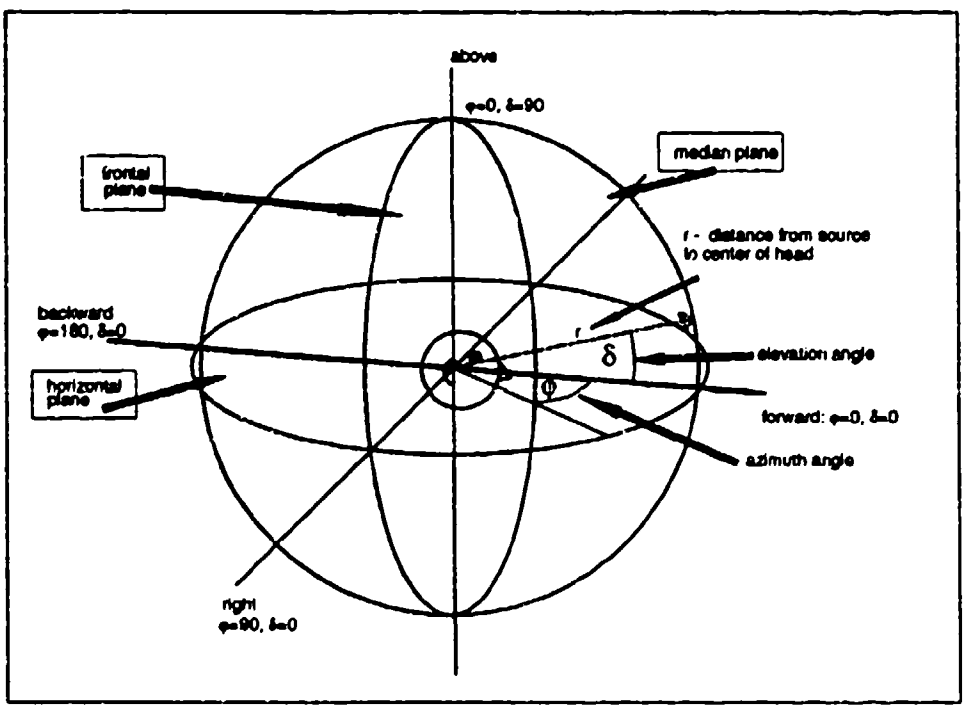


Figure 1. Head Related Planes, Coordinate Axes, and Angles (adapted from Blauert) (4)

2.2.3 Human Localization Performance Localization performance is most often tested by presenting a sound source at some location in azimuth and elevation and at some fixed distance; subjects are then required to indicate where the sound is originating from by some type of pointing or reporting method. It should be noted that this measurement (localization performance) is not the same as the measurement of audio acuity. Audio acuity measures the distance that two sources must be separated before they are heard as two distinct sources. Audio acuity is measured in terms of the minimum audible angle, and has been found to be about 1 degree for sources located in front of the listener (4:38). The minimum audio angle measurement is dependent on the spectral content of the source but, in general, the angle increases as the source is moved to the sides of the head, up to as much as 10 degrees for white noise pulses (4:41).

A 1984 effort by Oldfield and Simon (35)(36) tested localization performance capabilities for subjects under normal hearing conditions and with the cavities of the external ear filled, using a custom fit insert that allowed the entrance of the ear canal to be unobstructed. A loudspeaker on a movable boom was used to present the stimuli to the subjects at 171 positions in azimuth and elevation. The subjects pointed to the sound using a "special gun". The trigger of the gun activated three still cameras that recorded the position of the gun. These pictures were later analyzed to calculate the subject's estimate of sound source location. Under normal pinna conditions, the subjects had an average azimuth error of 9.1 degrees, and elevation error of 8.2 degrees. With the insert in the pinnae, the subjects had a large increase in elevation judgement errors and more front-back reversals.

More recently, a study by Mackous and Middlebrooks (29) found that most horizontal and vertical errors in localization were less than 10 degrees. They had subjects point to the sound source by looking at it. The head position and orientation were then measured using a magnetic head tracking system.

Localization performance has also been tested to compare the ability to localize free-field sounds with the ability to localize synthesized sounds (60)(14)(13).

2.2.4 Interaural Time Difference An early theory of localization was the duality theory (45:33). This theory asserts that for low frequencies, below 2,000 Hz, the interaural time difference (ITD) is the main cue for localization and for higher frequencies interaural amplitude differences is the primary cue. The arrival time of sounds to the two ears is dependent on the position of the source relative to the head and on the speed of sound. This time difference ranges from 0 to 750 μs (30:56-63). The effect of time delay has been studied extensively by presenting sounds over headphones and delaying, by some small amount, the sound presented to one of the listener's ears. This delay causes the listener to perceive a shift in the position of sound towards the ear receiving the first arriving sound (43). Obviously, the ITD is a very important localization cue; however, ambiguities arise because nearly identical time delay values can result from sources that are located symmetrically about the frontal plane. Therefore, ITD cannot be the only cue used for localization on the horizontal plane. The effective range of time delay that humans can perceive depends on the type of stimulus that is used. For pure tones, the time (or phase) information of the ongoing signal can be tracked until one of two limiting conditions, listed below, is met (13)(4:148-149).

1. The ITD is equal to or larger than one-half the period of the tone. This means that the phase can change more than 180 degrees in the time it takes to travel between the ears. Once this happens, subjects report hearing two separate tones, or a single tone offset towards the "leading" side. For sources located on the side of the head this occurs at about 750 Hz.
2. As the source is moved closer to the median plane the time delay decreases and the limiting frequency described above, increases. The limiting condition in this case is the firing rate of the auditory nerves. On average, the auditory

system can only track differences in the phase of tones for frequencies up to about 1600 Hz.

For sounds other than pure tones, the human auditory system is capable of detecting differences in ITD of about 10 μs (4:153, Table 2.3) in transient sounds or in low frequency envelopes (below 1600 Hz) of amplitude modulated signals.

2.2.5 Effects of the Head and Outer Ear The combined diffraction effects of head and pinnae, or outer ears, were studied in 1947 by Wiener (55). He found that by measuring the sound pressures at the entrance to the ear canal, he could estimate the "... obstacle effect of the head and pinna."

The pinnae act as filters that are dependent on source location. The spectral shaping applied by the pinna, has been shown to be important in localizing elevated sources on the median plane by Roffler and Butler (40). They used several types of sources and attempted to eliminate the effects of the pinnae by flattening them against the head with a Plexiglass headband. Holes in the headband were positioned over the entrances to the ear canals. They found that with the normal use of the pinnae, sounds were judged at the proper elevation if they were wide band, and contained frequencies above 7000 Hz. Without the use of the pinna, these sources were not properly located. Gardener (17) also found similar results by filling the folds of the pinna with putty. He showed the existence of a monaural component to the pinna cue, by presenting sounds when one pinna was occluded. His subjects were able to localize sounds better than when both pinnae were filled. The monaural cue of the pinna filtering relies on the shape of the spectrum from the single ear, and the binaural pinna cue discriminates differences in the spectra of the two ear inputs (29).

In a 1974 paper (48), Shaw combined the results of 12 studies that measured the frequency and directional response of the outer ear. Shaw shows the dependence of the transfer function of the head and outer ear on source location. Figure 2

(from Shaw (48:1856, Figure 9)) shows typical plots of the position and frequency dependence of the Head Related Transfer Function (HRTF).

The pinna's filtering effect is due to the resonances and reflections resulting from the shape of pinna (62). In 1977, Mehrgart and Mellert (31) measured the transfer function of the outer ear of human subjects for positions in azimuth and elevation, and provide HRTF data out to 15 kHz. Wightman and Kistler (59)(60) provide a detailed description of the process of measuring and processing HRTFs, and were successful in synthesizing the filtering properties of the HRTF using digital signal processing. They measured the response of the pinnae of 10 subjects at 144 positions in azimuth and elevation. Recently, Wightman and Kistler (61) found that ITD cues are dominant when compared to spectral cues as long as the sound stimuli contains frequencies below 5 kHz.

Acoustic manikins like the Knowels Electronics Manikin for Acoustic Research (KEMAR) have also been used to study the filtering effect of the pinna (6). KEMAR was used to measure HRTFs at one degree increments in azimuth for McKinley's thesis work (30:25-28). A detailed study of HRTFs using KEMAR is given in a 1991 paper by Han (22). He discusses how another cue for elevation may be the shifting of certain slopes present in the transfer function, and how this slope shift might be tracked by a neural receptive fields.

2.2.6 Head Motion Head motions are believed to be important because of the changes in the localization cues that result (45:41). Wallach (54) designed an experiment to show how side to side head movement can aid in the localization of sounds that are at some angle above ear level. He was able to simulate an elevated source by presenting sounds over a ring of speakers in the horizontal plane. The speaker selection was dependent on desired source elevation and head position. For instance, a source directly above the listener could be simulated by presenting the sound over the speaker directly in front for any head position. Since there was

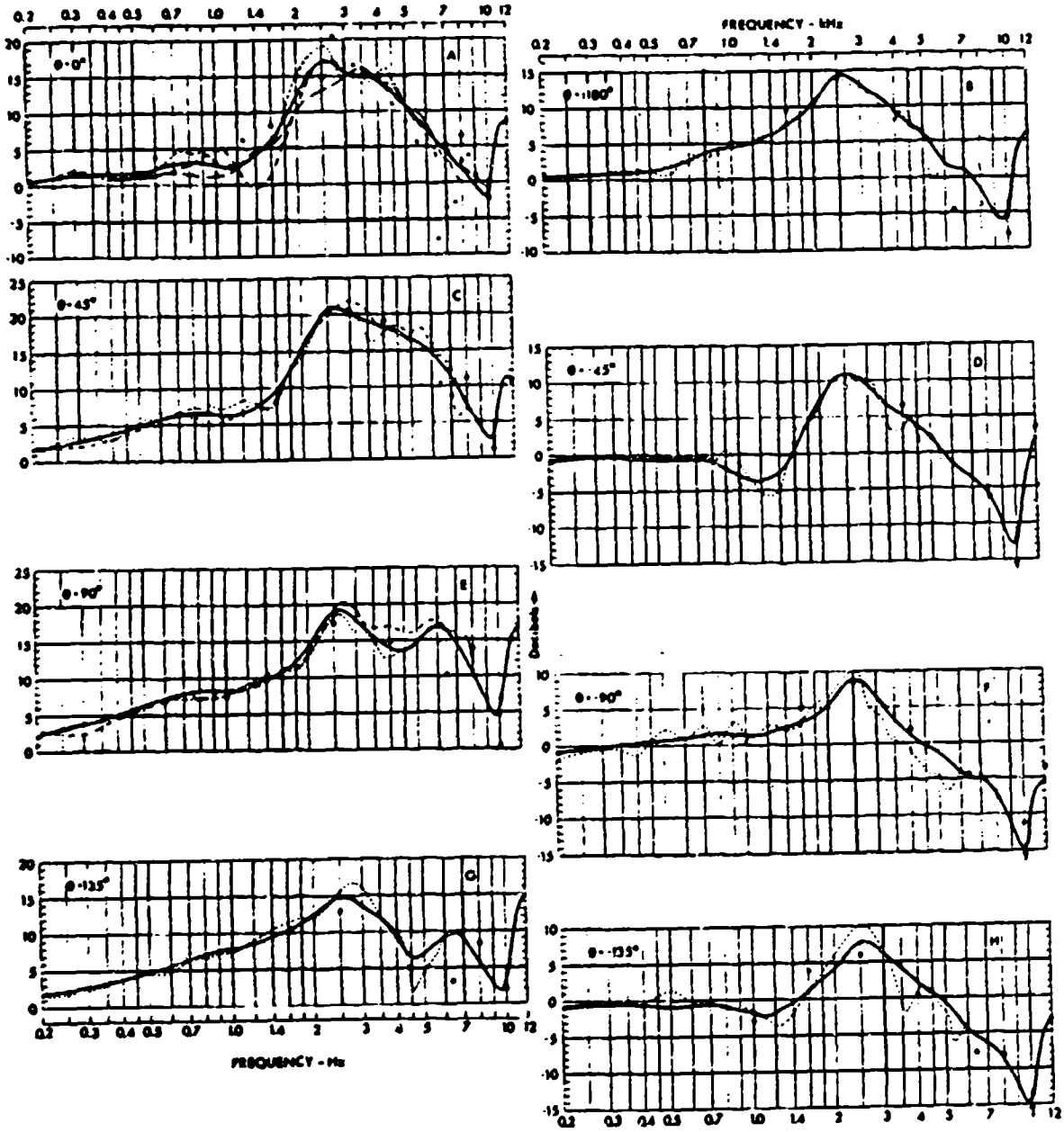


Figure 2. Typical Head Related Transfer Functions, from Shaw (48)

no change in the sound direction with head motion, the sound was perceived as coming from directly above. Lambert (27) worked out the theoretical calculations for localization of sounds in azimuth and range by using the rate of change of various parameters due to head motion.

2.2.7 Environmental Cues In normal listening conditions, the ear inputs not only contain the cues already discussed, they also contain the effects of the environment. Two environmental effects are reflections and reverberations. The term reflection refers to a version of the primary sound that results from the sound wave arriving at the ear from a path other than the direct one. Reflections are due to the sound impacting the ground, walls, or other nearby physical objects. Generally speaking, for flat surfaces, reflections can be modeled as sources located at the point of reflection. The length of time between the primary sound arriving and the reflection's arrival is a determining factor in the listener's perception. As discussed previously, time delays below about 1 msec are attributed to the direction of the source and the later arriving sound is inhibited. For time delays between 1 and 50 msec, reflections are perceptible as an increase in loudness and spaciousness (4:224). The reflection is perceived as a separate auditory event after about 50 msec, and is termed an echo (4:224). These findings show that when simulating reflections, delay time is a critical parameter that controls how the listener will perceive the delayed sound.

Cues from the environment have been found to play a large part in the perception of sounds originating from outside the head and the perception of auditory distance (39)(42)(7).

The concept of "out-of-head-locatededness" or externalization of sounds is fundamental to binaural synthesis. Plenge (39) showed that binaural sound from a normal li ing environment allowed listeners to localize the sounds externally. The ratio of energy from the direct path and the indirect paths is known as the Acoustic

Ratio (AR) and according to Sakamoto et al. (42) is "... one of the main factors which influence the subjective distance." They further assert that some amount of distance must be perceived in order to experience externalization. This report explains three experiments which manipulate AR levels in various ways and shows that when the AR is above a certain threshold, the incidence of subjects reporting externalization increases dramatically. The addition of just a few or even a single reflection can have an impact on externalization as shown in an experiment by Rubak (41). He added two pairs of reflections to a binaural recording made in an anechoic chamber in order to simulate a natural listening environment. Subjects reported better localization ability and reduced occurrences of in-head-localization (IHL).

The ratio of reflected energy to direct path energy is one of the cues responsible for distance perception that Coleman (10) discusses in his 1963 review of auditory depth perception. Coleman also discusses the modification of amplitude and the frequency spectrum due to changes in source to listener distance; both cues provided by the listening environment. Butler et al. (7) found that binaural recordings made in a reverberant chamber elicited increased (about four times greater) perception of distance than similar recordings made in an anechoic chamber.

The amplitude of the sound wave arriving at a point in a free field will fall off according to the inverse first power law (10): change in dB $= 20\log(r/r_0)$. In this equation, r is the distance to the source and r_0 is the distance to a reference point. Strybel and Perrott (51) found that at distances greater than 304 cm, subjects were apparently using the change in amplitude as their primary distance cue. At shorter distances 49 cm to 304 cm subject responses showed more error. This finding suggests that at shorter distances, other cues are used to judge distance and these other cues are less accurate.

Much effort has gone into developing programs and procedures for calculating how reflections will develop in a given room. The most popular technique seems to be the ray tracing method. This technique is described in (46)(25), and has been

used by NASA AMES researchers to simulate room acoustics (3) for possible use in their audio localization cue synthesizer.

2.2.8 Visual Cues Regarding the influence of the visual system on the ability to localize, Blauert states:

> The assumption underlying visual theories may be stated as follows: What the subject sees during sound presentation, and where he sees it, are factors determining the position of the auditory event (4:193).

The barn owl has been studied to help understand how the cortical map for auditory space is determined (24). By raising infant owls with eyeglasses that caused a shift of the visual image, researchers found evidence that the auditory map is learned after birth and trained by visual reinforcement. The receptive fields corresponding to spatially located visual and auditory stimuli were mapped together in these owls when the prisms were on. When the prisms were removed, the auditory map showed a shift by an amount corresponding to the visual shift provided by the glasses. This interesting finding may be the reason for the strong influence that the visual channel has on our ability to attribute a location to an auditory image.

Pick and Warren, in a 1969 paper (37), showed that the visual system biases the auditory system, and that the auditory system provides no bias to the visual system. They had subjects look at objects through prisms. An audio signal originated at the actual location of the object. The distorted visual image caused the subjects to distort their judgement of the location of the audio. This bias that the visual system applies to auditory system is often referred to as the "ventriloquism effect".

An experiment performed by Warren (47:16), explains the phenomenon of "visual facilitation of auditory localization" as being the improvement in audio localization precision when the eyes are open and looking at a textured visual field. Warren's experiments used sound sources that were not visible. He found that when subjects

had their eyes open and were looking at the visual field (a cloth screen), they could do better at an audio localization performance test then when they had their eyes shut.

Warren further investigated these findings in 1972 (38), to detail the effects of eye movement. He found that if the subjects were required to make eye movements towards the auditory target, localization performance improved. He also found that the target directed eye movements were more effective in aiding localization in a lighted environment than when made in the dark.

Jones and Kabanoff (23) confirmed the work of Warren described above, and found further that directed eye movements away from the target decreased localization precision as compared to having the subject fixate straight ahead. They assert that "... voluntary movement rather than visual map is likely to provide the framework for spatial judgements." In other words, eye movements are the key to updating spatial information, and these movements are better controlled in a lighted visual field.

The literature shows that the process of auditory localization is quite complex relying on many cues to discriminate direction and distance. Some of these cues are currently used to synthesize directional audio, this thesis will further the perception of directional audio by adding a synthetic distance cue (based on the inverse first power law), room reflections and attempt to exploit the "ventriloquism effect" by introducing visual reinforcement.

2.3 Binaural Recording

Sunier writes in a 1986 article (52) that the method of binaural, or two ear, recording is the best technique available for reproducing sounds as they are heard naturally. He gives a review of the history of binaural reproduction and explains the principles of "dummy head" recording in this article. The basic concept of binaural recording is to record, through separate channels, the outputs of microphones that

are separated by the distance of the head for playback though headsets (53). The effects of binaural recordings can be enhanced by placing the microphones in a model of the human head and ears (28)(18)(55).

This technique allows for capturing of all of the cues described above, except of course for visual cues. Audio engineers are looking into ways of processing binaural sounds like mixing, adding binaural reflections, and synthesizing binaural signals (20). Work is also being done to try to allow for binaural recordings to be played back over loudspeakers instead of headsets (21)(33). Although binaural recordings provide powerful examples of spatial hearing, they cannot provide any degree of interaction.

In an attempt to provide binaural interaction, systems have been developed that provided a listener with real-time audio from a manikin's ear microphones (8)(13). One of these systems used the listener's head position to position a manikin head (via servos) with respect to a fixed sound source. The other system used listener's head position to move a sound source location with respect to a fixed manikin head. Both systems allowed the listener to localize the sound, but were somewhat impractical in that they required a manikin located in an enclosure to generate the audio information.

To provide a truly interactive system, a method of synthesizing the effects that are captured with binaural recording is required.

2.4 Localization Synthesizers

The basic concept of present day audio localization cue synthesizers is the convolution of the input audio signal with the impulse response of the left and right HRTFs to generate the binaural signals (56:3). The appropriate HRTFs are selected based on the position and orientation of the listener's head and the location of the source. The premise behind the synthesis of localizable sounds is given by Wightman and Kistler as:

> The basic assumption that guides our approach is that, if the acoustical waveforms at a listener's eardrums are the same as in free field, then the listener's experience should be the same (59:859).

This review will cover two specific audio localization cue synthesizers, although there are several in various stages of development (57:5-9). The first system is the Convolvotron, built by Crystal River Engineering for the NASA Ames VIrtual Environment Workstation (VIEW) project (57:9), and currently available as a commercial product (12). The other system is the Directional Audio Display (DIRAD) (30) that was designed and fabricated at the Bioacoustics and Biocommunications Branch of the Armstrong Laboratory (AL/CFBA). A brief review of the Convolvotron system is given below, the DIRAD system will be covered in detail in Chapter III since it was the primary audio localization cue synthesizer used in this thesis work.

Wenzel, in her 1991 technical report (57), describes NASA's development of a three dimensional virtual acoustic display. The main cues that were used to transform the monaural audio into the binaural simulation in the Convolvotron were HRTFs, the listener's head position and orientation, and distance. The magnitude and phase differences of the HRTFs were measured from human subjects at 144 positions in azimuth and elevation around the subject (57)(59). The distance cue is simulated by amplitude scaling (16). The system is commercially available as set of two PC compatible boards. The boards accept as input head tracker coordinates, up to four analog audio inputs, and software specification of the desired source location. The output of the system is the analog left and right binaural synthesis that is presented over headphones. An interpolation routine is used to generate filter coefficients to position a sound at positions where the HRTF was not measured. The system requires a set of HRTFs to be loaded for use. Several sets of HRTFs, measured on different human subjects, are supplied with the system.

A paper by Foster, Wenzel, and Taylor (16) describes two methods that have been used to simulate environmental effects with the Convolvotron. The "static

modeling" technique uses HRTFs that are measured in "real" room, this is limited to either static head position, static source location, or both. The "dynamic modeling" technique allows for full interaction in that it places sources at the bounce points of the first few reflections. These "reflection sources" are movable and can be repositioned with listener's head movement or source movement.

2.5 Synthetic Environments

A synthetic environment is a computer generated graphics display that allows various means of user interaction. These interactions are designed to be natural and intuitive. Some of the human-machine interfaces that are used in synthetic environment research are:

- Head Mounted Displays (HMD)
- Head position and orientation measuring equipment
- Hand and finger position sensing equipment
- Force reflection devices
- Audio localization cue synthesizers

Gerken in his 1991 AFIT thesis (19) provides an introduction to most of these devices, and an excellent review of the synthetic environment research accomplished at AFIT over the last few years. As part of his thesis work, he concentrated on developing software for the DataGlove, a hand and finger position measuring system developed by VPL Research.

