TOWSON UNIVERSITY
COLLEGE OF GRADUATE STUDIES AND RESEARCH

COMPUTER-BASED RECORDED SPEECH

FOR

WORD RECOGNITION TESTING

By

Matthew H. Perry

A thesis

Presented to the faculty of

Towson University

In partial fulfillment

Of the requirements for the degree

Doctor of Audiology

May 2007

Towson University
Towson, Maryland 21252

TOWSON UNIVERSITY
COLLEGE OF GRADUATE STUDIES AND RESEARCH

DOCTORAL THESIS APPROVAL PAGE

This is to certify that the thesis prepared by Matthew H. Perry, entitled Computer-based Recorded Speech for Word Recognition Testing has been approved by this committee as satisfactory completion of the requirement for the degree of Doctor of Audiology (Au.D.) in the department of Audiology, Speech-Language Pathology and Deaf Studies

_____ _____
Diana C. Emanuel, Ph.D. Date
Chair, Audiology Doctoral Thesis Committee

_____ _____
Alex Storrs, Ph.D. Date
Committee Member

_____ _____
Steven Pallet, Au.D. Date
Committee Member

_____ _____
Jin K. Gong, Ph.D. Date
Dean, College of Graduate Studies and Research

ABSTRACT

COMPUTER-BASED RECORDED SPEECH

FOR

WORD RECOGNITION TESTING

Matthew H. Perry

Although the importance of using standardized recordings for word recognition testing is well established, many clinicians still routinely use monitored-live voice (MLV). A prototype software program was developed to combine the speed and flexibility associated with MLV with the standardization of recorded word lists. The sound quality of 20 personal computers was evaluated for total harmonic distortion plus noise (THD+N). The computers with the poorest sound quality were then used to replace the recorded speech sources of two clinical audiometers and tested for compliance with ANSI standard s3.36-1996. Of the 20 computers evaluated, only one had sound quality too poor to qualify for clinical use under the ANSI standard. These findings suggest the sound quality of most personal computers is adequate for clinical use, and future computer-based presentation methods for word recognition testing could provide an attractive alternative to traditional methods.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

The field of audiology evolves with the technology available. This is particularly evident in the area of amplification. Although the nature of hearing loss has remained constant through the years, the techniques used to make sounds audible to those with hearing loss have changed rapidly. Today's hearing aids were preceded by numerous amplification devices, including analog hearing aids, body-worn hearing aids, ear trumpets, and cupping the hand around the ear. As technology improved, it was adapted to assist in the task of providing audibility to those with hearing loss.

Diagnostic techniques used to assess hearing ability have also progressed with technological advances. It is a common practice to assess word recognition ability by using a standardized word list (Martin, Champlin & Chambers, 1998). Over the years, these word lists have been recorded on many types of media. Vinyl records were a popular medium due to their high fidelity (Hirsh et al., 1952). Reel-to-reel and cassette recordings took advantage of magnetic tape technology, resulting in better durability and a mild decrease in initial sound quality. However, since both vinyl and magnetic tape recordings use analog technology, they are susceptible to signal degradation over time. Today, many of the word lists are available on compact disc. The digital nature of these recordings enables the word lists to be presented across different clinics with little variation. A compact disc can be duplicated with no difference between recordings, and the sound quality does not degrade through repeated use. In theory, a listener presented

with a specific CD recording, at a specified level, would receive the same acoustic stimulus regardless of the location of the testing facility or the actual compact disc used.

Although technological advancement has enabled audiologists to provide superior hearing healthcare services, not all audiologists choose to use the most current technology available. For example, even though CD recordings are widely available, many audiologists still use live voice to assess word recognition ability (Martin, Champlin, & Chambers, 1998). One explanation for this is that the use of recorded word lists is too time consuming and inflexible.

The advancement of audio technology did not end with the development of the compact disc. Today, digital audio can be presented via the personal computer with extreme flexibility. Stach, Davis-Thaxton, and Jerger (1995) investigated the use of personal computers in word recognition testing and demonstrated that word lists could be presented using a personal computer in a much faster and flexible manner than a tape/CD presentation without a decrease in word recognition scores. Sixteen years have passed since Stach et al. demonstrated the utility of word recognition testing presented via personal computer, and the availability and capability of the personal computer has increased dramatically. Yet a PC-based method, similar to that proposed by Stach and colleagues, has not become part of the standard protocol in the audiology clinic even though computer-based pure tone audiometry and database management of audiological files is common.