Similar research is being carried out at the University of North Carolina (9), where researchers are developing two specific applications of synthetic environment technology. The first is an interactive molecular study environment where chemists can study and actually manipulate models of molecules. The other application is an architectural walk-through environment. This simulation allows users to walk

through a virtual building on a treadmill that has been incorporated into the display system.

NASA is also developing synthetic environment technology for use in simulations, and as a future interface to space based equipment (34).

2.6 Summary

This literature review provides a foundation for the task of enhancing the capabilities of an audio localization cue synthesizer. The areas of human localization, binaural recording, localization cue synthesizers, and synthetic environments were covered. This review provides the knowledge base required for the integration of environmental and visual cues and existing audio localization cue synthesizer, the DIRAD.

III. Methodology

3.1 Introduction

As stated previously, the DIRAD audio localization cue synthesizer was used to provide directional audio for the majority of this thesis. The process of adding environmental and visual cues to the DIRAD system was carried out in three stages:

- Binaural recordings made with KEMAR in a semi-reverberant setting were compared to recordings made with two audio localization cue synthesizers, the Convolvotron and the DIRAD to gain insight into the problem.
- Development of a set of graphical control panel programs to allow for source positioning with simulated distance cues and room reflections using the DIRAD.
- Development of virtual environments to provide visual reinforcement of the audio stimulus position.

This chapter starts out with a section on the details of the DIRAD system. Following that, the three main experiments are presented in the order in which they were performed.

3.2 DIRAD, an Audio Localization Cue Synthesizer

The DIRAD audio localization cue synthesizer used for this thesis was developed in 1987 at Armstrong Laboratory (30)(2). The DIRAD system block diagram is shown below in Figure 3. The system can be configured (via ROM change), to operate in several different modes. Two separate DIRAD systems were employed in this thesis. One system was configured for azimuth only source positioning. The other system that was used allowed for azimuth and elevation source positioning. Both systems permitted independent positioning of two audio channels.

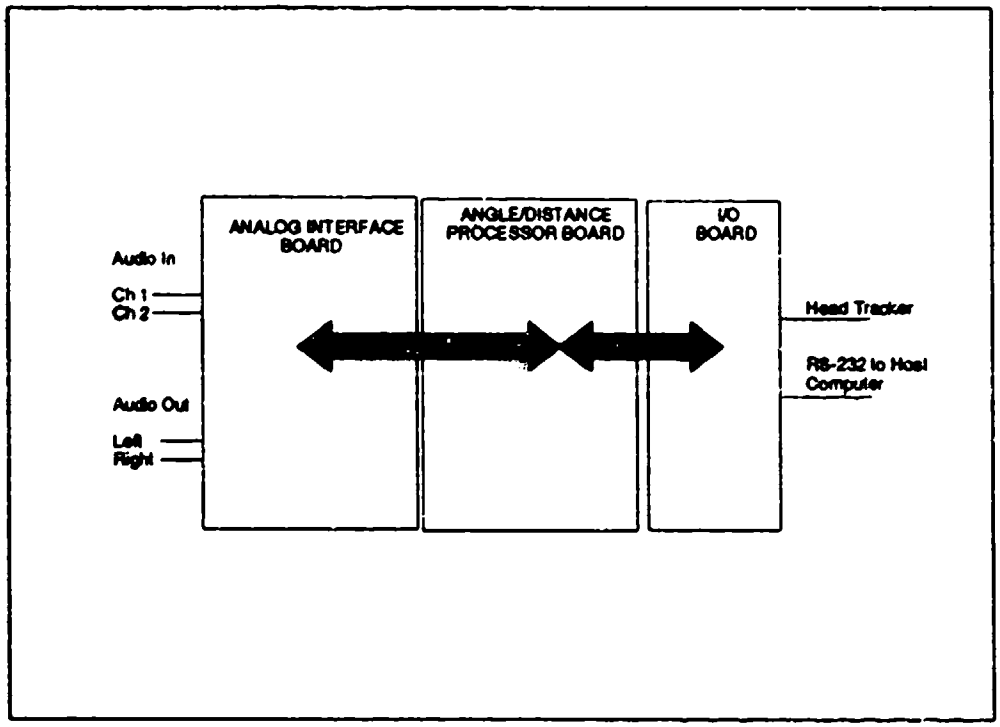


Figure 3. DIRAD System Block Diagram

The system communicates with a host computer via an RS-232 interface. This communication line is used to initialize the synthesizer firmware, position the audio sources, command the synthesizer to establish communications with the head tracker system, poll a subject response button, report the subjects head position to the host computer and to perform a variety of other commands (listed in Appendix A) (1).

Once commanded, the synthesizer can communicate with one of three head tracker systems: The Polhemus ISOTRACKER, the Polhemus 3SPACE tracker, and Ascension Technology Corporation's "The Bird" system. No interface between the host computer and the head tracker is required for standard operations.

The basic theory of operation of the DIRAD is as follows. The angle processor board computes the difference in angle between the desired source location and the subject's head position. This angle is used to call up stored HRTFs for the left and right ears. The DIRAD stores the magnitude of the HRTF only and the left and right pinnae are assumed to be identical. The directional pinna filtering is accomplished by carrying out the convolution of the impulse response of the left and right HRTFs with the digitized audio input.

A stored time delay value is also called up based on the head to source angle difference and is applied to the appropriate (left or right) signal. The resulting discrete signals are then fed to digital-to-analog converters, after which, the left outputs from channel 1 and channel 2 are mixed and the resulting amplified signal is sent to the left headphone loudspeaker, likewise for the right side outputs.

The current versions of the DIRAD employ five TMS320-C25 Digital Signal Processing (DSP) chips. One is used for handling all I/O functions, two are used for angle processing and two are used for "distance" processing. The distance processors are not, in current DIRADs, used for dynamic distance simulation. They have, however, been used to simulate a single, floor reflection at a fixed angle of $\delta = -37°$.

The HRTFs for the azimuth only DIRAD were measured using the KEMAR manikin at 1 degree increments in azimuth for a total of 360 stored HRTFs. The azimuth and elevation DIRAD contains 272 stored HRTFs. These HRTFs were measured with KEMAR in the center of a geodesic sphere of speakers located in the AL/CFBA anechoic chamber. The maximum separation angle between any two speakers on the geodesic sphere is 15°.

One other point of interest concerning DIRAD HRTFs is that a new manikin, the Bioacoustic Anthropomorphic Manikin for Auditory Research (BAMAR), has recently been developed by AL/CFBA to measure transfer functions. BAMAR allows individualized pinnae shapes to be used and the manikin's head width can be adjusted to match an individual's bi-tragion (ear to ear) distance.

3.3 Experiment 1, Comparative Recordings

3.3.1 Introduction The initial objective of this thesis was to make binaural recordings of sounds placed at specific locations around a listener's head using three techniques: the DIRAD, the Convolvotron, and real world examples using the acoustic manikin, KEMAR. Comparative listening between the synthesized recordings and the manikin recordings allowed us to appreciate the importance of the acoustic properties of the environment in helping to bring the sound "outside of the head".

3.3.2 Equipment The systems that were used to make these recordings are shown in Figure 4. The audio for this experiment was generated on a Sun workstation which sends its output to a Ariel Pro Port stereo audio DSP interface box, Model 656. A monaural recording of this audio was made using a Sony Digital Audio Tape recorder, Model TCD-D3. This recording was then output from the DAT to a Crown Model PS-200 Audio Power Amp which drove a 4.5 inch Bose loudspeaker, Model 108515. A KEMAR manikin was used to make the binaural recordings. KEMAR's

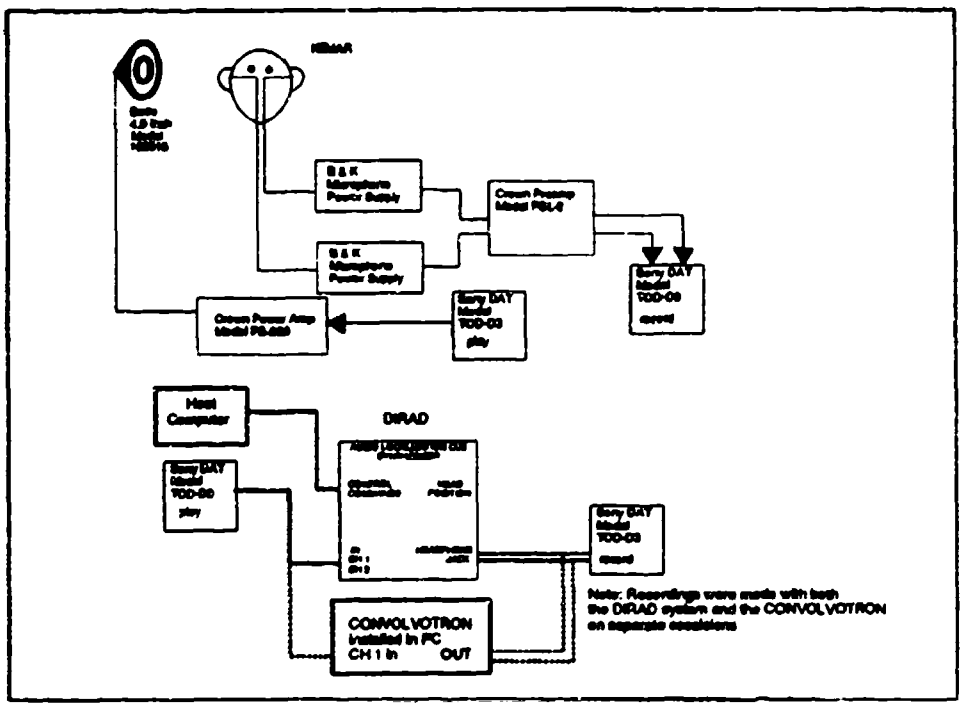


Figure 4. Comparative Binaural Recordings Block Diagram

left and right ear microphones were Brüel and Kjaer Model 4165, (1/2 inch diameter, free field laboratory condenser microphones) The microphones were powered with two Brüel and Kjaer microphone power supplies, Type 2804. The output of the microphones was sent to a Crown Audio Pre Amplifier, Model PSL-2, then to a second Sony DAT Model TCD-D3 for left and right channel recording.

3.3.3 Procedure The main use of these recordings was to evaluate the subjective differences between a real world binaural recording, and the synthesized binaural recordings. The audio source used in this experiment was speech from the DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT) database. Five sentences were recorded each from a male speaker and a female speaker. This was followed by five one second noise bursts generated by using the Entropic Research Laboratory, Inc. software package called: Entropic Signal Processing System (ESPS). A program in this package, TESTSD, was used to generate the uniformly distributed noise bursts. These selections were recorded on the DAT. The manikin was set up near the center of an auditorium. The auditorium is a 42 foot by 26 foot room that is furnished with cushioned chairs and is carpeted. Figure 5 illustrates the room layout.

The sounds were played on one of the DATs, through the power amplifier, over the loud speaker which was at ear level height to KEMAR. The microphone outputs in KEMAR's ears were sent to a second DAT. The left and right channels were recorded simultaneously. The position of the speaker was moved to eight different positions around the manikin, at 45 degree intervals at a fixed radius of seven feet. The distance of seven feet was chosen because the HRTFs for the DIRAD were measured at this distance. The HRTFs used in the Convolvotron were measured at 1.38 meters (4.55 feet) (59) however, the Convolvotron system has a software controlled distance parameter, that was set at seven feet. It should be pointed out that the Convolvotron was used only to gain insight into the use of a second synthesizer, and was not available for thorough comparison.

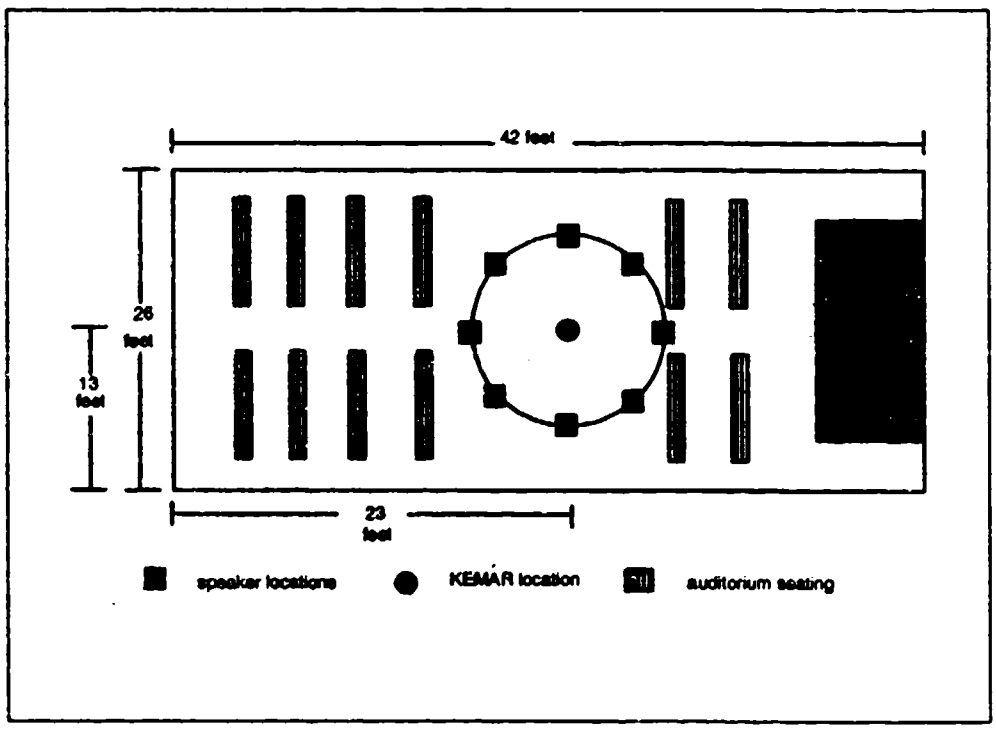


Figure 5. Room Layout for Binaural Recordings

After all eight positions were recorded, the equipment was taken into the AL/CFBA anechoic chamber. Another monaural recording was made of the voice and noise bursts as played through the same speaker set up that was used in the room recordings. This was accomplished so that any characteristics of the speaker would also be on the stimuli that was played through the synthesizers. The sound was recorded monaurally using the second DAT and a 1 inch diameter, Larson Davis type 2570 microphone. This microphone was mounted a microphone stand at equal height. The microphone was powered by a Brüel and Kjaer Type 2804 microphone power supply. The output of the microphone was sent to a Crown Audio Pre Amplifier Model PSL-2, then to a second Sony DAT Model TCD-D3 for monaural recording.

The DIRAD synthesizer recordings were made as follows. The recorded voice and noise bursts were used as input to channel one of the DIRAD. A Grid laptop personal computer running the DIRAD demonstration software was used to set the DIRAD configuration to: azimuth only, no head tracker. The demo software was then used to send the source position commands. Recordings of the source were made at the same locations as the manikin:

- azimuth: $\varphi = 0, 45, 90, 135, 180, 225, 270, 315$ degrees.
- elevation: $\delta = 0$ degrees. (ear level)

Next, the same procedure was accomplished with the Convolvotron. The input was connected to channel 1, (4 available) and a Convolvotron demonstration program was used to position the azimuth, elevation, and distance. The distance was set to seven feet, elevation to ear level, and azimuth at the required eight positions.

Selected portions of the recorded noise bursts were digitized using the SG 4D/35 audio system (described below). The recordings were digitized at 16kHz sample rate, 16 bit resolution. These sampled representations were then played over headsets, Yamaha, Model YH-1. A program was written that allows a user to choose

an azimuth position (from the eight that were recorded) and listen to the noise bursts from all three binaural techniques. This program was used to compare the out-of-head locatedness of the sounds at various positions, for each technique. The speech recordings were evaluated by listening to the DAT.

3.4 Experiment 2, Environmental Cue Simulation

3.4.1 Introduction This section describes the equipment and process used to interface the DIRAD to a Silicon Graphics workstation and how this configuration was used to simulate auditory distance and room reflections. A short background of audio distance perception and acoustic reflections is presented first, next the equipment used for these experiments is described, and finally, in chronological order, is a description of the distance and reflection simulations that were performed.

3.4.2 Background Much of this thesis effort was spent in simulating auditory distance and room reflections in order to improve the DIRAD's ability to generate externalized audio images for sources located on the median plane. Several variations of distance and reflection simulations were performed. The following background information presents the basics of human auditory distance perception and how reflections may influence distance perception and externalization.

The first objective of this thesis was to simulate auditory distance and incorporate room reflections using the DIRAD. Distance simulation required a scaling of the source gain setting based on distance. To simulate early reflections, two primary parameters were considered. First, the reflection was to be simulated by positioning a second audio source at the place where the primary sound would theoretically "bounce" off of the wall. This method has been used to simulate reflections using the Convolvotron (58). The other primary consideration was the time delay associated with the longer path length of the reflection. Figure 6 shows some of the possible reflections that can arrive at a given location from the surrounding walls.

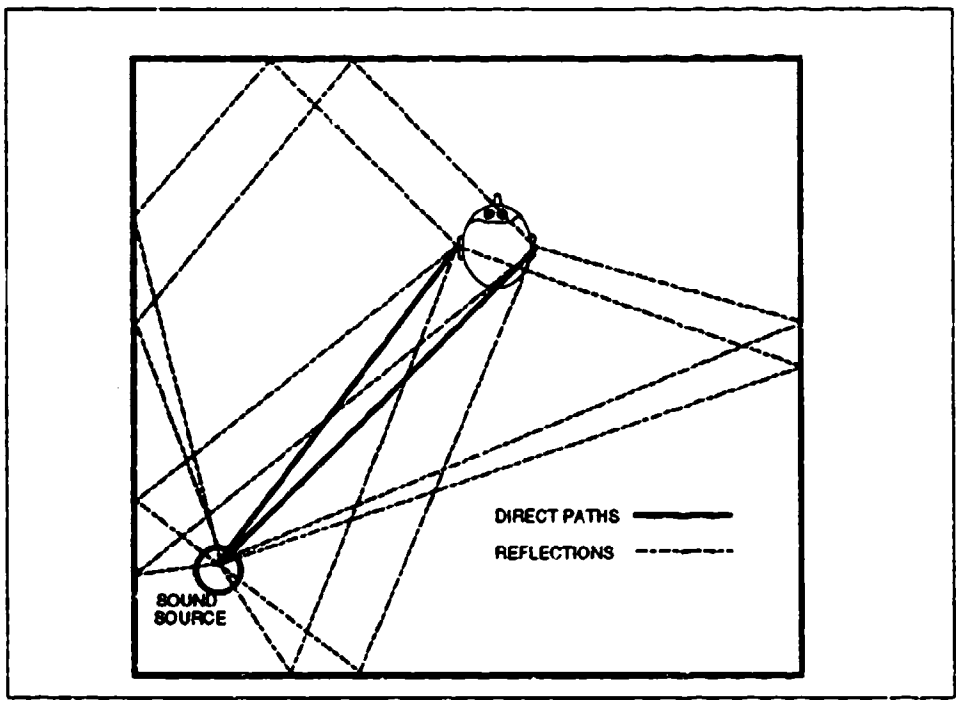


Figure 6. Room Reflections

AL/CFBA provided an PC based acoustical raytracing program that was developed to visualize reflection bounce point positions and incident angle changes, for a given room size and source location. This software was used to calculate the position of the bounce point and the time delay for a given reflection. For the case of first order (hitting only one wall) reflections this program was not really needed, but it did provide a means of very easily seeing the results of moving the source and how room size affected first order reflections. The program provides text based output of all ray lengths and angles, and provides a three dimensional graphics view of the room, source, target and rays. It was a very helpful tool for this work.

A typical room that was simulated is shown in Figure 7. This figure shows the room dimensions, listener location, and a view of one possible location of the source and one associated reflection.

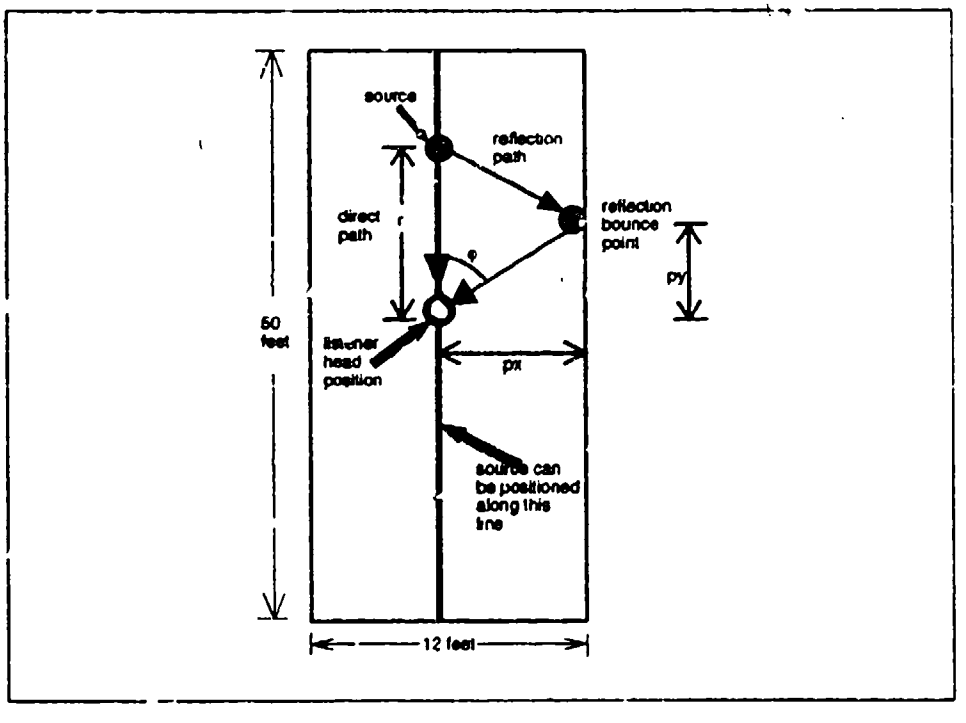


Figure 7. Simulated Room Dimensions

Figures 8 and 9 show, for this specific room, some of the changing parameters associated with the reflection simulation. Figure 8 shows how the difference in path lengths between the primary and the reflection changes with source to listener distance. Figure 9 shows how the incident azimuth angle φ changes as a function of source to listener distance for the given room. It can be seen that the path length difference (and associated time delay) becomes smaller with increasing distance, as does incident angle.

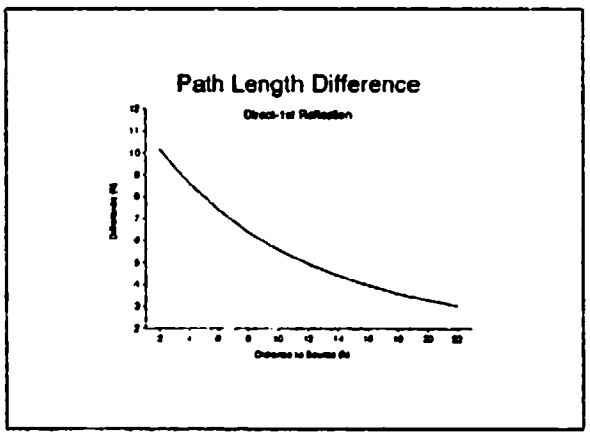


Figure 8. Path Length Difference vs. Distance

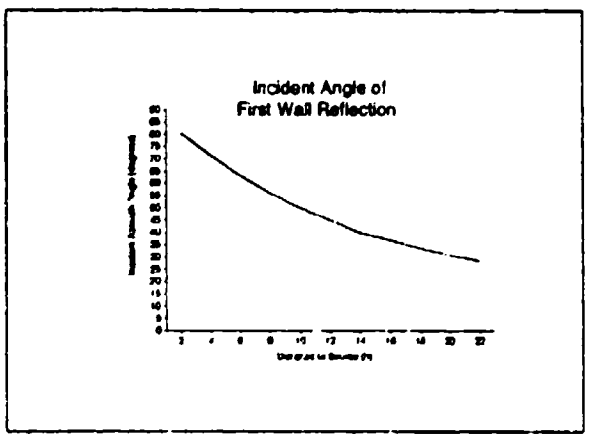


Figure 9. Reflection Incident Angle vs. Distance

The calculations use for finding the position, angle and time delay of the reflection are straightforward as shown below (refer to Figure 7):

$$p_y = (1/2)r \tag{1}$$

$$p_x = w \tag{2}$$

Where p_x is the distance in the lateral direction from the the listener to the wall: ($w = 6$ feet in this example), p_y is the distance that the reflection bounce point is located in front of or behind the listener, and r is the distance from the listener to the direct source as defined in Figure 1. The angle at which the reflection arrives at the center of the listener's head in azimuth is given as

$$\varphi = \tan^{-1}(p_x/p_y) \tag{3}$$

The time delay in seconds is computed with the following equation:

$$\Delta t = ((2 * \sqrt{p_x^2 + p_y^2}) - r)/1125 \tag{4}$$

Where 1125 is the approximate speed of sound in air at 20 degrees Celsius, measured in feet per second (44).

The other parameter that was simulated was the change in sound level due to distance. Sound levels can be expressed in terms of power or amplitude. Intensity is usually used to describe the power level of a sound at a particular place, (intensity is a measure of power flow per unit area), and amplitude is expressed using the root-mean-squared (RMS) value of the sound pressure (44:14- 4,5). Sound pressure and intensity are usually specified in decibels (dB). The intensity level of a sound is given by:

$$L = 10\log(\frac{I}{I_0})dB \tag{5}$$

The reference intensity (I_0) most often used to calculate an intensity level is the standard reference intensity. This standard is an intensity level that is close to the threshold of human hearing for a 1000 Hz tone (44:14-6). If a constant intensity source is measured at some point in a free field, the measured intensity level will drop 6 dB for each doubling of the source to measurement point distance. The equation that governs this is the inverse square law:

$$I = I_s/r^2 \tag{6}$$

In this equation, the intensity at any point I will be equal to the source intensity I_s divided by the distance r squared.

The RMS pressure level of the sound also drops 6 dB with each doubling of distance. The RMS pressure at any point will be equal to the source RMS pressure divided by the distance, this is the inverse first power law. For sound pressure generated by a loudspeaker, the RMS sound pressure level is directly proportional to the source input voltage. If the input voltage is scaled by the inverse of the distance, the RMS pressure will fall off at the required 6 dB for each doubling of distance. To simulate distance for this thesis the following equation will be used:

$$V = \frac{(V_0)(r_0)}{r} \tag{7}$$

Where V is the input voltage corresponding to some arbitrary distance, V_0 is the reference voltage corresponding to r_0, the closest allowable distance (three feet for all simulations), and r is the distance from listener to source. The value of V_0 was set to the maximum output voltage, so that when the source was at the minimum distance, the output gain would be at a maximum.

According to Blauert (4), for sources located in the range of 3 to 15 meters the only observable change in measurements made in the ear canals of subjects is the sound pressure level change (as the listener to source distance was changed).

For distances closer than 3 meters, changes in the spectral content of the ear signals are seen with changes in distance, due to the close proximity of the head and outer ears. Blauert concludes that the spectral changes for close sources are evaluated but that they play a small role in distance evaluation for sources located at distances greater than 25 cm. Blauert's review of distance perception also gives generalized results from several studies that show that a 6 dB drop in signal level is not enough to create the perception of a doubling of source distance. In order for subjects to experience a doubling of a source's distance, a reduction in signal level of about 20dB was required (4:122).

In this thesis, the effects of the changes in the spectral content will be ignored, the source distance is limited to the range of 3 to 24 feet, and the inverse first power law was implemented by scaling the input voltage by one over the distance.

3.4.3 Equipment The workstation employed for all experiments was a Silicon Graphics Personal Iris 4D/35 (a Silicon Graphics Iris 4D/440VGXT was used for one later experiment). This workstation is configured with an audio processor board that employs a Motorola DSP56001 digital signal processor. The audio processor board allows for various standard sample rates from 8 kHz to 48 kHz. The board can be used for stereo or monaural audio, with 8 or 16 bit (stereo) or 24 bit (mono) input resolution and 18 bit analog outputs. The audio inputs consist of an analog microphone jack, left and right line in jack, and digital serial input. The outputs provided are: left and right line out, left and right headphone out, and digital serial out. The audio system on the 4D/35 was used for several different functions in each stage of this thesis.

The other main components of the system are the Polhemus 3SPACE ISOTRACKER, and the DIRAD. The ISOTRACKER source was mounted on a piece of PVC tubing suspended 2 feet above the listener's head. The ISOTRACKER sensor was taped to the center of the headphone band. Two DIRAD (described above)

configurations were used for these experiments: 1) two channel, azimuth only, and 2) two channel azimuth and elevation. The Polhemus head tracker system is fully controlled by the DIRAD. The SG 4D/35 workstation communicates with the DIRAD via an RS-232 line, the pin connections are shown in Figure 10.

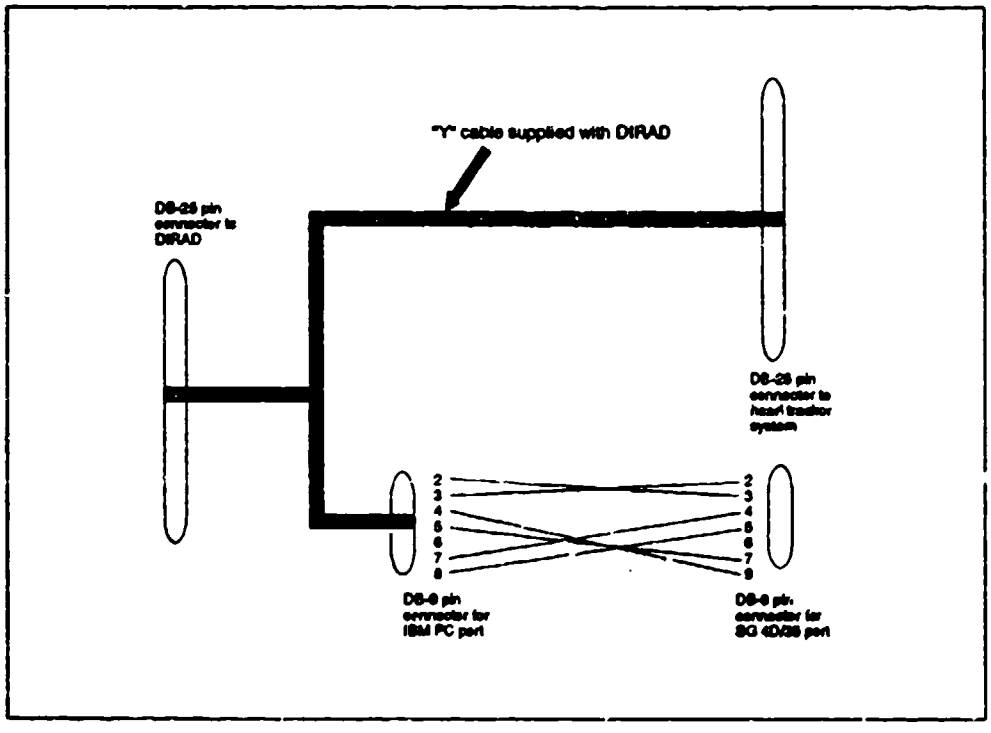


Figure 10. SG 4D/35 to DIRAD RS-232 Pin Out

The DIRAD is configured to respond to a set of commands that are listed in Appendix A. The commands allow for source positioning, head tracker initialization, monitoring of a subject response button, etc. The audio input to the system came from a variety of sources. For initial testing and software development a Sony Walkman Model, MWM-FX30 cassette player was used.

3.4.4 Two Source Control Panel The first requirement, in order to begin work on simulation of reflections and providing visual cues, was to port the control software for the DIRAD to the SG 4D/35. A two source control panel program was

written first to "wring the system out" and troubleshoot any possible hardware and software problems. Figure 11 shows the overall system block diagram.

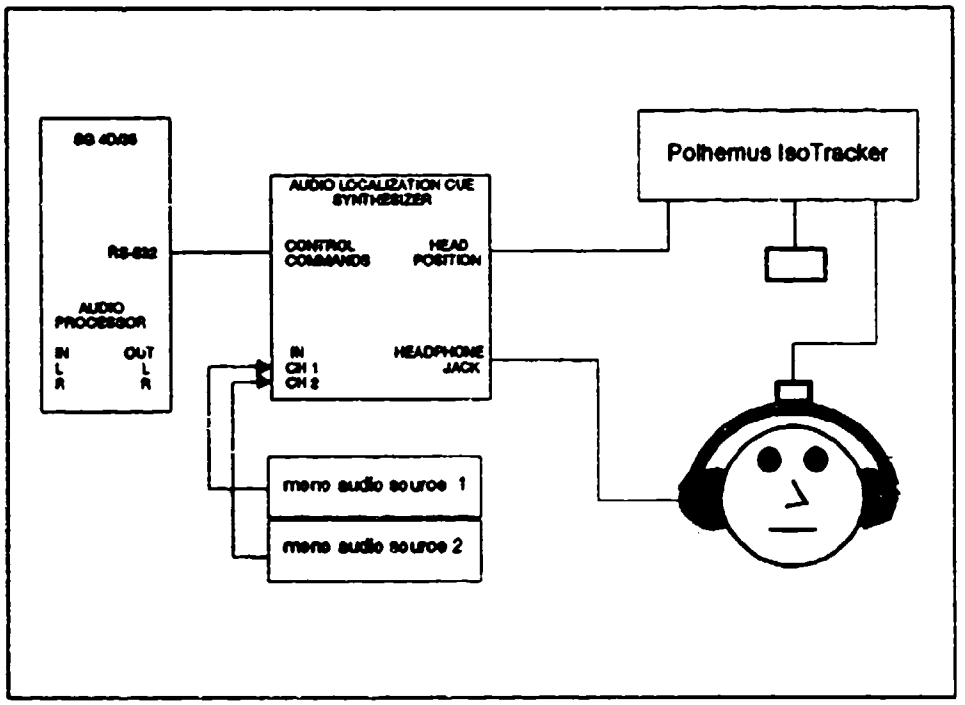


Figure 11. Two Source Control Block Diagram

Software was provided with the DIRAD to control the synthesizer using an IBM compatible personal computer. The routines for the issuing and receiving of commands were written in C++. This software was modified to allow it to run on the SG 4D/35. The majority of the changes that were made to this software were made in the RS-232 port commands and the user feedback commands. The modified Localization Device Class is provided in Appendix B. A general layout of the software structure is shown in Figure 12 for this application. The control panel application has access to the functions of the Localization Device Class. The Localization Device Class relies on the methods of the unmanaged RS-232 Class, described in Captain Simpsons's 1991 AFIT thesis, (49) to perform two way communications with the DIRAD.

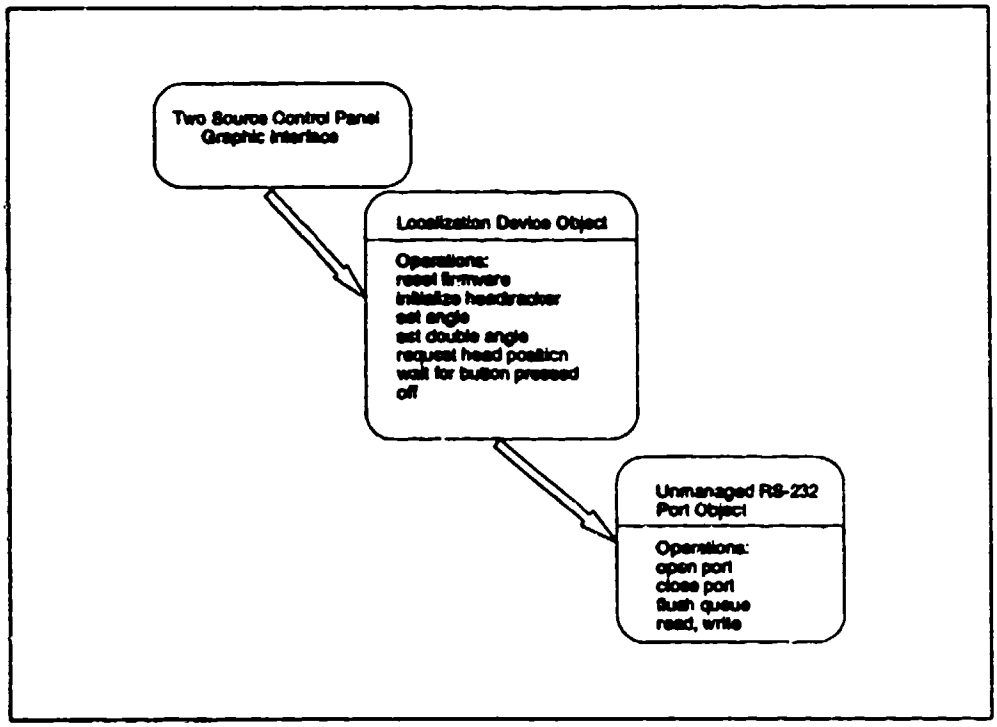


Figure 12. Control Panel Software Structure

AL/CFBA uses a text based menu driven front end program for issuing configuration and positional commands to the synthesizer. It was decided to use the graphics capability of the SG 4D/35 to provide a graphical control panel for this application. As shown in Figure 13, the control panel program was written to be simple to use.

Upon execution, the control panel automatically initialized the DIRAD, and both sources are initially off. The user is presented a window with an icon representing his head, a large circle surrounding his head representing the allowable locations for sound sources, and two small colored circles that represent the two sound sources. A sound source can be made "active" by clicking on its button. The active source can be moved by clicking the left mouse button on or near the desired point. The user can also drag the icon to the proper position. The drag function allows for fine position adjustments. Other options provided by the control panel include:

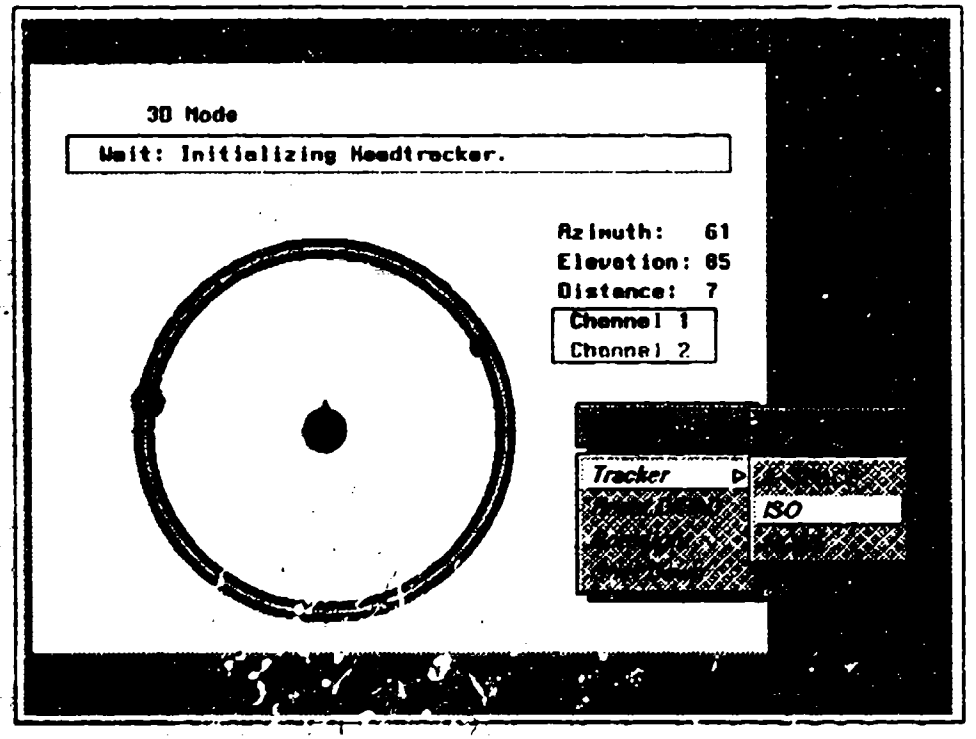


Figure 15. Control Panel Interface

- Top-Line status bar of current synthesizer mode and head tracker type in use.
- Message area providing information on required user actions.
- Text display of coordinates of the selected source.
- Channel selector buttons.
- Pop-up menu selections to choose head tracker, boresight, firmware reset, and DIRAD shut down.

The development of this control panel provided a means of testing the control software that was modified to run on the 4D/35 and also provided an easy to use demonstration of the DIRAD's capabilities.

9.4.5 Distance and Reflection Simulation: Prototype A prototype version of the reflection simulation was written to test out the feasibility of using the 4D/35 to

perform the time delay and output gain control of the audio signal. The hardware configuration for the initial reflection simulations is shown in Figure 14.

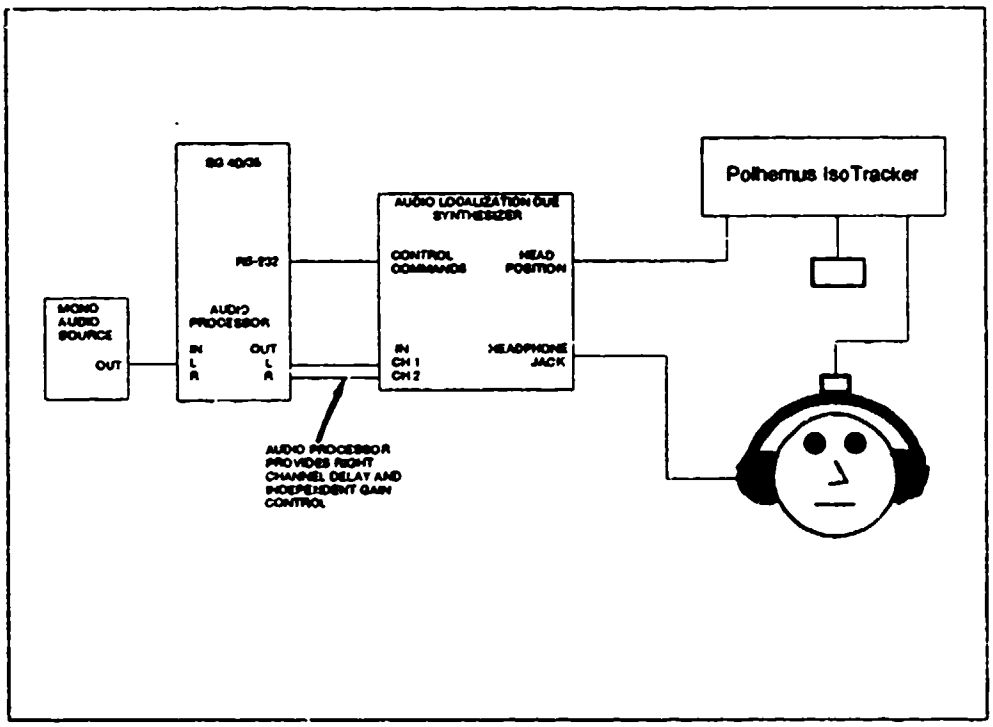


Figure 14. Single Reflection Block Diagram

The simulation program was based on a 12 foot wide by 25 foot long room, with the listener positioned three feet from the back wall and centered between the side walls. The interactive program allowed for the user to position an audio source at any distance from three to 22 feet in front of him. The interface control panel, shown in Figure 15, displays the position of the listener, defines the walls of the room, and shows the location of the direct or primary source, and the location of the reflection point.