The present study was designed to re-investigate the concept of computer based word recognition testing. Using a prototype software program developed by the author, this study investigated the sound quality of common personal computers and assessed

whether the average personal computer can be used to present WAV file speech stimuli under the current electroacoustic quality standards for audiometry. If the sound quality of common personal computer sound cards is adequate, incorporating computer-based presentation methods into audiometric speech testing should be considered. The use of WAV files for word recognition testing has the potential to decrease testing time, and increase flexibility, while providing a valid and reliable test method.

Chapter 2

Literature Review

Origins of Speech Audiometry

The human voice has likely been used to assess hearing ability, informally, since the beginning of oral communication. Urbantschitsch (as cited in Silverman, 1983) reported the use of speech stimuli in a clinical setting as early as the sixteenth century. However, until the twentieth century, the lack of technology prohibited standardization and limited the usefulness of speech testing. The first documented, standardized approach to the measurement of speech perception was conducted in 1910 (Mendel & Danhauer, 1997). This approach was not designed to measure human hearing ability; rather, it was used to test the effectiveness of telephone communication channels. A talker would read a list of nonsense syllables at one end of a telephone channel while a listener would record what he or she heard at the other end. If the telephone channel was clear, the report from the listener would match what the talker had read. Harvey Fletcher of Bell Laboratories reported using a similar technique for testing telephone channels using words and sentences. He eventually introduced the Western Electric 4A audiometer, which incorporated phonographic recordings to assess hearing loss (ASHA, 1988; Fletcher, 1929).

In 1947, Harvard Psychoacoustic Laboratory (PAL) developed the PAL Auditory Test Number 9 and PAL Auditory Test Number 12 (Hudgins, Hawkins, Karlin, & Stevens, 1947). One year later, they released the PAL PB-50 word list. Although the

PAL tests were originally designed to evaluate military communication equipment, they were quickly adapted for the clinical assessment of human hearing ability (Egan, 1948). Today, numerous forms of speech stimuli are used by clinicians as a routine part of their audiological assessment test battery. Martin, Champlin, and Chambers (1998) surveyed 500 certified audiologists, randomly selected from the membership directory of the American Academy of Audiology, to assess the most commonly used clinical practices. They found the most frequently obtained speech measurements to be the speech recognition threshold and word recognition score.

Routine Clinical Speech Tests

Speech Recognition Threshold

The speech recognition threshold (SRT) is obtained by presenting a list of words through an audiometer at various intensity levels. Spondaic words, or "spondees," are commonly used as the stimulus. A spondee is a two-syllable word presented in a way that both syllables receive equal emphasis (Hudgins, Hawkins, Karlin, & Stevens, 1947). A listener typically repeats each word after it is presented, and the lowest intensity level at which the listener can correctly repeat the words 50% of the time is recorded (ASHA, 1988; Hood & Poole, 1977). The SRT serves a variety of purposes. The basic purpose is to determine the threshold of hearing for speech (ASHA, 1988). Although pure-tone testing results in frequency specific thresholds, the SRT can provide an estimate of hearing sensitivity across a broad range of frequencies. This can be helpful when pure-tone testing is not possible, such as when age or mental ability prohibit reliable behavioral responses to tone stimuli. The second purpose of the SRT is to validate pure-tone thresholds. The spondaic SRT is closely correlated to the two and three frequency

pure-tone average (Martin & Jansen, 1985). The SRT can be particularly important in pediatric assessment or in other situations where the accuracy of the pure-tone thresholds is in question. The third purpose of the SRT is to provide a reference point for suprathreshold speech tests (Kamm, Morgan, & Dirks, 1983). Many speech tests require the presentation level to be at a certain sensation level. Generally, a sensation level for speech is referenced to the SRT (e.g. 40 dB re: SRT).

Word Recognition Score

Suprathreshold speech tests can be conducted using a variety of stimuli. Clinicians frequently test listeners using monosyllabic word lists, producing what is called a word recognition score (WRS) (Martin et al., 1998). The words are presented through an audiometer at an audible intensity level, which is held constant while the listener repeats each word. The percentage of words repeated correctly is recorded, and additional intensity levels may be tested if necessary.