The software control panel for this prototype reflection simulation furnished the following options:

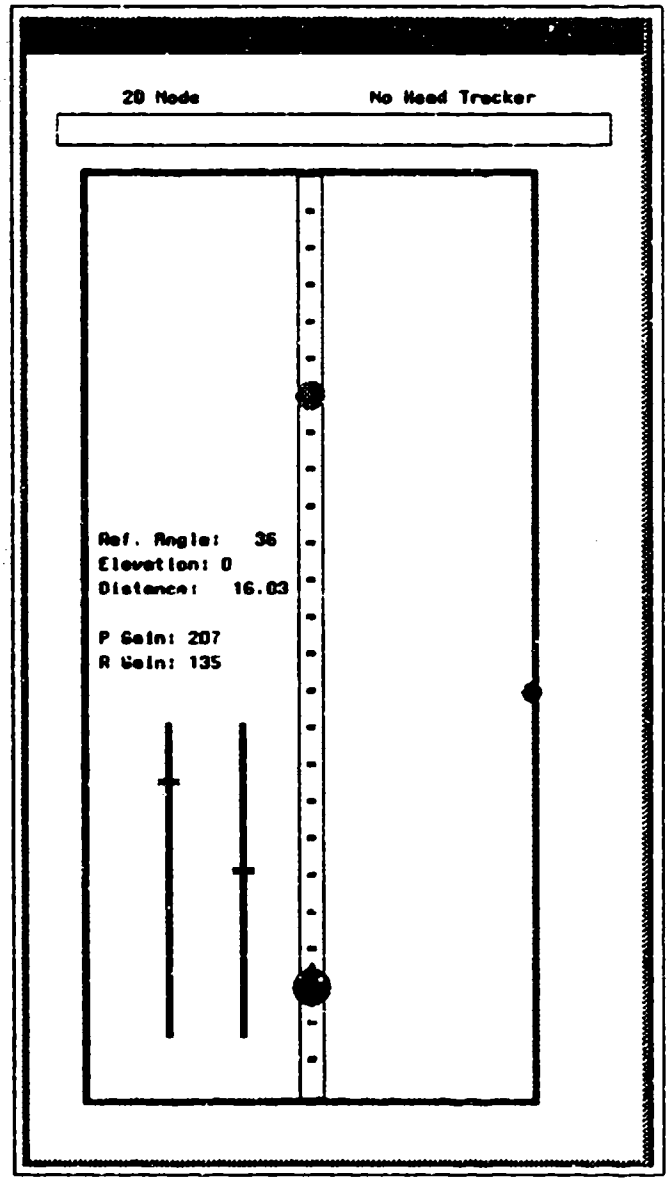


Figure 15. Reflection Simulation Control Panel

- Two slider controls allow for independent control of the output gains of the primary and reflected sources.
- The primary source location is positioned via mouse click or drag to any point along the zero angle azimuth.
- The reflection bounce point on the right wall is calculated by the program and the reflection icon is automatically positioned on the right wall.
- The time delay is also calculated by the program based on the location of the primary source.

The simulation starts by issuing the DIRAD reset command. The primary source is initially positioned three feet in front of the listener. The user can move the source to any location directly in front of him via the mouse. The program sets the output gain of the primary source using the inverse first power law then positions the primary source at 0° azimuth using the first channel of the DIRAD. Once the primary source is positioned, the program sets the output gain of the reflection using the inverse first power law with the total reflection path length as the distance traveled. The reflection audio is delayed in time due to the increased distance that it traveled to arrive at the listener. The reflection is simulated by using the second channel of the DIRAD to position the amplitude scaled, time delayed version of the primary audio at the azimuth angle of the calculated bounce point. The user can then experiment with the gain settings of both the reflection and primary source by using the slider gain controls at the lower left side of the control panel. When the primary source is moved the gain settings are reset to the "proper" level based on the inverse first power law.

The software actually consists of two parallel processes: the control panel interface described above and a separate program that is spawned from the control program to handle the output gain changes and time delay for the reflection. The two processes communicate via a Unix pipe. The Unix pipe is an interprocess com-

munication channel that can be written to or read from by cooperating processes. The control panel sends the output gain settings and time delay value (in number of samples) to the pipe whenever a change has been made. The audio program constantly checks to see if a new set of parameters has been sent and if so, reads them in and makes the required updates. A general layout of the software structure is shown in Figure 16 for this application.

The audio is input to the left analog input of the audio processor board, sampled at 16 kHz, 16 bit resolution. The audio is read into a buffer that holds 500 frames, a frame being a set of four bytes, two for each input channel. The right channel is not used for input although both left and right channels are required for output. For this reason, the audio processor board is used in stereo mode. A 500 frame output buffer is put together using the input, as is, for the left channel and a time delayed version, based on the current time delay value, is used for the right channel.

The software that is used to generate the variable time delay was verified by the following procedure:

1. A test pulse was generated using an HP 8116A Pulse Function Generator. A single burst of a 1 kHz, 90% duty cycle square wave was output every 30 msec.
2. This pulse was input to the left channel of the 4D/35 audio processor board.
3. After software processing, the right channel was to be delayed in time with respect to the left.
4. The left and right outputs from the 4D/35 audio processor board were connected to a B+K Precision Dual Time Base Oscilloscope, Model 1570A for time delay measurements.
5. The time delay was measured for several points and the desired delay was found to be equal to the measured delay.

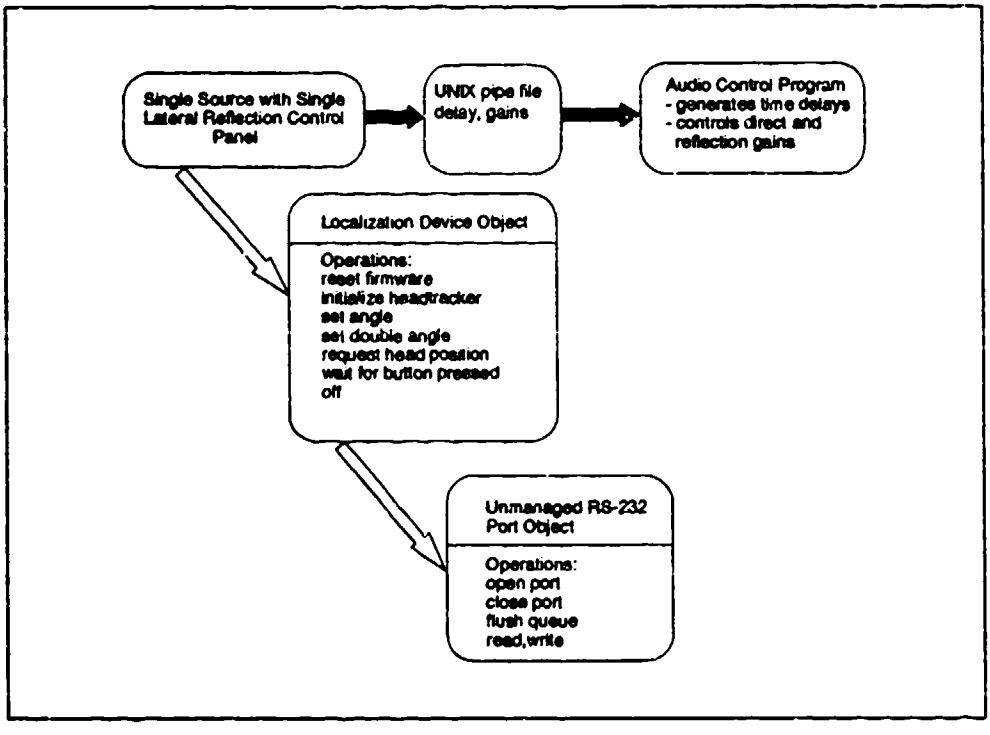


Figure 16. Reflection Simulation Software Structure

The simulation of distance amplitude scaling was verified using the following procedure:

1. A 1 kHz tone was applied to the left audio input of the audio processor board from a Wavetex 20 MHz AM/FM/PM Generator Model 148.
2. Headphones, Sennheiser HD 520, were connected to the output of the DIRAD.
3. The headphones were placed on a Brüel and Kjaer Artificial Ear Type 4152.
4. The output from the Artificial Ear was measured with a Brüel and Kjaer Precision Sound Level Meter Type 2218 and its associated Brüel and Kjaer Octave Band Filter Set Type 1613.
5. The DIRAD main volume control was set to read a convenient level on the Sound Level Meter, 60 dB for instance, for some simulation distance.
6. The distance was then doubled with the simulation software.
7. The 6 dB drop was verified on the Sound Level Meter.
8. This procedure was repeated for several positions using both the primary and reflection sources.

The prototype reflection simulation confirmed the ability to perform real-time control of the time delay and gain settings using the 4D/35's audio processor board.

3.4.6 Distance and Reflection Simulation: Single Lateral Reflection The prototype reflection simulation program was refined to allow for an initial simulation of auditory distance with a single reflection. This program applied the amplitude scaling of the audio output voltages from the audio processor board based on distance. Also included in this version of the simulation was the ability to position the source behind the listener's head (along the $\varphi = 180°$ line) and the ability to use the head tracker. The simulated room size for this experiment was 24 feet wide and 50 feet long. The listener was positioned at the center, so that he had a wall 12 feet on

either side of him and walls 25 feet in front of and behind him. The primary source position could be moved to any distance, but no closer than 3 feet from the listener, within the room on the median plane. The audio level of the primary source was controlled by scaling the output voltage based on the $1/r$ distance equation. The reflection's gain was set by computing the same equation using the reflection path length distance. One other feature that was added at this point was the ability to let the user turn off either the reflection or the primary source by keyboard input.

The control panel is similar to Figure 15, except the listener's head is centered both in room length and width, and the slider gain controls are not used. The program is used like the prototype version. The primary source is positioned with the mouse, and the software generates the audio and positions it with the DIRAD. The "R" key was used to turn off the reflection audio. Clicking on the primary source icon was used to turn the reflection back on.

3.4.7 Distance and Reflection Simulation: Dual Lateral and Ceiling Reflections AL/CFBA supplied a second DIRAD to use in this final version of the reflection simulation. The firmware in this second DIRAD system allowed for two source azimuth and elevation positioning. The system was tested using the previously described two source control panel, which was modified to allow keyboard input to provide changes in elevation. The results of this initial check-out are discussed in chapter IV.

The last version of the distance and reflection program was written to supply an early reflection from both the left and right wall, and through the use of the second DIRAD, a ceiling reflection was simulated. The room size for this program is initially set to 12 feet across, 50 feet long, however the width of the room was made variable. By pressing the "B" key, the distance to the side walls could be increased, and the "S" key made this distance smaller. The simulated distance above ear level to the ceiling is equal to the distance to the side walls so that a single time delay

could be used to generate all three reflections. A software listing for this control panel and the associated audio control program can be found in Appendix C. A block diagram of the equipment used for this experiment is shown in Figure 17.

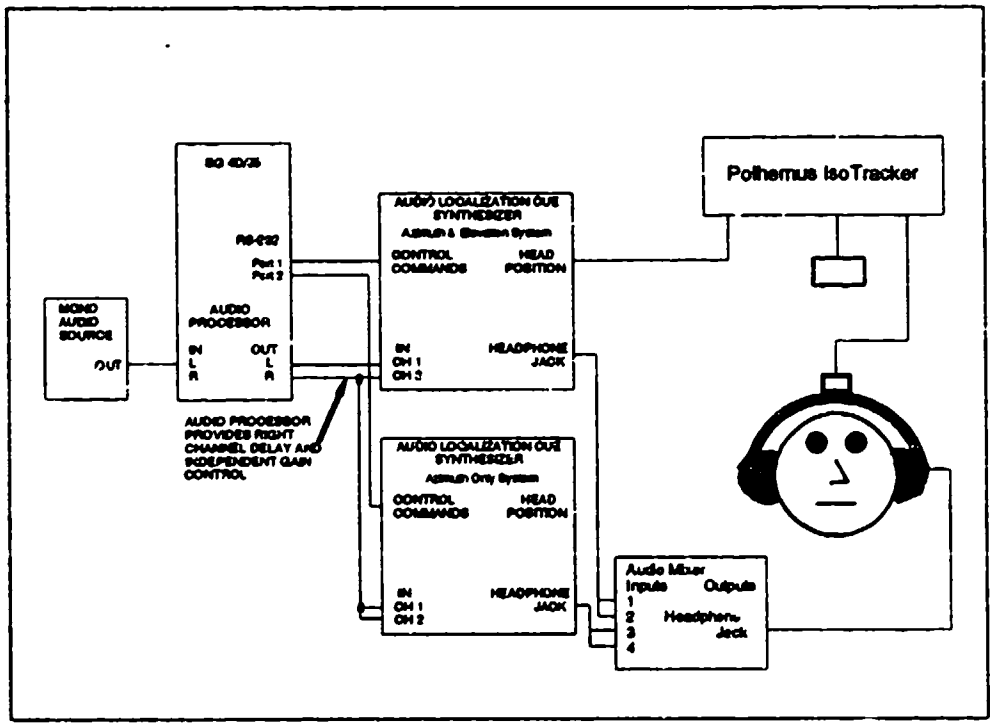


Figure 17. Three Reflection Simulation Block Diagram

The stereo outputs from both DIRAD systems were mixed using a Teac Model 3 Audio Mixer. The same basic control panel interface was used with this version showing the locations of all three simulated reflections. The head tracker could only be connected to one of the DIRAD systems, however, both systems needed to adjust the source locations to compensate for head movements. The solution to this was to connect the head tracker to the azimuth only DIRAD, then continually request head position data from it, using the "track" command, to use to calculate a relative angle to position the second DIRAD's sources.

This set of experiments accomplished the first objective of this thesis, that of incorporating real-time simulated room reflections to enhance the DIRAD's capabil-

ity to simulate distance and improve externalization. The next set of experiments achieves the second objective, the addition of visual cues to the synthesized audio.

3.5 Experiment 3, Incorporation of Visual Cues

3.5.1 Introduction The incorporation of visual cues to accompany the synthesized audio was approached in two ways. The first approach was to present the listener with a three dimensional graphics scene which was presented on the CRT of the 4D/35. The second approach was to display the visual information to the listener using an optical display device (ODD) that has the ability to display certain video images as three dimensional objects which appear to "stand up" on a horizontal viewing plane.

Both methods modeled the listener as a graphics figure in a synthetic environment (this figure representing the listener will be referred to as the *projection* for the rest of this thesis). This approach requires the listener to "become" the projection at the location in the environment. This method is different from a more traditional synthetic environment approach where the viewer is not modeled in the scene but is actually "in" the scene. The methods that were developed to combine synthesized spatial sound with three dimensional graphics objects could easily be used with the more traditional synthetic environment approach.

Modeling the listener in the environment was used in order to test the feasibility of adding spatial sound to the ODD which is being studied at AFIT. The ODD will allow an observer of a battle, for instance, to see their projection at their location on the battlefield from a God's eye perspective.

For these experiments, the location of the projection was fixed. The God's eye view point could be changed from the keyboard. The listener/viewer (the user) would have sensory information provided from two perspectives: his *visual information* would come from the God's eye view and his *auditory information* would be generated as if he were at his projection's location.

Both of the audio/visual experiments described below relied heavily on AFIT graphics software which was developed for a synthetic environment simulation of the Air Force's Red Flag fighter pilot training exercise.

3.5.2 4D/35 CRT Graphics Display The initial test of visual cues was performed by programming a graphics display on the CRT of the 4D/35 that showed the projection, a background scene and a small model of a speaker that rotated around the projection's head. This model was used to develop the programs that calculated and sent the graphic object's position (azimuth angle) relative to the projection to the DIRAD. The object's distance from the projection was also calculated and sent to a process running on the 4D/35's audio processor board which was programmed to provide distance simulation by output gain scaling. A block diagram of this systems is shown in Figure 18.

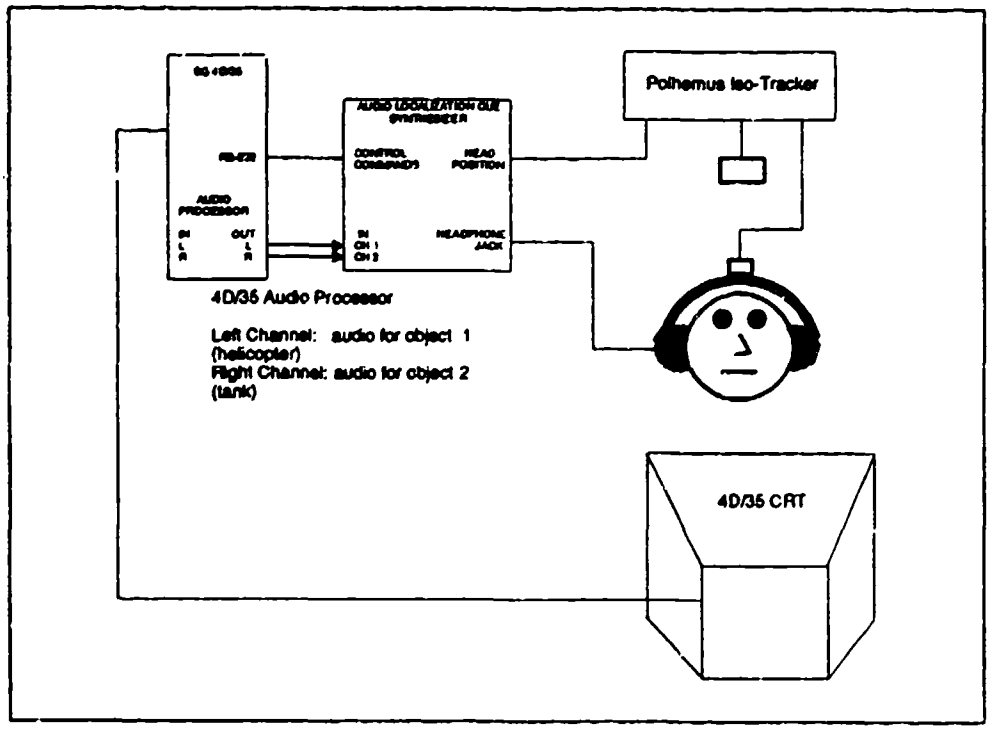


Figure 18. Visual Cues on 4D/35 CRT Block Diagram

The audio for this simulation was provided from stored sound files that were saved as Audio Interchange File Format (AIFF) (8 bit, 22.225 kHz sample rate) on an Apple Macintosh IIci, using SoundEdit^{TM} Pro software. These files were then transferred to the 4D/35 and converted to 8 bit, 16 kHz, two's complement integer data files, using the sfconvert program. This conversion to raw was done so that reading and playing short (about 2.5 seconds) segments of the sound could be programmed more easily (the AIFF file structure would not have to be decomposed to get the data).

By only using a short segment of the audio it was possible to get a "constant" sound to accompany the visual image. This is important because it is not desirable to send an audio input that already contains spatial information to the DIRAD. For example, if a recording of a jet flying away from the listener (decreasing gain with distance) was used to accompany computer generated visual cues, the simulation parameters (movement of the image) would have to be tied to the recording, or conflicting cues would be sent to the user. In contrast, if a somewhat constant jet sound is used, the sound's distance and position can be simulated in real-time to reflect the positional changes of the visual simulation.

The overall process was carried out as follows:

- A short segment of audio was read into a buffer and continuously output to the audio input of the DIRAD.
- The associated graphic image followed a predetermined flight path.
- The distance from the projection to the object was calculated and sent to the audio processor board to scale the output gain based on the inverse first power law.
- The azimuth angle between the projection and the graphic object was calculated and sent to the DIRAD to "position" the sound.

Upon completion of the interface between the graphics program, the DIRAD software and the audio processor software, the prototype was modified to include more realistic images. The moving speaker was replaced with a three dimensional model of an Apache helicopter, and a helicopter sound file provided the audio to the DIRAD. The graphics scene now included the projection, some terrain (ground, sky and mountains), and the Apache. A flight path data file was generated that moved the helicopter in the following manner: It started out behind the distant mountains, flew up over the mountains directly at the projection, once it was at close range it circled completely around the projection, then turned and flew back over the mountains. The flight path data file contains, for each increment of simulation time, the object's position and orientation with respect to the world coordinates.

After this program written, it was obvious that the second audio processor channel could be used to feed the second DIRAD channel and provide simultaneous spatial audio for another visual image. The image that was chosen was a model of an M1 tank. The tank's motion was programmed by a second entry in the flight path data file. The graphics images for this simulation can be seen in Figure 19.

Thus, the final version of this simulation software had two independent visual images coupled with spatial sound via the DIRAD and distance scaling via the audio processor board. Appendix D contains the software for the graphics images and the associated audio program. The angles and distances that were used to generate the audio information were calculated "on-the-fly". This is important because the software is flexible enough to allow for non-scripted motion of objects. In other words, the position of the images could be controlled by the user, (positioned anywhere) and the proper sounds would still be generated. The head tracker was used to couple the listener to the DIRAD, however, the head tracker was not interfaced to the visual display.

Figure 19. Graphics for CRT Display

3.5.3 Visual Cues Using the ODD Visual cues were also provided with the use of a different type of graphics display, the ODD. In order to accomplish this, the above described software was modified for use on the ODD. The ODD is actually a modified Time Traveler^{TM} video game made by SEGA. The three dimensional display relies on a spherical mirror that projects images from a CRT to a horizontal viewing plane (5). The images formed on this viewing plane have a three dimensional appearance, and seem to be standing on the plane. In order to enhance the 3-D effect on the ODD, real-time shadows must be added to all objects in the field of view. Because of limitations in the usable display area of the ODD, this version had only one moving object. Other differences between this simulation and the CRT display are:

1. Graphics software was run on a Silicon Graphics Iris 4D/440VGXT with an internal NTSC video encoder which was required to drive the CRT for the ODD.
2. This simulation had no distance cues generated for two reasons: there is no audio processor on the 4D/440VGXT at AFIT, and the optics of the ODD limited the range of visual distance that could be simulated.
3. Since distance was not feasible, it was decided to use the azimuth and elevation DIRAD system, to test the ability of the graphics to enhance the synthesized elevation cues.

The audio for this display came from the 4D/35 audio processor board. The audio was the helicopter sound file continually played at a constant gain setting. A block diagram of this system is shown in Figure 20.

The user of this display sees the projection located in the center of the viewing plane, with the helicopter image circling at ear level. The flight path for this simulation consisted of:

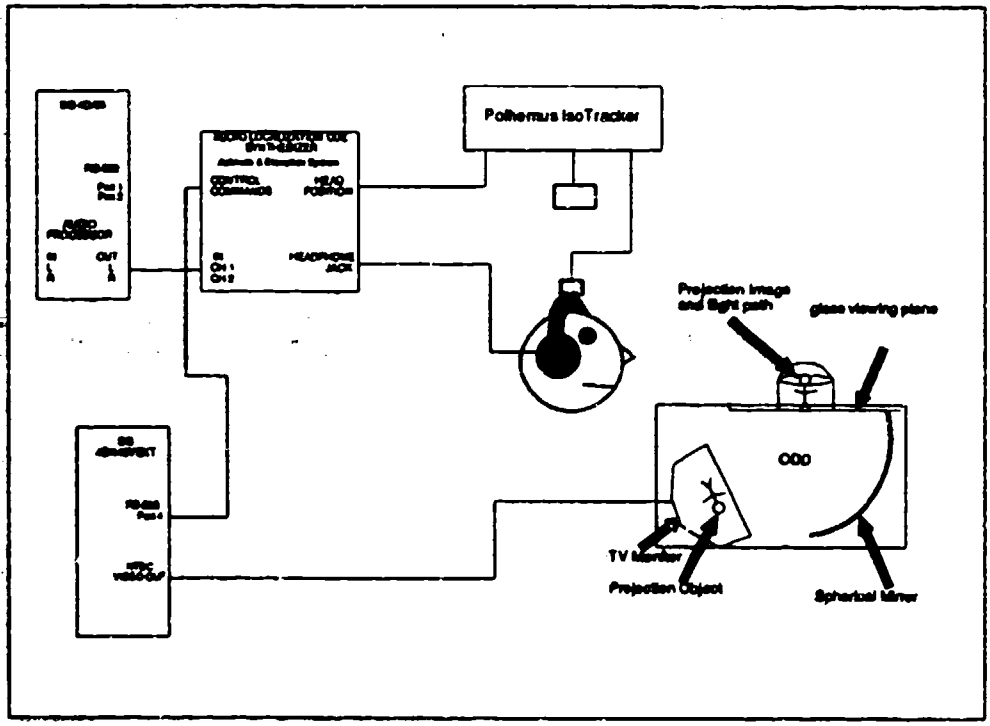


Figure 20. Visual Cues on ODD Block Diagram

- The helicopter started at ear level height on the left side ($\varphi = 270°$)
- Made one complete revolution around the projection's head
- Dropped straight down to ground level
- Flew back up and directly over the head of the projection
- Dropped down to ground on the $\varphi = 90°$ side
- Flew back up and over the projection to the starting point.
- Repeated.

3.6 Summary

This chapter has presented the various methods that were used to achieve the objectives set forth in this thesis, to enhance the DIRAD output with the environmental cues of distance and reflections and to provide visual cues to reinforce the synthesized audio. A background on the AL/CFBA DIRAD system was presented along with a background on human auditory distance perception. The reflection/distance simulations were presented with special emphasis on the generalizations of these programs that make them valuable for future research.

Two specific examples were presented showing how existing AFIT graphics hardware and software may be combined with the DIRAD system to provide real-time visual information to enhance and reinforce spatial audio signals. Also presented in this chapter were the techniques used to make binaural recordings using KEMAR, the DIRAD and the Convolvotron. The next chapter will present and analyze the results of these experiments.

IV. Results and Analysis

4.1 Introduction

The objectives of this thesis, to incorporate certain environmental effects and visual cues with synthesized directional audio, were accomplished. This chapter will describe the findings from the experiments and provide subjective analysis of these findings. The emphasis of this thesis was to explore the technical means of simulating environmental cues and integrating synthetic audio and visual images, rather than perform a detailed human factors analysis. However, in order judge any improvement in spatial perception with these systems, several observational experiments were necessary.

4.2 Experiment 1, Comparative Recordings

The first experiment that was performed was making several binaural recordings. One result of this work is an interactive program that allows for an informal comparison of the three techniques. The recordings that were made provided a way to subjectively compare the use of a manikin in a semi-reverberant space to binaural recordings made from the DIRAD and Convolvotron. No formal study was undertaken to determine the perceptual qualities of the three recordings. The general finding, from listening to the recordings, is that the semi-reverberant space enhances the effect of the sound source originating from a location away from the listener's head. The Convolvotron and the DIRAD were both perceived about the same, that is the sounds were slightly outside the head for lateral positioned sounds and generally inside or very close to the head for front and rear positioned sounds. The synthesizers both tend to elevate sounds in front of the listener. Some of the recorded sounds were never perceived as being located out in front of listener's heads.

4.3 Experiment 2, Environmental Cue Simulation

4.3.1 General DIRAD and Control Panel Results As stated in Chapter III, the DIRAD was used for all experiments after the recordings were made. First, general results will be discussed for the system as used with the initial version of the control panel.

The use of two independent sources was demonstrated in the following manner. The audio input was from the Sony Walkman cassette player with a male voice (TV news) recorded on the left channel, and a different male voice (FM radio news) on the right channel. During the experiment, the subjects could position the talkers independently in any location using the graphical interface. The system provided good localization quality, but the sources, other than being stable with respect the head position, were not located far outside of the head.

Sources located on either side, 90° or 270° azimuth, were generally perceived "further" outside the head, than sources positioned directly at the front or rear. Also, it was found that listening with both channels in use, one talker positioned in front and one positioned in the rear, there was ambiguity of which speaker was in which position when no head motion was allowed. The use of the head tracker system improved the ability to discriminate these front back reversals. It was also noted that the stability of the sound sources with respect to head motion provided by the head tracker increased the perception that the sounds were originating outside of the head.

The second system that AL/CFBA provided was configured to simulate azimuth and elevation for both channels and had head tracking capability. The range of possible elevations was from directly overhead ($\delta = 90°$) to directly below the listener ($\delta = -90°$). The elevation function seemed to be limited to ear level and above for most listeners, although in the case of the author, who listened to the system for literally hours a day, an "adaptation" seems to have occurred. This is discussed in more detail below.

AL/CFBA also found the elevation limited to ear level and above to be the case for most listeners with this version of the firmware. The sound has different tonal qualities at the extreme positions. Several general statements may be made about the use of the elevation function.

1. Elevation was more difficult to perceive for sounds positioned directly in front of or behind ($\varphi = 0°$ or $180°$) the listener.
2. For side positioned sounds, ($\varphi = 90°$ or $270°$) as the elevation setting was increased the perception was that the sound moved up and over the head.
3. Again for side positioned sounds, ($\varphi = 90°$ or $270°$) as the elevation setting was decreased the perception was that the sound moved up (not down) and over the head. And the sound took on a muffled quality.
4. The overall audio output quality of this box was not as good as the azimuth only box. This may be due to the lower spatial resolution of transfer functions used in this system.

The author noticed that an adaptation of sorts occurred with the elevation function. After about a month of working with the system, sounds were begun to be perceived below ear level as the elevation setting was decreased. This type of adaptation has been reported elsewhere for synthesized spatial audio(57:23)(26). Wenzel (57:23) discusses two possible reasons that adaptation may lead to better perceptual results: 1) the sources are simulated as if originating in an anechoic space and this requires familiarity, or 2) because the HRTFs are not individualized (in the case of the DIRAD, the user is listening through KEMAR's pinnae).

4.3.2 Reflection and Distance Simulation Results The main result of the reflection simulation was that it was found that the 4D/35 audio processor board can be used in conjunction with the DIRAD to provide real time dynamic distance and room reflection cues. The secondary results from this work suggest that the use of

distance and room reflections enhance the perception that sources are located at some distance outside of the listener's head.

The audio processor was used to update the gain settings and time delay approximately once every [illegible] milliseconds. The audio signal was sent to the left input channel of the audio processor board, sampled at 16 kHz with 16 bit resolution. A frame consisted of four bytes, two for each channel. Once 500 frames (about 31 milliseconds) were read in, the right channel output buffer was set up to reflect the required delay value and the output gains were updated. This output buffer was then sent out to the DIRAD. The time delay was verified to be correct by the method described in chapter three. This technique allowed for a fairly wide range of distance scaling (3 to 24 feet) and accurate (within 1 sample or 62.5μs). fast time delay generation. Time delays of up to 31 milliseconds can be generated with the software as it is written, however, it could be modified to allow for longer time delays. The maximum of 31 millisecond delay was sufficient for all room sizes that were simulated in this thesis.

The first reflection simulation was used to determine how the addition of a single lateral reflection changed the perception of externalization and distance. A recorded male speaker was used as the source for these presentations. The addition of the single reflection had two main effects, first the position of the sound seemed to be biased towards the reflection, and second, the sound seemed more spacious. Also some listeners, found that by turning off the refiection the sound position abruptly moved in closer to their head. This suggests that the addition of the reflection improved the "externalization" of sounds synthesized by the DIRAD.

The final version of the reflection experiment provided two lateral reflections and a ceiling reflection. The results of interest with this simulation are that it is possible to use more than one DIRAD from one controlling program, and that the use of symmetric reflections served to prevent the position shift of the sound that was seen with the single reflection.

Two problems were encountered with this simulation, 1) the azimuth and elevation DIRAD had different audio output characteristics making it difficult to correctly mix both DIRADs, and 2) the means of providing head position information to the second DIRAD created a time lag in the source positioning. This time lag was due to the way in which the head position information was obtained, processed, and sent to the second DIRAD. For each iteration of the main program loop, the azimuth only DIRAD was polled for the latest head position, this value was returned to the 4D/35 and used to calculate the difference between the source position and the head position angles. This difference angle was then sent to the azimuth and elevation DIRAD. This method seemed to work fine until it was realized that when the listener's head turned quickly, the position of the sources from the azimuth and elevation DIRAD lagged behind listener's head position. When the listener stopped moving his head, the sound sources converged to the correct locations.

Another obvious limitation is that only one time delay can be generated, thus limiting all reflections that are simulated to have equal path lengths. The three reflection simulation used walls (and the ceiling) that were equidistant from the listener, allowing the same time delayed audio to simulate all three reflections. Other areas that were not addressed in this simulation were reverberation and the spectral filtering effects of the reflecting surfaces.

4.4 Experiment 3, Incorporation of Visual Cues

As expected, the use of visual cues provided further reinforcement of the audio cues. The first use of visual cues provided the listener with a screen version of the projection display. The listener was presented a scene of a person in a field with mountains in front of him. This presentation was shown to several people after they had heard the audio portion alone. The general perception was that the visual cues helped to move the sound to the proper location. The visual cues seemed to be a factor in bringing the sound out in front of listeners.

One problem associated with this use of visual cuing is that the audio program as written, only allows for two sounds to be used for the entire simulation. The program should provide the ability to bring in sound files as objects move within some defined range. This feature should be added if this system is to be used in future simulations. Another limitation is that the graphics program should be provided access to the users head potion information, this would allow for the use of head mounted displays.

The use of the ODD provided the means to see if the addition of visual cues could improve the elevation function of the DIRAD. With no visual cues, the audio's location, as it moved along the flight path, was found to be very confusing. This flight path, described in chapter three, moved the object around the listener's head at ear level, then from one side to the other by going directly overhead. The path around the head was easier to discriminate than the path which moved the sound over the head.

With the use of the ODD, the visual cues provided enough of the "ventriloquism effect" to allow the listeners to associate the image with the spatial sound. It was interesting to note that the visual cues were not enough to resolve the below ear level problem discussed previously. The visual image of the helicopter going down provided enough bias to keep the audio from going up; however, the audio did not appear to follow the image down. The audio remained at ear level.

The field of view limitations inherent in the ODD make more realistic simulations difficult at this time. As objects are moved away from the center of the field of view, they become distorted due to the spherical mirror. Another drawback of this system is that the means of controlling audio inputs to the DIRAD has not been worked out. This is due to the fact that the graphics programs were run on AFIT's Silicon Graphics 4D/440 which has no audio capability.

4.5 Summary

The results of this thesis show some of the limitations of current audio localization cue synthesizers and how environmental and visual cues may be incorporated to the DIRAD system. The use of the audio processor proved to be effective and flexible in supplying distance scaling and time delay generation. The reflection experiments suggest that the addition of early reflections can aid in simulating a more externalized audio image, and the use of a simple voltage scaling equation can be used effectively for simulating the effects of distance. The visual cue experiments provided a means of coupling existing AFIT graphics resources with the audio localization cue synthesizer, thus allowing for more realistic simulations where the visual and audio cues can serve to reenforce each other.

V. Conclusions and Recommendations

5.1 Summary

The research effort centered on two main areas, 1) simulation of auditory distance, (via amplitude scaling), and early reflections, and 2) generating visual cues to support the synthesized directional audio from the DIRAD. Also, binaural recordings were made using an acoustic research manikin and two audio localization cue synthesizers to provide a means to appreciate the effects of environmental cues.

Binaural recordings were made using KEMAR in a semi-reverberant environment, the DIRAD, and the Convolvotron synthesizers. The recordings were made using voice and white noise bursts, at eight positions in azimuth. The noise burst portions were digitized using the audio processor board of the 4D/35. A program was written that allows for comparative listening of the three techniques.

Two DIRAD systems were interfaced to graphics based control panels. The first control panel allows for two source positioning, head tracking, and, if used with the azimuth and elevation DIRAD, provides elevation positioning. Three other control panel programs were written that use one channel of the DIRAD as a direct source and the other to simulate an early room reflection. Distance amplitude scaling and the generation of the reflection time delay was simulated with the 4D/35 audio processor board.

The final version of the reflection/distance simulation used both DIRAD systems to generate two side wall reflections, a ceiling reflection, and the primary source. An additional variable in this simulation is that the distance to the walls (or room width) can be changed at any time. Some problems with this system included: the two DIRAD systems were found to have different audio output quality, and the method of supplying head position data to the second DIRAD created a time lag.

Two methods were used to supplement the synthesized audio with visual cues. The first relied on a three dimensional view displayed on the screen of the SG 4D/35. Software was written that uses the position of the graphics objects to generate distance scaled audio, located at the proper spatial location with the use of the DIRAD. The second method used to generate visual cues was an ODD that is being studied for display use at AFIT. The ODD uses a CRT, spherical mirror, and glass viewing plane to create three dimensional projections of images.

5.2 Conclusions

The audio processor board on the 4D/35 proved to be an effective means of preprocessing audio to provide distance amplitude scaling and time delay generation. The resulting audio signals from the preprocessing stage were used as input to the DIRAD systems to simulate various sources and associated reflections. The DIRAD azimuth only configuration provided high quality spatial sound in the horizontal plane. The azimuth and elevation DIRAD had reduced output quality and the 'down' elevation function was not perceived by most listeners. The spatial audio generated by the DIRAD was coupled to two visual displays. The software that was written for this purpose is flexible enough to use with other AFIT visual display research.

5.3 Recommendations for Future Research

The research described in this thesis has led to a number of topics that could be investigated. One of the problems with putting together a system like this is that in order to truly judge its effectiveness, a well designed human factors study must test the perceptual results. The programs created for these experiments can be used to perform perceptual studies in several areas:

1. Distance perception with and without reflections present.

2. Degree of externalization with different gain settings for both the primary and reflection.
3. Localization with and with out reflection to test the bias effect of the reflection.
4. Front to back reversals with and without reflections.
5. What combination of reflections (if any) should be provided when the DIRAD is used to provide audio to a pilot? Should some virtual acoustical room surround the pilot?

Other technical issues to enhance the simulations could also be researched:

1. Implement the filtering effects of the reflecting surfaces. Allow the user to specify the absorption coefficient of the wall material. This could be carried out as a filtering algorithm on the host computer.
2. Provide low-pass filtering for to simulate the attenuation of high frequencies due to propagation through the air.
3. Implement a reverberation algorithm to supplement the early reflections.
4. Create a 'sound class' software interface that will provide the means of calling up appropriate sound files to play when an graphics object is within range, to include azimuth and elevation positioning, and distance scaling.
5. Develop software that will allow the head tracker system to be used for both the visual cues (head mounted displays) and audio cues.
6. Use the methods developed here to implement a similar system on a PC with audio board(s).

One final suggestion pertaining to the development of the DIRAD system is to develop a means of controlling the gain of the synthesized audio channels independently by the host computer. This is obviously a problem that would require

modifications to the DIRAD hardware and firmware. The gain setting of each channel would have to be adjusted before the audio is mixed for output over the headset. This modification would provide great flexibility to the user by allowing distance scaling to be a user defined parameter. This would also allow for the output gain to be used for other than distance, the gains could be changed independently to use for various warnings or to provide other information requiring attention.

Appendix A. *DIRAD Control*

This appendix provides a list of the available DIRAD control commands from the *Auditory Localization Cue Synthesizer (ALCS) User's Manual* (1), with comments on how the commands were used for this research. Appendix B contains the Silicon Graphics version of the Localization Device Class. The Localization Device Class was modified to use with the existing `uRS232port.c` class written for use on AFIT Silicon Graphics systems. Appendix C contains the three reflection control panel program and the associated audio processor program. Appendix D gives an example of the software used to combine the graphics with synthesized audio.

A.1 Available Commands

A.1.1 Host Commands The host commands are those commands that the host computer (the Silicon Graphics Systems) can issue to the DIRAD.

- System Reset - Resets the system software. Used as a pop-up menu selection.
- Zero Queue - Used to define a zero degree reference for the head tracker. This was not implemented.
- Request Angle - Used to request the most recent head position in azimuth and elevation. Causes an interruption in the audio outputs. This was implemented but never used.
- Set Angle - Used to set one of the two channels source position in azimuth and elevation (depending on system configuration). Used in all programs.
- Set Double Angle - Used to set both angles with one command. This command can only be used with azimuth only systems. Used in reflection and visual simulations.

- Tracking/Update - Used to position channel one and receive most current head position information. Allows for fastest possible head position updates with no effect on the DIRAD audio outputs. Used in the three reflection simulation program.
- Retransmit Response - Used to request DIRAD to retransmit its last response. Implemented and used within the Localization Device Class.

A.1.2 Response Commands These are the commands that the DIRAD will send to the host computer in response to Host Commands.

- Response Button - Used to detect when a subject response button was pressed. This button is part of the cabling supplied with the DIRAD. Not implemented.
- Angle Read - This is the response to the Angle Request Command. Implemented but not used.
- Tracking/Update Response - This is the return command for the Tracking/Update Command. This response command was used in the three reflection simulation.
- Retransmit Operation - Used to request host computer to send the last command again. Implemented and used within the Localization Device Class.

Appendix B. *Localization Device Class Code*

This appendix contains a listing of the Localization Device Class.This code is a modified version of the Localization Device Class provided by AL/CFBA that was written for use on PCs.

B.1 Locdev.h

```
#ifndef H_LOCDEV
#define H_LOCDEV
#include "uRS232port.h"
/*************************************************************************
// Class LocalizationDevice
//
// This object class is designed to implement the serial port
// driver commands developed to operate both the speaker
// control hardware and the ALCS synthesizers for two and three
// dimensional sound localization.
//
// Modified to use Unmanaged RS - 232 port handler for AFIT Silicon
// Graphics workstations, not all methods were included
*************************************************************************/
class LocalizationDevice {

        private:
    int       initialized;
    int       mode, port_no;
    boolean         reset;
    boolean         sendcommand(char *command, char length = 0);
    boolean         receivecommand(char *command);
    boolean         repeatcommand(char *command);
    void        delay(int ms = 0);
    Unmanaged_RS232_Port *Loc_Port;
    /*The RS232 port handler*/

    public:
    LocalizationDevice(int mode = 0, int port_no = 1);

    boolean         resetfirmware();
    boolean         initializetracker(int trackertype);
    boolean         angleset(int azimuth, int channel = 1,
          int elevation = 90);
    boolean         doubleangleset(int azimuth1, int azimuth2,
```

```
        int elevation1 = 90, int elevation2 = 90);
    boolean         passthrough(int channel = 1);
    boolean         channeloff(int channel = 1);
    boolean         headposition(int *, int *);
    boolean         track(int, int *, int, int *);
    boolean         off();
    int           bufsize();
    };
```

B.2 Locdev.c

```
/*****************************************************************
// This class contains methods developed by AL / CFBA
// for controlling the DIRAD with a PC.The methods were
// modified to allow AFIT Silicon Graphic workstations to
// control the DIRAD.The Unmanaged RS - 232 class is used
// to interface to the selected port.
// Note: for a complete list of DIRAD commands see the
// Auditory Localization Cue Synthesizer(ALCS) User 's
// Manual and AL / CFBA 's Locdev.cpp code.
//
//Modified by:      Capt Eric L.Scarborough, Summer 1992
*****************************************************************/
extern "C" {
#include <stdlib.h>
#include <ctype.h>
#include <stdio.h>
    }
#include <strings.h>
#include  <unistd.h>
#include "locdev.h"

#define ok 1
#define unsuccessful 0

/*****************************************************************
//LocalizationDevice constructor(int mode int, port_no)
//
//Initializes the serial port.
// Modified to allow for selection of port, this makes
// it possible to use multiple DIRADs from the same program.
// Free Field is set as the default, as is 2 D mode.
// Initialized prevents repetitive tracker initializations.
*****************************************************************/
LocalizationDevice::LocalizationDevice(int format, int port_no)
{
```

```
    if (port_no == 1) {
    Loc_Port = new Unmanaged_RS232_Port(Port::port_one,
                    Port::b9600,
                    Port::raw,
                    Port::terminal);
    }
    if (port_no == 2) {
    Loc_Port = new Unmanaged_RS232_Port(Port: :port_two,
                    Port: :b9600,
                    Port: :raw,
                    Port: :terminal);
    }
    if (port_no == 4) {
    Loc_Port = new Unmanaged_RS232_Port(Port: :port_four,
                    Port: :b9600,
                    Port: :raw,
                    Port: :terminal);
    }
    Loc_Port→Open_Port();
    reset = FALSE;
    mode = format;
    initialized = 2;

}

/*********************************************************************
//LocalizationDevice: :resetfirmware()
//
//Resets the firmware in the ALCS
// This method was added to separate the reset from the
// constructor so that messages can be posted into the
// open window.THIS MUST BE CALLED BEFORE USING ANY OTHER
// METHOD
*********************************************************************/
boolean LocalizationDevice: :resetfirmware()
{
    /*Resets DIRAD firmaware*/
    sendcommand("R");
    sleep(1);
    reset = TRUE;
    return (ok);
}

/*********************************************************************
//LocalizationDevice: :initializetracker(int trackertype, int mode)
//
//Initializes the headtracker to run in one of three modes:
```

```
//0:3 Space parallel headtracker
// 1:Isotracker serial tracker
// 2:No tracker - free field mode
//
//The default on startup is Free Field mode.
// This command sets the mode and the headtracker type.
// If a headtracker is selected the synthesizer mode is picked
// in 2 d or 3 d(mode = 2, mode = 3).
//The correct roms MUST BE PRESENT !
**********************************************************************/

boolean LocalizationDevice::initializetracker(int trackertype)
{
   int count;

   if (initialized == trackertype && mode == 2)
      return (1);

   if (trackertype == 2) {
      /*No tracker*/
         if (mode == 3)
         return (sendcommand("F\x03"));
      /*3 D Free field mode*/
         else
         return (sendcommand("F\x02"));
      /*2 D Free field mode*/
   }
   if (trackertype == 0)
      /*THREE - SPACE TRACKER*/

         if (sendcommand("LP") == unsuccessful)
         return (unsuccessful);
      else
         initialized = 0;

   if (trackertype == 1)
      /*ISOTRACKER*/

         if (sendcommand("LS") == unsuccessful)
         return (unsuccessful);
      else
         initialized = 1;

   cout.flush();
   cout << "Please Wait, Initializing Headtracker.\n";
   cout.flush();
```

```
    for (count = 15; count > 0; count--) {
       printf("%d seconds remaining     \n", count);
       sleep(1);
    }

    printf("\n");
    if (mode == 2)
       return (sendcommand("H\x02"));
    /*2 D synthesizer*/
       else if (mode == 3)
       return (sendcommand("H\x03"));
    /*3 D synthesizer*/

}
/*************************************************************************
//LocalizationDevice: :angleset(int azimuth, int channel,
//                int elevation)
//
//Sets one or both channels to a certain azimuth and elevation.
// Channel can be 1, 2 or 3, where 3 sets both channels.
// Note that 360 and 361 are valid angles.These are used for
//off and passthrough for synthesizer roms dated 7 / 1 / 91 or later.
// Use with other roms or the speaker control hardware will yield
// unpredictable results.
//
//The channel defaults to 1 and the elevation defaults to 0
*************************************************************************/
boolean LocalizationDevice. :angleset(int azimuth, int channel,
                int elevation)
{
   char command[10];

   if (azimuth < 0 || azimuth > 361) {
      printf("Azimuth is Out of Range, Press Any Key to Continue");
      getchar();
      return (unsuccessful);
   }
   if (elevation < 0 || elevation > 179) {
      printf("Elevation is Out of Range, Press Any Key to Continue");
      getchar();
      return (unsuccessful);
   }
   if (channel < 1 || channel > 3) {
      printf("Improper Channel Selection, Press Any Key to Continue");
      getchar();
      return (unsuccessful);
   }
```

```
    if (channel == 1)
        /*Channel 1*/
        command[0] = 'A';
    else if (channel == 2)
        /*Channel 2*/
        command[0] = 'B';
    else
        /*Both channels*/
        command[0] = '2';
            command[1] = azimuth / 256;
            command[2] = azimuth % 256;
            command[3] = elevation;

    if (channel == 3) {
    /*Both channels*/

        command[4] = azimuth / 256;
        command[5] = azimuth % 256;
        command[6] = elevation;
        return (sendcommand(command, 7));
    } else
        return (sendcommand(command, 4));

}
/*************************************************************************
//LocalizationDevice: :doubleangleset(int azimuth1, int azimuth2,
//                        int elevation1, int elevation2)
//
// Sets both channels to independent angles.
// Elevation are optional and default to zero.
*************************************************************************/
boolean LocalizationDevice::doubleangleset
    (int azimuth1, int azimuth2,int elevation1, int elevation2)
{
    char command[10];

    if (azimuth1 < 0 || azimuth2 < 0 ||
            azimuth1 > 361 || azimuth2 > 361) {
        printf("Azimuth is Out of Range, Press Any Key to Continue");
        getchar();
        return (unsuccessful);
    }
    if (elevation1 < 0 || elevation2 < 0 || elevation1 > 179
        || elevation2 > 179) {
        printf("Elevation is Out of Range, Press Any Key to Continue");
        getchar();
        return (unsuccessful);
```

```
    }
    command[0] = '2';
    /*Both channels*/

    command[1] = azimuth1 / 256;
    command[2] = azimuth1 % 256;
    command[3] = elevation1;
    command[4] = azimuth2 / 256;
    command[5] = azimuth2 % 256;
    command[6] = elevation2;

    return (sendcommand(command, 7));
}

/*****************************************************************
//LocalizationDevice: :passthrough()
//
//Sets a sound channel to passthrough in synthesizer mode.
// The channel can 1, 2, or 3, where 3 sets both channels.
// Requires a synthesizer with roms dated after 7 / 1 / 91, as it
// works by implementing angle 361. Otherwise unpredictable results
// will occur.
*****************************************************************/
boolean LocalizationDevice: :passthrough(int channel)
{

    return (angleset(361, channel));
}

/*****************************************************************
//LocalizationDevice: :channeloff(int channel)
//
//Turns off a sound channel in synthesizer mode.
// The channel can be 1, 2, or 3, where 3 turns off both channels.
// Requires a synthesizer with ROMS dated after 7 / 1 / 91, as it
// works by implementing angle 360. Otherwise unpredictable
// results will occur.
*****************************************************************/
boolean LocalizationDevice: :channeloff(int channel)
{

    return (angleset(360, channel));

}

/*****************************************************************
//boolean LocalizationDevice: :headposition(int *, int *);
```

```
//
// Queries the device for the latest head position in azimuth and
// elevation and returns the specified values.If a location for
// elevation is not sent the value will be ignored.Note that the
// variables are passed as C++ specific references.
// Do not pass an elevation value initialized to - 1 or no value will
// be returned.
// Note: causes an interuption of the sound field, must be
// reset with an angle set command.
*************************************************************************/
boolean LocalizationDevice: :headposition(int *azimuth,
                    int *elevation) {
      char          response[10];
      int           az, el;
      Loc_Port→Flush_Queue;
      /*Clear out buffer*/

      if (sendcommand("Q") == unsuccessful)
         return (unsuccessful);
      delay(30);
      if (receivecommand(response) == unsuccessful) {
         printf("unsuccessful receive\n");
         return (unsuccessful);
      }
      if (response[1] == 'r') {
         repeatcommand(response);
      }
      delay(100);
      if (response[1] ≠ 'q') {
         printf("Angle Response Not Sent, Press any Key to Continue...");
         getchar();
         return (unsuccessful);
      }
      az = (unsigned char) response[2] * 256 +
          (unsigned char) response[3];
      *azimuth = az;

      if (*elevation ≠ -1) {
         el = (unsigned char) response[4];
         * elevation = el;
      }
      return (ok);
}

/*************************************************************************
//Success LocalizationDevice: :track(int azimuth, int &headazimuth,
//              int elevation, int &headelevation);
```

```
//
// Sets the angle to a new position and returns the current
// head position.Does not work unless an angle has already
// been set by the angleset command.Works only for channel
// one. 3D mode only enabled when 3D synthesizer has been
// activated.
*************************************************************************/
boolean LocalizationDevice: :tracl:(int azimuth, int *headazimuth,
                int elevation, int *headelevation) {
   char          command[4];
   char          response[6];
   int           az, el;

   command[0] = 'T';
   command[1] = azimuth / 256;
   command[2] = azimuth % 256;

   if (elevation ≠ -1)
      /*Check for 3D mode*/
      command[3] = elevation;
   else
      command[3] = 0;

   if (sendcommand(command, 4) == unsuccessful)
      return (unsuccessful);

   delay(8);

   if (receivecommand(response) == unsuccessful)
      return (unsuccessful);

   if (response[1] == 'r') {
      /*Button Pressed*/
         return (unsuccessful);
   }
   if (response[1] ≠ 't') {
    printf("Improper Response Received: Tracking Response Not Sent Press
any Key to Continue...\n");
      getchar();
      return (unsuccessful);
   }
   az = (int) ((unsigned char) response[2] * 256 +
         (unsigned char) response[3]);
   *headazimuth = az;

   if (elevation ≠ -1)
      el = (unsigned char) (response[4]);
```

```
    *headelevation = el;
    return (ok);
}

/*************************************************************************
//boolean LocalizationDevice: :off()
//
// This command turns off channels one and two(if in 2 d mode) and
// makes a dummy head position read to turn off the headtracker and
// sound field.
*************************************************************************/
boolean LocalizationDevice: :off()
{
    int             dummyaz, dummyel;

    if (mode == 2) {
        channeloff(3);
        delay(15);
    }
    Loc_Port→Close_Port();
    return (headposition(&dummyaz, &dummyel));
}

/*************************************************************************
//delay loop(makes about a 10 ms delay)
*************************************************************************/
void            LocalizationDevice::delay(int tens_of_ms)
{
    int             i, j;
    float           a = 99., b = 99., c = 0;

    for (i = 0; i < tens_of_ms * 16; i++) {
        printf("delay: %i %i\n", i, tens_of_ms);
        for (j = 0; j < 1000; j++) {
            a = b * b * c * c;
        }
    }
}
/*************************************************************************
//boolean LocalizationDevice: :sendcommand(char *command, int length)
//
//Sends a command string to the device through the serial port.
// This command automatically appends bytecount and CRC to the
// command string so length only needs to be specified when there
// are zeroes('\x00') present.Note that at least two milliseconds
// of delay are required between commands.
*************************************************************************/
```

```
boolean LocalizationDevice: :sendcommand(char *command, char length)
{   /* Send a command string */
   int count;
   char checksum = 0, *cksm, *len, *thischar;


   if (length == 0)
      length = (strlen(command) + 2);
   else
      length += 2;

   len = &length;

   Loc_Port→Write_To_Port(len, 1);

   checksum = length;

   for (count = 0; count < length - 2; count++) {

         /* Send main body of
          * command */

      thischar = &command[count];
      Loc_Port→Write_To_Port(thischar, 1);
      checksum = checksum ^ command[count];
   }
   cksm = &checksum;

   /* Send checksum terminator */
   Loc_Port→Write_To_Port(cksm, 1);
   return (ok);
}
/*************** ************************************************
//boolean LocalizationDevice: :receivecommand(char *response)
//
//Receives a command from the serial port.The response string
// is returned with bytecount and CRC.
*****************************************************************/
boolean LocalizationDevice: :receivecommand(char *response)
{
   char        *temppointer;
   int         count, i = 1;
   int         checksum = 0;

   temppointer = response;
   temppointer[0] = 0;
```

```
    Loc_Port→Flush_Queue;
    do {
       printf("%i\n", i);
       if (i ≥ 2000) {
          return (unsuccessful);
       }
       Loc_Port→Read_From_Port(temppointer, 1);
       i++;
    } while (*temppointer == 0);
    checksum = checksum ^ *temppointer;
    temppointer++;

    for (count = 1; count < response[0]; count++) {
          Loc_Port→Read_From_Port(temppointer, 1);
       checksum = checksum ^ *temppointer;
       temppointer++;
    }

    if (checksum) {
       printf("Checksum Error, No Receive Done");
       return (unsuccessful);
    }
    return (ok);

}
/*****************************************************************
//boolean LocalizationDevice: :repeatcommand(char *command);
//
//Returns a copy of the last command sent by the ALCS if
//working properly.
*****************************************************************/
boolean LocalizationDevice: :repeatcommand(char *command) {

   Loc_Port→Flush_Queue();
   /*Clear out buffer*/
      if (sendcommand("X") == unsuccessful)
      return (unsuccessful);
   delay(10);

   if (receivecommand(command) == unsuccessful)
      return (unsuccessful);

   return (ok);
}
```

Appendix C. *Three Reflection Simulation Code*

This appendix contains a listing of the programs that were written to perform the reflection and distance simulations. The first program is the control panel interface, note that this program shows how two DIRADs can be used from one program. The second program is spawned from the control panel and controls the audio processing. This program generates the time delay and provides distance simulation by control of the output gains.

C.1 Three Reflection Simulation Control Program

```
/*************************************************************************
*
*    This program is a control panel that allows for control of
* the DIRAD audio localization cue synthesizer. It is specificaliy
* designed to show how environmental cues, (room reflections) may
* enhance the pcception  of auditory depth or distance.
*      It is written for use on an Silicon Graphics 4D/35 with an
* audio processor board installed.
* Note: this version will run 2 DIRADs for multiple reflections:
*    left and right walls and ceiling.
*    DATE:    2 SEP 92
*    AUTHOR:  Eric L. Scarborough
*************************************************************************/


extern "C" {
#include <gl.h>
#include <device.h>
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>
}
#include "locdev.h"
#include <strings.h>

#define X      0
#define Y      1
```

```
#define CH1    0
#define CH2    1
#define CH3    2
#define CH4    3

/* GLOBAL VARIABLES */

/*system error variables */
extern int errno;
extern char *sys_errlist[];

/*mouse variables*/
long org[2],wmval[2];
short mval[2];

/*DIRAD variables*/
LocalizationDevice *ld1,*ld2;
int mode = 2,tracker = 1,curtracker;
double angle1 = 0.0, ceiling_ang = 0.0,oldangle1;
double rwall_ang,lwall_ang;

/*global program variables */
char string[10],buffer[22];
float distance = 1.0;
int elevation = 0,sendit=0;
int active_chan = CH1,done=FALSE;
boolean mov=TRUE;
int pos[4][4];
int pix_to_wall = 90,wall[2];
int pix_to_p = 30;
int head_y = 425;

/*audio control variables*/
int p_vol=128,r_vol=128,delay=0;

/*functions called by main*/
void writemessage(char *message);
void erasemessage();
void makeframe();
void movsources(long my);
void makefigure();
void calc_delay_vol_angle();

main()
{

   short val;
```

```
char strng[25],zer[1];
ld1 = new LocalizationDevice(3,1);
ld2 = new LocalizationDevice(2,2);
int head_az,head_el;
int pfiles[2];
int leftmouse_down=0,exerr=0;
int olddone, olddelay, oldp_vol, oldr_vol;
int who = 1,changed_params=0;
int counter=0,toggle=0;
Device dev,mdev[2];


pos[CH1][X]=230;        /*primary, on elbox ch1*/
pos[CH1][Y]=head_y+45;

pos[CH2][X]=230;        /*ceiling refl, on elbox ch2*/
pos[CH2][Y]=head_y+22;

pos[CH3][X]=320;    /*right wall refl, az box ch1*/
pos[CH3][Y]=head_y+22;

pos[CH4][X]=140;    /*left wall refl, el box ch2*/
pos[CH4][Y]=head_y+22;

wall[0]=135;
wall[1]=325;
olddone = done; olddelay = delay;
   oldp_vol = p_vol; oldr_vol = r_vol;

/* set up the window */
prefsize(500,900);
winopen("Synthetic Reflection Control Panel (SRCP)");
doublebuffer();
gconfig();
frontbuffer(TRUE);
makeframe();

/* these devices will be queued for input */
qdevice(LEFTMOUSE);
qdevice(MOUSEX);
qdevice(MOUSEY);
qdevice(ESCKEY);
qdevice(RKEY);
qdevice(PKEY);
qdevice(JKEY);
qdevice(BKEY);
qdevice(SKEY);
```

```
/*draw the control panel*/
makefigure();

/*position source icons*/
color(GREEN);
circf(pos[CH1][X],pos[CH1][Y],12);
color(MAGENTA);
circf(pos[CH2][X],pos[CH2][Y],8);
circf(pos[CH3][X],pos[CH3][Y],8);
circf(pos[CH4][X],pos[CH4][Y],8);

/*initialize the DIRADs*/
writemessage("      Wait: Resetting Synthesizer Software.");
/*elevation box, ch1 primary, ch2 ceiling*/
ld1->resetfirmware();
/*azimuth box,walls, ch1 right, ch2 left*/
ld2->resetfirmware();   erasemessage();
writemessage("      Wait: Initializing Headtracker Mode");
sleep(2);
ld1->initializetracker(2);
ld2->initializetracker(1);
erasemessage();

/* start audio interface program*/
/* set up a pipe for interprocess communications*/
zer[0] = '0';
exerr = pipe(pfiles);

if (exerr == -1) {
        printf ("\tpipe failed\n");
        printf ("\t%s\n",sys_errlist[errno]);
        exit (127);
    }

/*write the initial settings to the pipe*/
calc_delay_vol_angle();
sprintf(strng, "%3i\t%3i\t%3i\t%3i\t",done,delay,p_vol,r_vol);
printf("bigroom: done: %i delay: %i p_vol: %i r_vol: %i\n"
   ,done,delay,p_vol,r_vol);
write(pfiles[1],strng,(int)strlen(strng));

/*fork the program that plays the audio*/
who = fork();
if (who == 0) {
   exerr = execlp("ref_delay","ref_delay",zer);
   if (exerr == -1) {
```

```
            printf("exec failed\n");
            printf("\t%s\n",sys_errlist[errno]);
            exit (127);
        }
}
printf("%i %i\n",pfiles[0],pfiles[1]);
close(pfiles[0]);
frontbuffer(FALSE);

/*set initial mouse position settings*/
mdev[X]=MOUSEX;
mdev[Y]=MOUSEY;
mval[X]=0;
mval[Y]=0;
getdev(2, mdev, mval);
wmval[X] = mval[X] - org[X];
wmval[Y] = mval[Y] - org[X];
unqdevice(MOUSEX);
unqdevice(MOUSEY);
qreset();

/*set the initial angles on the DIRADs*/
ld1→angleset(0,1,90);
ld1→angleset(0,2,(int)ceiling_ang);

ld2→doubleangleset((int)rwall_ang,(int)lwall_ang,0,0);


while(!done) {
        /* request headposition from az only DIRAD
        // in order to calculate a relative angle for
        // the az & el DIRAD, so it can compensate
        // for head movements also. (this causes a lag
        // in the az & el positioning for fast head
        // motions.*/

        if(ld2→track((int)rwall_ang,&head_az,90,&head_el)){
            if(pos[CH1][Y] < head_y){
                if(head_az≤180){
                    angle1=180-head_az;}
                else if(head_az>180){
                    angle1=180+360-head_az;}
            }
            else
                {angle1=360-head_az;
            }
            if(angle1 ≠ oldangle1){
```

```
            oldangle1 = angle1;
            ld1→angleset((int)angle1,2,
               (int)ceiling_ang);
            ld1→angleset((int)angle1,1,90);
         }
      }
   /* check for user input*/
   if(qtest()){
   switch (dev=qread(&val)) {
      case REDRAW:
         makeframe();
         break;
      case ESCKEY:
         done=TRUE;
         break;
      case RKEY:
            /*reflection off*/
         r_vol=0;
         break;
      case PKEY:
            /*direct off*/
         p_vol=0;
         break;
      case JKEY:
         /* reset gains*/
         calc_delay_vol_angle();
         break;
      case BKEY:
         /* make room wider*/
         if(wall[0]>10){
            wall[0]=wall[0]-1;
            wall[1]=wall[1]+1;
            pix_to_wall=((wall[1]-5)-(wall[0]+5))/2;
            mov=TRUE;
            movsources(pos[CH1][Y]);
            makefigure();
            /*redraw source icons*/
            color(GREEN);
            circf(pos[CH1][X],pos[CH1][Y],12);
            color(MAGENTA);
            circf(pos[CH2][X],pos[CH2][Y],8);
            circf(pos[CH3][X],pos[CH3][Y],8);
            circf(pos[CH4][X],pos[CH4][Y],8);
            calc_delay_vol_angle();
            swapbuffers();
         }
         break;
```

```
case SKEY:
   /* make room narrower*/
   if(wall[0]<200){
      wall[0]=wall[0]+1;
      wall[1]=wall[1]-1;
      mov=TRUE;
      pix_to_wall=((wall[1]-5)-(wall[0]+5))/2;
      movsources(pos[CH1][Y]);
      makefigure();
      /*redraw source icons*/
      color(GREEN);
      circf(pos[CH1][X],pos[CH1][Y],12);
      color(MAGENTA);
      circf(pos[CH2][X],pos[CH2][Y],8);
      circf(pos[CH3][X],pos[CH3][Y],8);
      circf(pos[CH4][X],pos[CH4][Y],8);
      calc_delay_vol_angle();
      swapbuffers();
   }
   break;
case LEFTMOUSE:
   leftmouse_down = val;

   if (leftmouse_down) {
      /*move sources to desired locations*/
      getdev(2, mdev, mval);
      wmval[X] = mval[X] - org[X];
      wmval[Y] = mval[Y] - org[Y];
      if(wmval[Y]<802&&wmval[Y]>47) {
         movsources(wmval[Y]);

         /*redraw figure*/
         makefigure();

         /*calculate new settings*/
         calc_delay_vol_angle();
         if(pos[CH1][Y] < head_y){
            lwall_ang=rwall_ang+180;
            rwall_ang=180.0-rwall_ang;
         }
         if(mov){
            ld2->doubleangleset(
            (int)rwall_ang,
               (int)lwall_ang,0,0);

         }
```

```
                /*redraw source icons*/
                color(GREEN);
                circf(pos[CH1][X],pos[CH1][Y],12);
                color(MAGENTA);
                circf(pos[CH2][X],pos[CH2][Y],8);
                circf(pos[CH3][X],pos[CH3][Y],8);
                circf(pos[CH4][X],pos[CH4][Y],8);
                swapbuffers();
            }
        }
        qreset();
        break;


    }/* switch*/

}

/* test to see if any of the shared variables have changed*/
if(done ≠ olddone || delay ≠ olddelay ||
     p_vol ≠ oldp_vol || r_vol ≠ oldr_vol ) changed_params = 1;

   olddone = done; olddelay = delay;
   oldp_vol = p_vol; oldr_vol = r_vol;

  /* if shared varibles have changed send to pipe*/
if(changed_params){
   printf("bigroom:done: %i delay: %i p_vol: %i r_vol: %i\n"
      ,0,delay,p_vol,r_vol);
   sprintf(strng, "%3i\t%3i\t%3i\t%3i\t",0,delay,p_vol,r_vol);
   write(pfiles[1],strng,(int)strlen(strng));}
changed_params = 0;
}/*while*/

/* send out done to pipe to kill audio program*/
sprintf(strng, "%3i\t%3i\t%3i\t%3i",1,delay,p_vol,r_vol);
write(pfiles[1],strng,(int)strlen(strng));
printf("Exiting...\n\n");
cout.flush();
exit(1);
}/*main*/

/*writes a message in the box at the top of the control panel*/
void writemessage(char *message)
{
   frontbuffer(TRUE);
```

```
    color(WHITE);
    cmov2i(30,833);
    charstr(message);
    frontbuffer(FALSE);
}

/* erases message in box at top of control panel*/
void erasemessage()
{
    frontbuffer(TRUE);
    color(BLACK);
    rectf(26,826,474,849);
    frontbuffer(FALSE);
}

/* redraws the entire window*/
void makeframe()
{
    reshapeviewport();
    getorigin(&org[X],&org[Y]);
    color(BLACK);
    clear();
}

/* if in an allowable area, updates positions of source locations
// also updates globals pix_to_p*/
void movsources(long my)
{

    if(my < head_y+45 && my > head_y-45) mov = FALSE;
    if(mov){

        pos[CH1][X]=(230);
        pos[CH1][Y]=(my);
        pos[CH2][X]=(230);
        pos[CH2][Y]=(head_y+(int)((float)(my-head_y)/2.0));
        pos[CH3][X]=(wall[1]);
        pos[CH3][Y]=(head_y+(int)((float)(my-head_y)/2.0));
        pos[CH4][X]=(wall[0]);
        pos[CH4][Y]=(head_y+(int)((float)(my-head_y)/2.0));
        pix_to_p = abs(pos[CH1][Y]-head_y);

        }
    mov = TRUE;
}

/* draws the graphics*/
```

```
void makefigure()
{
   int i;
   char string[10];
   /* the room*/
   color(BLACK);
   clear();
   color(BLUE);
   rectf(wall[0],45,wall[1],805);
   color(BLACK);
    /*15 pixels/foot, 24'x 50' room*/
   rectf(wall[0]+5,50,wall[1]-5,800);
   color(BLUE);
   rect(220,50,240,800);
   color(RED);
   for(i=    ;i<25; i++){
      rectf(227,i*30+80,233,i*30+83);}

      /* the head */
   color(YELLOW);
   circf(230,head_y,8);
   arcfi(230,head_y+12,7,2400,3000);

      /* the message box*/
   rect(25,825,475,850);

      /* Mode settings*/
   cmov2i(80,860);
   color(GREEN);
   if(mode==2) charstr("2D Mode");
   if(mode==3) charstr("3D Mode");
   cmov2i(280,860);
   if(tracker==0) charstr("3 Space Tracker");
   if(tracker==1) charstr("ISO Tracker");
   if(tracker==2) charstr("No Head Tracker");

   color(WHITE);
   sprintf(string, "%6.2f",(float)pix_to_wall/15.0);
   cmov2i(150,20);
   charstr("Distance to Wall: ");
   charstr(string);


}

/*calculates the reflection time delay, both channel gains,
//and reflection's incident angle */
```

```
void calc_delay_vol_angle(){

    float r,h;

    /*distance and delay*/
    distance = (float)pix_to_p/15.0;
    printf("dist %f\n",distance);
    h=((float)pix_to_wall/15.0);
    printf("h %f\n",h);
    r=sqrt((distance/2.0)*(distance/2.0)+(h*h));
    printf("r %f\n",r);
    delay = (int)(((2*r-distance)/1128.6)*10000);

    /*wall reflection angles*/
    rwall_ang=atan2((double)(pix_to_wall),((double)(pix_to_p)/2.0));
    rwall_ang = rwall_ang*180/M_PI;

    lwall_ang = 360-rwall_ang;

    /*ceiling reflection angle*/
    ceiling_ang = 90-rwall_ang;


    /*gain settings based on 1/r distance scaling*/

    p_vol = (int)((255.0*3)/(distance));

    r_vol = (int)((255.0*3)/(2*r));

}
```

C.2 Time Delay and Volume Control Program

```
/*********************************************************
*
* This program reads in audio on the left channel of
* the audio processor board of the 4D/35 and outputs
* the same audio on the left output, and a delayed
* version of this audio on the right channel. Another
* function of the program is to provide real-time gain
* control. The program is designed to be spawned from a
* parent process that sends delay and gain variables,
* and sends a done signal when the parent is killed.
* Author: ERIC L. SCARBOROUGH
* Date  : July 92
*********************************************************/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include "audio.h"
#include <signal.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>

static ALport iport;
static ALport oport;
static ALconfig theconfig;

static char *myname;
static long  qsize;
static long  nchannels;
static long  sampfmt;
static long  rate;
static int   samples_per_frame;
static int   bytes_per_sample;
static int   frames_per_transfer;

static int   ms_delay;
unsigned char *buf, *outbuf,*rmdbuf, *first_rmdbuf;
unsigned char *second_rmdbuf, *endrmdbuf;
unsigned char temp1,temp2;

main()
{
    int i,k,j,l,m,num_delay_samps=0;
    int done,delay,p_vol,r_vol,bytes_read,olddelay;
    long pvbuf[4];
    long pvbuflen;
    char rdbuf[20];

    /*
     * defaults
     */
    qsize               = 100000;
    frames_per_transfer  = 500; /* this number of frames processed each time*/
    sampfmt             = AL_SAMPLE_16;
    nchannels           = AL_STEREO;



    /* 16 bit stereo */
    sampfmt=AL_SAMPLE_16;
```

```
bytes_per_sample = 2;
samples_per_frame = nchannels;

/* set up configuration */
theconfig = ALnewconfig();
ALsetqueuesize(theconfig, qsize);
ALsetwidth(theconfig, sampfmt);
ALsetchannels(theconfig, nchannels):

/*
 * use 16000 sample rate for default
 */
pvbuf[0] = AL_INPUT_RATE;
pvbuf[2] = AL_OUTPUT_RATE;
pvbuf[1] =  AL_RATE_16000;
pvbuf[3] =  AL_RATE_16000;
pvbuflen = 4;
ALsetparams(AL_DEFAULT_DEVICE, pvbuf, pvbuflen);

/* open audio ports */
oport = ALopenport("the ouput port", "w", theconfig);
if (!oport)
{
   fprintf(stderr,"%s: failed to open audio write port\n", myname);
   exit(1);
}
iport = ALopenport("the input port", "r", theconfig);
if (!iport)
{
   fprintf(stderr,"%s: failed to open audio read port\n", myname,i);
   exit(1);
}

 /* default values for variable params */
 done = 0;
 delay = 160; /* 10 ms delay */
 p_vol = 255;
 r_vol  = 0;


/* set up pipe files. note: the file descriptor #s are
 specific to the parent process*/

close(8); /* close the write file, we just read*/
fcntl(7,F_SETFL,O_NDELAY); /* set for no delay if file empty*/

/* make the first read for shared variables */
```

```
bytes_read = read(7,rdbuf,15);
sscanf(rdbuf,"%i\t%i\t%i\t%i\n",&done,&delay,&p_vol,&r_vol);
num_delay_samps = delay;
printf("ref_delay: done: %i delay: %i p_vol %i r_vol %i\n",done,delay,p_vol,r_vol);

/* allocate buffer space */
buf = (void *)
  malloc(frames_per_transfer*
bytes_per_sample*samples_per_frame);
outbuf = (void *)
  malloc(frames_per_transfer*
bytes_per_sample*samples_per_frame);
rmdbuf = (void *)
  malloc(frames_per_transfer*
bytes_per_sample*samples_per_frame);

/* initialize pointers */
first_rmdbuf=second_rmdbuf=rmdbuf;
endrmdbuf=rmdbuf+frames_per_transfer*
bytes_per_sample*samples_per_frame;

/* loop while the parent is alive */

k=0;
olddelay=0;
while (done≠1)
{
 ALreadsamps(iport, buf, frames_per_transfer*
       samples_per_frame);
 bytes_read = read(7,rdbuf,15);
 if(bytes_read) {
      sscanf(rdbuf,"%i\t%i\t%i\t%i\n",&done,&delay,&p_vol,&r_vol);
   num_delay_samps = delay;
   printf("ref_delay: done: %i delay: %i p_vol %i r_vol %i\n"
    ,done,delay,p_vol,r_vol);
 }
 if (delay ≠ olddelay) {
    first_rmdbuf=second_rmdbuf=rmdbuf;
       endrmdbuf=rmdbuf+frames_per_transfer*
    bytes_per_sample*samples_per_frame;
   }

 /* set gains */
 pvbuf[0] = AL_LEFT_SPEAKER_GAIN;
    pvbuf[1] = p_vol;
    pvbuf[2] = AL_RIGHT_SPEAKER_GAIN;
    pvbuf[3] = r_vol,
```

```
pvbuflen = 4;
   ALsetparams(AL_DEFAULT_DEVICE, pvbuf, pvbuflen);

/* generate output buffer, right channel is delayed */
for(i=0;i<frames_per_transfer*samples_per_frame*2;i+=4){

   outbuf[i]=buf[i];
   outbuf[i+1]=buf[i+1];
   temp1=buf[i];
   temp2=buf[i+1];

   if(i<num_delay_samps*4&&&delay≠olddelay) {

      outbuf[i+2]= *first_rmdbuf;
      first_rmdbuf[0]=temp1;
      /*printf("rmd: %u\n",*first_rmdbuf);*/

      first_rmdbuf++;
      outbuf[i+3]= *first_rmdbuf;
      first_rmdbuf[0]=temp2;
      /*printf("rmd: %u\n",*first_rmdbuf);*/

      first_rmdbuf++;

   }
   else{
      if(second_rmdbuf==endrmdbuf)
         second_rmdbuf=rmdbuf;
      outbuf[i+2]= *second_rmdbuf;
      second_rmdbuf++;
      outbuf[i+3]= *second_rmdbuf;
      second_rmdbuf++;
      first_rmdbuf[0]=temp1;
      first_rmdbuf++;
      first_rmdbuf[0]=temp2;
      first_rmdbuf++;

   }

}
olddelay=delay;
endrmdbuf=first_rmdbuf;
first_rmdbuf=rmdbuf;
j=l=0;
/* play the new output buffer */
   ALwritesamps(oport, outbuf, frames_per_transfer*
      samples_per_frame);
```

```
    }
    /* close up */
    ALcloseport(iport);
    ALcloseport(oport);
    exit (0);

} /*main*/
```

Appendix D. *Visual Cue Simulation Code*

This appendix contains a listing of the programs that were written to perform the incorporation of visual images with the spatially located audio from the DIRAD. The first program is the main program, it is a modified version of the Redflag simulation program developed at AFIT.

D.1 Graphics Synthetic Environment Main Program

```
/* ---------------------------------------------------
//    Original Description (Feb 92):
//      This is a C++ re-write of the Red Flag
//          driver program. It is based on Mark Gerken's
//      battle manager software, Gary Lorimor 's Red
//      Flag software,and the support software by
//      John Brunderman, Joe Simpson and Bob Olson.
//---------------------------------------------------
//      Modified to provide visual cues for DIRAD
//          Audio Localization Cue Synthesizer (Summer 92)
//      by Capt Eric L. Scarborough
//
//      This program uses the basics of the Red Flag
//      driver program, however, the use of the polhemus
//          has been deleted.
//      The sound is supplied to the moving objects by
//      a spawning a child process that plays sounds
//      using the audio processor board. This program
//      sends gain settings via a Unix pipe to allow
//      for real time distance simulation. The object's
//      angle is calculated and sent to the DIRAD along
//      with the audio processor outputs to "spatially
//      position" the sounds.
// ---------------------------------------------------*/

extern "C" {
#include <math.h>
#include <string.h>
#include <stdlib.h>
#include <gl.h>
#include <device.h>
}
```

```
#include <unistd.h>
#include <math.h>

#include "readfile.h"
#include "tgrid.h"
#include "translator.h"
#include "worldwin.h"
#include "locdev.h"
#include "redflag.h"

#define OPTIONS "?hns t:r:"
#define USAGE  "Usage: testworld [-?/h] [-n] [-t <terrain file>] \n"
#define USAGE2  " -r redflag data\n -s sound"
#define USAGE3 "\n"
#define NORM_SPEED 6.
#define MAX_CHARS 80
#define EPSILON 0.025

extern        "C" {

   int getopt(int argc, char **argv, char *optstring);
}

World_Window   *The_World = NULL;
PhigsList      *acPHIGS = NULL;        /* dummy PhigsList pointer*/
TGrid          *Grid = NULL;         /* terrain grid*/
/*Holds the terrain and other static objs */
Translator     *Tlist = NULL;
Translator     *Olist = NULL;
LocalizationDevice *ld;
/*Light position*/
float  LightPos[] = {POSITION, 0.1, 0.0, 0.3, 0.0, LMNULL};

/* By defining the array to be 5600 we allow about 90 minutes of data */
/* database of object movement */
DataBase            rfData[MAX_RECORDS];
objDef              players[MAX_SESSIONS];
DataBase            time_temp,delta;
float             acMatrix[4][4];

/*-------------------------
//-------SUPPORTING METHODS------
//-------------------------*/

/* read Red Flag Data, fills rfData  */
int readRFfile (FILE*, objDef*, int*, int* );
void makeDeltas(DataBase *time_temp, DataBase *delta, float dSTEP, int j);
```

```
void move_plane(DataBase *time_temp, DataBase *delta, int i);
void set_plane(DataBase *time_temp, int i, float m[4][4]);

/* These two functions calculate angle and distance for
// a given object for sound simulations */
int set_sound(DataBase *time_temp, int i);
int set_gain(DataBase *time_temp, int i);

void terminate() {

   if (The_World->Get_NTSC())
      The_World->Set_HZ60();

   cout << "Red Flag system terminating.\n";
   cout.flush();
   exit(127);
}

/*-----------------------
//-------MAIN PROGRAM---------
/'-----------------------*/

int
main(int argc, char *argv[])
{
   foreground();
   /*Finite state machine */
   long       qdev;       /* queued device */
   short      but;        /* mouse button presses*/

   extern char   *optarg;
   extern int    optind;
   int           option;
   int           dg_color = 0;

   Point         ORP;
   Point         VRP;    /* view reference point */
   Point         VUP;    /* view up vector */
   Point         VPN;    /* view plane normal */
   UVNPoint      PRP;    /* projection reference point */


   /*declarations for database files*/
   char          *terrainlnkfile = (char *) NULL;
   char          *terraindbsfile = (char *) NULL;
```

```
float       deltax, deltay, deltaz;    /* movement increments */
int         rfData = FALSE;
/* pointer to database */
FILE        *rfFP;
char        rfDataFile[30];    /* name of red flag data file */
/* total number of data records from RF data */
int         totalFrames;
int         numAC;    /* number of objects */
int         offset;
int         slot_pos; /* integer identifies of session*/
int        i,j;
int        tween;
float       dSTEP;
Placement    *acNode;

Boolean     NTSC_mode = FALSE;
Boolean        done = FALSE;

float          front = 0.9; /*Front clipping plane*/
float          back = -120000.0; /*Back clipping plane*/

float          umin = -0.5206;
/*55 degree FOV horizontal */
float          umax = 0.5206;
/*with a 4 x5 aspect ratio*/
float          vmin = -0.4165;
float          vmax = 0.4165;

/* 3D sound variables */
Boolean            SOUND,olddone=FALSE;
int        mode = 2;
   int         tracker = 1;
   int        who = 1;
char         zer[1];
int         exerr = 0;
int            pfiles[2];
char        strng[25];
int        ch1_vol=0,ch2_vol=0;
int        ch1_ang,ch2_ang;
int        oldch1_vol=1,oldch2_vol=1;
int        changed_params = 0;
/* constructor for localization device class, 2D mode, port 2 */
ld = new LocalizationDevice(mode,2);

/* ------------------------
// ------ Parse command line ------
// ------------------------*/
```

```
rfData = FALSE;


while ((option = getopt(argc, argv, OPTIONS)) ≠ −1) {
   switch (option) {
   case 'r':
      strcpy (rfDataFile, optarg);
      if ((rfFP = fopen (rfDataFile "r")) == NULL) {
         printf ("\t*** Could not open RF file %s\n",rfDataFile);
         exit (99);
      }
      rfData = TRUE;
      break;

   case 'n':
      /*turn on NTSC mode; for HMD most likely*/
      NTSC_mode = TRUE;
      break;

   case 't':
      /*terrain.dbs and.lnk files*/
      terrainlnkfile = new char[strlen(optarg) + 5];
      terraindbsfile = new char[strlen(optarg) + 5];
      strcpy(terrainlnkfile, optarg);
      strcat(terrainlnkfile, ".lnk");
      terraindbsfile = strdup(terrainlnkfile);
      char          *tmp = strstr(terraindbsfile, ".lnk");
      if (tmp)
         strcpy(tmp, ".dbs");
      else {
         delete        terraindbsfile;
         terraindbsfile = NULL;
      }
      break;


   case 's':
      SOUND = TRUE;
      printf("sound on\n");
      break;
   case 'h':
   case '?':
   default:
      printf(USAGE);
      printf(USAGE2);
      printf(USAGE3);
      exit(127);
```

```
        break;
    }
}

/* It is mandantory that there is a Red Flag data file*/
if (!rfData) {
    printf ("No input Red Flag Data file provided!\n");
    exit (99);
}

/* if the -s option is given reset DIRAD initialize
// the headtracker, headtracker for audio only (not graphics)*/
if(SOUND){
        ld →resetfirmware();
        ld→initializetracker(tracker);
}

noport();
/*inits graphics without opening window*/
long   temp = winopen("");
cout ≪ "\n Please Standby - Reading files\n";
cout.flush();


Tlist = new Translator(terrainlnkfile);

if (terraindbsfile) {
    Grid = new TGrid(terraindbsfile, Tlist, 55, 2);
}

cout ≪ "\nReading Red Flag Data File\n";
cout ≪ "\nPlease wait, initializing objects...\n";
cout.flush();

if (Grid) {
    VRP.x = 10.0;
    VRP.y = 0.0;
    VRP.z = 10.0;
} else {
    VRP.x = 0.0;
    VRP.y = 0.0;
    VRP.z = 0.0;
}

VUP.x = 0.0;
VUP.y = 0.0;
VUP.z = 1.0;
```

```
VPN.x = 1.0;
VPN.y =  .0;
VPN.z = .5;

PRP.u = 0.0;
PRP.v = 0.0;
PRP.n = 1.0;
/*55 degree FOV*/
ORP.x = 0.0;
ORP.y = 0.0;
ORP.z = 0.0;

if (NTSC_mode)
   The_World =
   new World_Window(Grid, acPHIGS,XMAX170+10,
   YMAX170+10,10,10);
else
   /*choose full screen or small sreeen
   The_World =
   new World_Window(Grid, acPHIGS, 1280, 1024, 0, 0); */

   The_World = new World_Window(Grid, acPHIGS, 800, 600, 0, 0);

The_World→Set_Aspect_Ratio(4, 3);
The_World→Set_Border(FALSE);
The_World→Open_Window();
The_World→setPRP(PRP.u, PRP.v, PRP.n);
The_World→setWin(umin, umax, vmin, vmax);
The_World→setFB(front, back);
The_World→setVUP(VUP.x, VUP.y, VUP.z);
The_World→setVRP(VRP.x, VRP.y, VRP.z);
The_World→setVPN(VPN.x, VPN.y, VPN.z);
The_World→Set_Axis_Origin(ORP.x, ORP.y, ORP.z);
The_World→setProjType(PERSPECTIVE);
The_World→computeProjection();
The_World→compute3Dview();
The_World→Set_Light_Index(1);
The_World→Show_Grid (FALSE);
The_World→Show_Axes (FALSE);

if (NTSC_mode) {
   The_World→Show_Info(FALSE);
   The_World→Set_NTSC();
} else
   The_World→Show_Info(TRUE);
```

```
lmdef(DEFLMODEL, 1, 0, (float *) 0);
lmdef(DEFLIGHT, 1, 6, LightPos);
lmbind(LMODEL, 1);

deltax = -0.707;
deltay = -0.707;
deltaz = 0.;

The_World->Redraw_Window();

/* Set up the devices that will be placing values on the queue*/
qdevice(LEFTMOUSE);
qdevice(RIGHTMOUSE);
qdevice(KEYBD);
cursoff();      /* make the cursor invisible */

winclose(temp);

/* Read the data file containing movable objects, flight paths*/
totalFrames = readRFfile (rfFP, players, &offset, &numAC);

/* start audio interface program
// first set up a pipe to pass gain settings and a done
// variable to the audio program */
if(SOUND){
      zer[0] = '0';
   exerr = pipe(pfiles);

   if (exerr == -1) {
         printf ("\tpipe failed\n");
         printf ("\t%s\n",sys_errlist[errno]);
         exit (127);
      }
   /* the pipe file descriptors are "hard coded" in
   // the child process
   //printf("%i\t%i\n",pfiles[0],pfiles[1]);
   // send initial settings to the pipe */

   sprintf(strng, "%3i\t%3i\t%3i",done,ch1_vol,ch2_vol);
   write(pfiles[1],strng,(int)strlen(strng));

   /*now fork to spawn the child process, helloop.c
   //originally just played helicopter loop, now
   //both tank and helicopter are played */

   who = fork();
   if (who == 0) {
```

```
            exerr = execlp("helloop","helloop",zer);
            if (exerr == -1) {
               printf ("\texec failed\n");
               printf ("\t%s\n",sys_errlist[errno]);
               exit (127);
            }
         }

         close(pfiles[0]);
      }

      cout << "Red Flag system is ready.\n";
      cout.flush();

   /*-------------------
   //-Done with initializations,-
   //-Entering the main loop-----
   //-------------------*/

      slot_pos = 0;
         players[slot_pos].stop = totalFrames;

      The_World->Set_Dynamic_Object_List (players[slot_pos].acPhigs);
         dSTEP = NORM_SPEED;

      do {

      /* Now, the *BIG* loop through all the time steps*/
         for (j=0; j<totalFrames; j++){

         /* Change the players if the stop time has been exceeded */
         if(j > players[slot_pos].stop ) {
            slot_pos++;
            The_World->
            Set_Dynamic_Object_List (players[slot_pos].acPhigs);
         }

         /* get the distance between this and the next frame */
         /* create a delta for each value */

         makeDeltas (&time_temp, &delta, dSTEP, j);

         /* Show movement between the data points with linear interpolation
            using the variables calculated above */

         for (tween = 0; tween < dSTEP; tween++){
```

```
/* Check to see if there is mouse input or keyboard input*/

if ((qdev = qtest()) ≠ NULL) {

   if (qdev == RIGHTMOUSE) but =1;
   else if (qdev == LEFTMOUSE) but =2;
   else {
      qread(&but);
   }
qreset();
switch (but) {

 /* Move forward in the scene by a small step*/

   case 1:
      VRP.x += deltax * 1.;
      The_World→setVRP(VRP.x, VRP.y, VRP.z);
      break;

 /* Move backward in the scene by a small step */

   case 2:
      VRP.x -= deltax * 1.;
      The_World→setVRP(VRP.x, VRP.y, VRP.z);
      break;

/* set normal display speed */

case 'n':
   dSTEP = NORM_SPEED;
   break;

 /* set display at half the normal rate */

case 'h':
   dSTEP = NORM_SPEED*2;
   break;


 /* Set the display to twice the normal speed */
case 'd':
      dSTEP = NORM_SPEED / 2;
      break;


case 'q':
   done = TRUE;
```

```
        break;

    default:
        break;
    }
    but = NULL;
    }/* end of check for something on the event queue */
    The_World→setVPN(VPN.x, VPN.y, VPN.z);
    The_World→setVUP(VUP.x, VUP.y, VUP.z);

/*MOVE OBJECTS*/

    for (i=0; i<numAC; i++) {
        move_plane(&time_temp, &delta, i);
        set_plane(&time_temp, i, acMatrix);
        acNode = (Placement *)players[slot_pos].acPhigs→
        GetNode ((long)players[slot_pos].actype[i]);
            acNode→LoadPositionMatrix(acMatrix);
        /* here we calculate angle and distance to
        // the two objects, then send the angles to
        // the DIRAD */
        if(SOUND){
            if (i==0&&done==FALSE) {
                ch2_ang=set_sound(&time_temp, i);
                ch2_vol = set_volume(&time_temp, i);}
            if (i==1&&done==FALSE) {
                ch1_ang=set_sound(&time_temp, i);
                ch1_vol = set_volume(&time_temp, i);}
              /*DIRAD command sets position of both
            //channels */
            ld→doubleangleset(ch1_ang,ch2_ang,0,0);
        }
        }



    if (done) break;
    if(SOUND){
     /* if audio program needs to be updated */
       if(done ≠ olddone ||
          ch1_vol ≠ oldch1_vol || ch2_vol ≠ oldch2_vol )
              changed_params = 1;

          olddone = done;
          oldch1_vol = ch1_vol; oldch2_vol = ch2_vol;
       /* send new settings to pipe */
       if(changed_params){
```

```
            sprintf(strng, "%3i\t%3i\t%3i",done,ch1_vol,ch2_vol);
            write(pfiles[1],strng,(int)strlen(strng));
        }
          changed_params = 0;
        }
      The_World→Redraw_Window();

    }    /* for (tween*/
    }
    } while (!'one);

    curson();
    /* if were done, kill audio program */
    if(SOUND){
        sprintf(strng, "%3i\t%3i\t%3i",done,ch1_vol,ch2_vol);
        write(pfiles[1],strng,(int)strlen(strng));
    }
    cout.flush();
    sleep(2);
    terminate();
}
```

D.2 Distance and Angle Calculation

These two short routines are used to find the distance and angle to the specified object with respect to the lister's position. The return values are sent to the audio program for distance scaling, and to the DIRAD for sound source positioning.

```
#include "redflag.h"
#include <math.h>
#include <gl.h>

/*****************************************************************
*
*    DATE: 5 Aug 92
*    VERSION: 1.0
*
*    NAME: set_sound                                              *
*    DESCRIPTION: positions sound according to location
*                 of graphic object
*    PASSED VARIABLES: time_temp
*    RETURNS: azimuth angle to object
*    AUTHOR: Eric L. Scarborough
*    HISTORY:
*
```

```
*****************************************************************/
int set_sound(DataBase *time_temp, int i)
{
   /* time_temp holds the current object's position and
   orientation */

   float xcenter,ycenter;
   double angle,offset;
   /* if listener is not at 0,0 facing forward */
   xcenter = 0.0;
   ycenter = 0.0;
   offset  = 0.0;

   /* calculate azimuth angle according to the new data */
   angle=atan2((double)(time_temp->y[i]-ycenter),
         (double)(-time_temp->x[i]-xcenter));
   angle = angle*180/M_PI-offset;
   if(angle<0.0) angle += 360.0;
   return (int)angle;
}
#include "redflag.h"
#include <stdlib.h>
#include <gl.h>
#include <math.h>

/*****************************************************************
*
*    DATE: 5 Aug 92
*    VERSION: 1.0
*    DESCRIPTION: finds the distance from the location of
*                 the current graphics object to the
*          listener, calculates gain required to
*          simulate this distance
*    PASSED VARIABLES: time_temp
*    RETURNS: gain setting                    *
*
*    AUTHOR: Eric L. Scarborough
*    HISTORY:
*
*****************************************************************/
int set_gain(DataBase *time_temp, int i)
{
   int gain;
   float distance;

   distance = sqrt(time_temp->x[i]*time_temp->x[i]+
         time_temp->y[i]*time_temp->y[i]);
```

```
    /*use 1/distance scaling, distance of 4 feet gives
    //max gain */
    gain = abs((int)(255*4/distance));
    if(gain>255) gain = 255;
    return gain;
}
```

D.3 Audio Control Program

This program is spawned from the main program and is used to read in stored sound files for the graphics objects, play them until the simulation is over, and set the volume levels of the two audio output channels based on the object's distance.

```
/*************************************************************
*
*    This program sets up and controlls an audio port on
* the SG 4D/35, specifically to play a stored raw sound
* file for use as input to the DIRAD. The right channel
* provides audio for a helicoper and the left channel is
* used for a tank.
* The gain settings can be changed at any time, to
* simulate distance.
*    This program relies on a "parent" process to launch it,
* set up a pipe to communicate the shared varibles, and
* stop it.
*    Author: Eric L. Scarborough
*    Date:   27 August 92
*************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include "audio.h"
#include <signal.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>

static ALport iport;
static ALport oport;
static ALconfig theconfig;

static char *myname, *snd;
static long  qsize;
static long  nchannels;
static long  sampfmt;
static long  rate;
static int   change_rate;
```

```
static int   samples_per_frame;
static int   bytes_per_samp;
static int   frames_per_transfer;
static long  new_rate;
static long  old_inrate;
static long  old_outrate;
unsigned char sndbuf1[120000],sndbuf2[120000];

main()
{
   int i,fd,k,j,l,m,num_delay_samps=0;
   int done,ch1_vol,ch2_vol,bytes_read,finished;
   long pvbuf[4];
   long pvbuflen;
   long filled;
   long samps_read,p;
   char rdbuf[20],filename[13];

   int errno =0;

   /*
    * defaults
    */
   qsize              = 200000;
   frames_per_transfer  = 500;
   sampfmt            = AL_SAMPLE_8;
   bytes_per_samp       = 1;
   nchannels          = AL_STEREO;

   /*
    * use 16000 sample rate for default
    */

   samples_per_frame = nchannels;
   theconfig = ALnewconfig();
   ALsetqueuesize(theconfig, qsize);
   ALsetwidth(theconfig, sampfmt);
   ALsetchannels(theconfig, nchannels);

   /* default values for variable params */
   done = 0;
   ch1_vol = 128;
   ch2_vol  = 0;

   /* set up pipe files */
   /* note these are "hard coded" and depend on the
      parent process  */
```

```
close(8);
fcntl(7,F_SETFL,O_NDELAY);
/* read the initial settings */
bytes_read = read(7,rdbuf,15);
sscanf(rdbuf,"%i\t%i\t%i\n",&done,&ch1_vol,&ch2_vol);

/*
 * open audio ports, malloc buffer space
 */

pvbuf[0] = AL_OUTPUT_RATE;
pvbuf[1] = AL_RATE_8000;
pvbuflen = 2;
ALsetparams(AL_DEFAULT_DEVICE, pvbuf, pvbuflen);

oport = ALopenport("the ouput port", "w", theconfig);
if (!oport)
{
   fprintf(stderr,"%s: failed to open audio write port\n", myname);
   exit(1);
}

/* open helicopter sound file */
strcpy(filename, "hel.raw");
if ((fd = open(filename, O_RDONLY)) < 0)
   {
      fprintf(stderr, "%s: can't open %s: %s\n", myname, filename,
         strerror(errno));
      exit(1);
   }
/* the first read is used to "find" a good representation of
   the sound, the second reads in 2.5 seconds worth,
   the sound files are 16 kHz, 8 bit, stereo */
bytes_read = read(fd, sndbuf1,10000);
bytes_read = read(fd, sndbuf1,80000);

/* open the tank sound file */
strcpy(filename, "tank.raw");
if ((fd = open(filename, O_RDONLY)) < 0)
   {
      fprintf(stderr, "%s: can't open %s: %s\n", myname, filename,
         strerror(errno));
      exit(1);
   }

bytes_read = read(fd, sndbuf2,10000);
```

```
    bytes_read = read(fd, sndbuf2,80000);
    samps_read = bytes_read/bytes_per_samp;

    /*combine the 2 sounds into one output buffer,
      left = tank, right = helicopter */

    printf("%i\n",samps_read);
    for(p=0;p<samps_read;p+=2){
    sndbuf1[p]=sndbuf2[p];}

    done = 0;

    while (done ≠ 1)
    {

    bytes_read = read(7,rdbuf,15);

    if(bytes_read) {
       sscanf(rdbuf,"%3i\t%3i\t%3i\n",&done,&ch1_vol,&ch2_vol);}

    /* set gain for object's distance */
    pvbuf[0] = AL_LEFT_SPEAKER_GAIN;
       pvbuf[1] = ch1_vol;
       pvbuf[2] = AL_RIGHT_SPEAKER_GAIN;
       pvbuf[3] = ch2_vol;
    pvbuflen = 4;
       ALsetparams(AL_DEFAULT_DEVICE, pvbuf, pvbuflen);
    /* if output que getting empty, refill */
    if(ALgetfilled(oport) <4000) {
    ALwritesamps(oport, sndbuf1, samps_read);}

    } /*while*/
    sleep(2);
    close(fd);
    ALcloseport(oport);
    exit (0);

} /*main*/
```

Bibliography

1. Armstrong Laboratory Biodynamics & Biocommunications Division, Bioacoustics & Biocommunications Branch (AL/CFBA), Wright-Patterson AFB, OH 45433-6573. *Auditory Localization Cue Synthesizer (ALCS) User's Manual*, March 1992.
2. Armstrong Laboratory/CFBA. *3-D Audio Display Technology*. Research Description Brochure. Wright-Patterson AFB, OH 45433.
3. Begault, Durand R. and Elizabeth M. Wenzel. *Techniques and Applications for Binaural Sound Manipulation in Human-Machine Interfaces*. Technical Report NASA TM-102279, National Aeronautics and Space Administration, 1991.
4. Blauert, Jens. *Spatial Hearing: The Psychophysics of Human Sound Localization*. MIT Press, Cambridge, Massachusetts, 1983.
5. Brandt, Capt Jim. *Real Imaging Display System*. MS thesis, AFIT/GEO/ENG/92D-03, Air Force Institute of Technology (AU), 1992.
6. Burkhard, M.D. and R.M. Sachs. "Anthropometric manikin for acoustic research," *Journal of the Acoustical Society of America*, *58* (1975).
7. Butler, Robert A., et al. "Apparent Distance of Sounds Recorded in Echoic and Anechoic Chambers," *Journal of Experimental Psychology*, *6* (1980).
8. Calhoun, Gloria L., et al. "Three-Dimensional Auditory Cue Simulation for Crew Station Design/Evaluation." *Proceedings of the Human Factors Society*. 1987.
9. Chung, J.C. and others. "Exploring virtual worlds with head-mounted displays." *Proceedings of the SPIE1083*. 1989.
10. Coleman, Paul D. "An Analysis of Cues to Auditory Depth Perception in Free Space," *Psychological Bulletin*, *60* (1963).
11. Craig, Rushby C. *Binaural Sound Localization Using Neural Networks*. MS thesis, AFIT/GE/ENG/91D-13, Air Force Institute of Technology (AU), 1991 (AD-A243878).
12. Crystal River Engineering. *The Convolvotron: Synthetic 3D Audio*. Sales Brochure. 12350 Wards Ferry Road, Groveland, CA 95321, 1990.
13. Doll, T.J., et al. *Development of simulated directional audio for cockpit applications*. Technical Report AAMRL-TR-86-014, Armstrong Aerospace Medical Research Laboratory, 1986.
14. Ericson, Mark A. and Richard L. McKinley. "Auditory Localization Cue Synthesis and Human Performance." *IEEE National Aerospace and Electronics Conference*. 1989.

15. Focal Point 3-D Audio. *Focal Point: 3D Audio for The Macintosh II*. Sales Brochure. 1402 Pine Avenue, Suite 127, Niagra Falls, NY 14301, 1991.

16. Foster, Scott H., et al. "Real Time Synthesis of Complex Acoustic Environments." Sent with Convolvotron sales brochure.

17. Gardener, M.B. "Some moaural and binaural facets of median plane localization," *Journal of the Acoustical Society of America*, *54* (1973).

18. Genuit, Kluas and Wade R. Bray. "The Aachen Head System," *Audio* (December 1989).

19. Gerken, Capt, Mark J. *An Event Driven State Based Interface for Synthetic Environments*. MS thesis, AFIT/GCS/ENG/91D-07, Air Force Institute of Technology (AU), 1991 (AD-A243765).

20. Gierlich, H.W. and K. Genuit. "Processing Artificial-Head Recordings," *Journal of the Audio Engineering Society*, *37* (1989).

21. Greisinger, David. "Equalization and Spatial Equalization of Dummy-Head Recordings for Loudspeaker Reproduction," *Journal of the Audio Engineering Society*, *37* (1989).

22. Han, H.L. "Measuring a Dummy Head in Search of Pinna Cues." presented at the 90th Convention of the Audio Engineering Society, Paris, 1991 Feb. 19-22, preprint 3066.

23. Jones, Bill and Boris Kabanoff. "Eye movements in auditory space perception," *Perception and Psycophysics*, *17* (1975).

24. Knudsen, Eric I. and Michael S. Brainard. "Visual Instruction of the Neural Map of Auditory Space in the Developing Optic Tectum," *Science* (July 1991).

25. Krokstadt, A., et al. "Calculating the acoustical room response by the use of a ray tracing technique," *Journal of Sound and Vibration*, *8* (1968).

26. Kulkarni, Abhijit and H.Steven Colburn. "Auditory Image Processing in a Virtual Acoustic Environment." *Proceedings of the 1991 Seventeenth Annual Northeast Bioengineering Conference*. 1991.

27. Lambert, R. M. "Dynamic theory of sound-source localization," *Journal of the Acoustical Society of America*, *56* (1974).

28. Lanier, Robin. "The Blue Max Affair," *Audio* (November 1989).

29. Mackous and D. M. Middlebrooks. "Two-dimensional sound localization by human listeners," *Journal of the Acoustical Society of America*, *87* (1990).

30. McKinley, Richard L. *Concept and Design of an Auditory Localization Cue Synthesizer*. MS thesis, AFIT/GE/ENG/88D-29, Air Force Institute of Technology (AU), 1988 (AD-A203053).

31. Mehrgardt, S. and V. Mellert. "Transformation characteristics of the external human ear," *Journal of the Acoustical Society of America*, *61* (1977).

32. Mike Haas, Armstrong Laboratory, "Personal Conversation," September 1992.

33. Moller, Henrik. "Reproduction of Artificial-Head Recordings through Loudspeakers," *Journal of the Audio Engineering Society*, *37* (1989).

34. "NASA's Virtual Workstation: Using Computers to Alter Reality," *NASA Tech Briefs*, *12* (July/August 1988).

35. Oldfield, Simon R. and Simon P. A. Parker. "Acuity of sound localisation: a topography of auditory space. I. Normal hearing conditions," *Perception*, *13* (1984).

36. Oldfield, Simon R. and Simon P. A. Parker. "Acuity of sound localisation: a topography of auditory space. II. Pinna Cues Absent," *Perception*, *13* (1984).

37. Pick, Herbert L. Jr., et al. "Sensory conflict in judgments of spatial direction," *Perception and Psycophysics*, *6* (1969).

38. Platt, Bruce B. and David H. Warren. "Auditory localization and the importance of eye movements and a textured visual environment," *Perception and Psycophysics*, *12* (1972).

39. Plenge, G. "On the differences between localization and lateralization," *Journal of the Acoustical Society of America*, *56* (1974).

40. Roffler, S. K. and R. A. Butler. "Factors that influence the localization of sound in the median plane," *Journal of the Acoustical Society of America*, *43* (1968).

41. Rubak, Per. "Headphone Signal Processing System for Out-of-Head Localization." presented at the 90th Convention of the Audio Engineering Society, Paris, 1991 Feb. 19-22, preprint 3063.

42. Sakamoto, N., et al. "On "Out-of-Head Localization" in Headphone Listening," *Journal of the Audio Engineering Society*, *24* (1976).

43. Sayers, Bruce McA. and E. Colin Cherry. "Mechanism of Binaural Fusion in the Hearing of Speech," *Journal of the Acoustical Society of America*, *29* (1957).

44. Scharf, Bertram and Soren Buus. "Audition I." *Handbook of Perception and Human Performance, Volume 1, Sensory Processes and Perception* edited by Kenneth R. Boff, et al., New York: Wiley-Interscience, 1986.

45. Scharf, Bertram and Adrianus J. M. Houtsma. "Audition II." *Handbook of Perception and Human Performance, Volume 1, Sensory Processes and Perception* edited by Kenneth R. Boff, et al., New York: Wiley-Interscience, 1986.

46. Schroeder, M.R. "Digital Simulation of Sound Transmission in Reverberant Spaces," *Journal of the Acoustical Society of America*, *47* (1970).

47. Sedgwick, H.A. "Space and Motion Perception." *Handbook of Perception and Human Performance, Volume 1, Sensory Processes and Perception* edited by Kenneth R. Boff, et al., New York: Wiley-Interscience, 1986.

48. Shaw, E.A.G. "Transformation of sound pressure from the far field to the eardrum in the horizontal plane," *Journal of the Acoustical Society of America*, *56* (1974).

49. Simpson, Capt Dennis J. *An Application of the Object-Oriented Paradigm to a Flight Simulator*. MS thesis, AFIT/GCS/ENG/91D-22, Air Force Institute of Technology (AU), 1991 (AD-A243624).

50. "Sounds Good," *VR World*, *1* (1992).

51. Strybel, Thomas Z. and David R. Perrott. "Discrimination of relative distance in the auditory modality: The success and failure of the loudness discrimination hypothesis," *Journal of the Acoustical Society of America*, *76* (1984).

52. Sunier, John. "A History of Binaural Sound," *Audio* (March 1986).

53. Sunier, John. "Ears Where the Mikes Are, Part I," *Audio* (November 1989).

54. Wallach, H. "On sound localization," *Journal of the Acoustical Society of America*, *10* (1939).

55. Weiner, F.M. "On the diffraction of a progressive sound wave by the human head," *Journal of the Acoustical Society of America*, *19* (1947).

56. Wenzel, Elizabeth M. *Technical Aspects of a Demonstration Tape for Three-Dimensional Sound Displays*. Technical Report NASA TM-102826, National Aeronautics and Space Administration, 1991.

57. Wenzel, Elizabeth M. *Three-Dimensional Virtual Acoustic Displays*. Technical Report NASA TM-103835, National Aeronautics and Space Administration, 1991.

58. Wenzel, Elizabeth M. and others. "Realtime Digital Synthesis of Localized Auditory Cues over Headphones." Sent with Convolvotron sales brochure.

59. Wightman, Frederic L. and Doris J. Kistler. "Headphone simulation of free-field listening. I: Stimulus synthesis," *Journal of the Acoustical Society of America*, *85* (1989).

60. Wightman, Frederic L. and Doris J. Kistler. "Headphone simulation of free-field listening. II: Psychophysical validation," *Journal of the Acoustical Society of America*, *85* (1989).

61. Wightman, Frederic L. and Doris J. Kistler. "The dominant role of low-frequency interaural time differences in sound localization," *Journal of the Acoustical Society of America*, *91* (1992).

62. Wright, D., et al. "Pinna reflections as cues for localization," *Journal of the Acoustical Society of America*, *56* (1974).

Vita

Eric L. Scarborough was born on 9 August 1961 in Philadelphia, Pa. In 19 he graduated from Wissahickon High School and enlisted in the United States Air Force. He was trained as a Aircraft Communication Repairman and served at Davis-Monthan AFB AZ, Eielson AFB AK, and Holloman AFB NM. Through the Airman's Education and Commissioning Program, he received a Bachelor of Science in Electrical Engineering from New Mexico State University in 1987. Upon commissioning from Officer's Training School in 1988, Captain Scarborough was assigned to the Human Systems Division, Harry G. Armstrong Aerospace Medical Research Laboratory, Acceleration Effects Branch, WPAFB OH. During this assignment, he served as head of the Medical Electronics Section and was a volunteer test subject for numerous sustained high acceleration experiments on the Dynamic Environment Simulator, a man-rated, three axis centrifuge. He also served as associate investigator for several research protocols including high G tests of night vision helmets and Positive Pressure Breathing flight suits.

Captain Scarborough met his beautiful wife, Jan, while attending NMSU and they were married on 6 Feb 1988. The Scarboroughs have two daughters, Cara Lynn born in 1990 and Leah Ann born in 1992.

Permanent address: 416 Red Bud Lane
Wright-Patterson AFB,
Ohio 45433

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1992	Master's Thesis

4. TITLE AND SUBTITLE
Enhancement of Audio Localization Cue Synthesis by Adding Environmental and Visual Cues

5. FUNDING NUMBERS

6. AUTHOR(S)
Eric L. Scarborough, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology, WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GE/ENG/92D-34

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Mr. Mark A. Ericson
AL/CFBA, WPAFB, OH 45433

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

An audio localization cue synthesizer, the DIRectional Audio Display (DIRAD) was used to simulate auditory distance, room reflections, and to provide spatial audio for computer graphics images. The DIRAD processes input audio signals to generate spatially located sounds for headphone listening. The DIRAD can position audio sources around the head and these sounds are stable with respect to the listener's head position. An interactive, real-time simulation of auditory distance and room reflections was accomplished using the DIRAD in combination with a Silicon Graphics audio processor board installed in a Personal Iris 4D/35. Several demonstrations of auditory distance and the effects of early reflections are detailed, including a simulation of a direct sound source and three reflections that employed two DIRAD systems. Stored sound files were used to accompany three dimensional graphics images that were displayed on both a Silicon Graphics CRT and a three dimensional optical display device. The use of the 4D/35 audio processor board proved to be an effective means of preprocessing audio for the DIRAD for these simulations. The combination of AFIT's Silicon Graphics workstations and the DIRAD proved to be a practical solution to the problem of combining virtual visual and audio cues.

14. SUBJECT TERMS
auditory localization, auditory perception, virtual reality, 3-D audio, acoustic reflection, sound equipment

15. NUMBER OF PAGES
129

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reports. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to stay *within the lines* to meet *optical scanning requirements*.

Block 1. Agency Use Only *(Leave blank)*

Block 2. Report Date. Full publication date including day, month, and year, if available (e.g. 1 Jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element number(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the report.

Block 9. Sponsoring/Monitoring Agency Name(s) and Address(es). Self-explanatory.

Block 10. Sponsoring/Monitoring Agency Report Number. *(If known)*

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with ... Trans. of ... Was published in ... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denotes public availability or limitations. Cite any availability to the public. Enter additional limitations or special markings in all capitals (e.g. NOFORN, REL, ITAR).

DOD	- See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	- See authorities.
NASA	- See Handbook NHB 2200.2.
NTIS	- Leave blank.

Block 12b. Distribution Code.

DOD	- Leave blank.
DOE	- Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA	- Leave blank.
NTIS	- Leave blank.

Block 13. Abstract. Include a brief *(Maximum 200 words)* factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code *(NTIS only)*.

Blocks 17 - 19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contains classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.