The word lists used when obtaining WRS are usually phonemically ("phonetically") balanced lists. A phonemically balanced (PB) word list contains all the speech sounds of a language in their appropriate proportion, based on the frequency of each sound in a given dialect. For example, the PB-50 word list was phonemically balanced using a sample of 100,000 words of newsprint (Egan, 1948). Later lists, including the popular W-22 and NU-6 lists, were balanced using samples of spoken English as well as print (Hirsh et al., 1952; Lehiste & Peterson, 1959). The earlier PB lists were referred to as "phonetically balanced." However, when Lehiste and Peterson (1959) developed the word list that was eventually adapted to become the NU-6 list, they suggested that the term "phonemically balanced" be used to reflect that spoken English

is recognized more by its phonemes than phonetics. Most English PB word lists have been balanced for the American dialect (Tillman & Carhart, 1966), however other PB lists exist for other dialects. For example, Clark (1981) developed a list that is phonemically balanced for Australian English. While much attention has been paid to balancing PB lists, it should be noted that the importance of phonemic balance in word recognition testing has been questioned (Carhart, 1965; Tobias, 1964).

Word recognition scores are diagnostically useful for a variety of reasons. First, they are strongly correlated with pure-tone hearing thresholds (Jerger, Jerger, & Pirozzolo, 1991). As hearing threshold decreases, word recognition ability also decreases (Schwartz & Surr, 1979; Dubno et al., 1995). Due to this strong correlation, if word recognition scores are higher or lower than would be expected in comparison to the pure-tone thresholds, they can be suggestive of certain pathologies. In fact, the WRS can assist in differential diagnosis of cochlear and retrocochlear sites of lesion (Meyer & Mishler, 1985). An abnormally low WRS can also be indicative of auditory neuropathy (Gates, Feeney, & Higdon, 2003; Ptok, 2000).

In addition to the usefulness of WRS testing in differential diagnosis, WRS can estimate how well a listener understands speech in an everyday listening situation (Epstein, 1978). WRS can also be used to monitor changes in speech recognition ability over time, which can be useful in evaluating the effects of auditory therapy (Hnath-Chisolm, 1997) and surgical procedures (Teoh, Pisoni, & Miyamoto, 2004). Furthermore, WRS over time can quantify decreasing hearing ability due to aging (Dubno, Lee, Matthews, & Mills, 1997) or other progressive forms of hearing loss (e.g. hearing loss due to pre-natal cytomegalovirus exposure).

Word recognition scores are sometimes used to show the improvement seen with the use of a hearing aid. The speech recognition ability of a listener can be measured with and without the assistance of a hearing aid (Flynn, Dowell, & Clark, 1998; Gatehouse, Naylor, & Elberling, 2003). Although not a common clinical practice, scores are sometimes used in a research setting to show the effect of various hearing aid programs and settings. Word recognition scores are obtained using different programs and settings of a single hearing aid and then compared (Bentler, Tubbs, Egge, Flamme, & Dittberner, 2004). These measurements can demonstrate the communicative improvement provided by the amplification or digital signal processing and can greatly increase the listener's awareness of hearing aid benefit.

Word recognition scores, as well as a number of other speech measures, are critically important in the fitting of cochlear implants. The WRS is used to determine cochlear implant candidacy by quantifying speech perception ability (UK Cochlear Implant Study Group, 2004). Once a patient has been fitted with an implant, word recognition scores are used to compare postoperative and preoperative speech perception ability (Cullen et al., 2004; Svirsky, Teoh, & Neuburger, 2004), as well as to aid in programming the speech processor (Holden et al., 2005).

The clinical usefulness of the WRS depends heavily on the testing conditions and design of the test materials. For example, a score obtained at an average conversational level can provide an estimate of everyday communicative ability (Egan, 1948; Lehiste & Peterson, 1959); however, this score alone does not provide much information about the auditory site of lesion. When administering and interpreting speech tests, it is important that the interpretations made are appropriate for the given testing conditions (Thornton

& Raffin, 1978). For example, there are numerous methods used when obtaining word recognition scores, and many methods exist to provide unique clinical information. Unfortunately, what is commonly practiced in the clinic does not vary substantially (Martin et al., 1998). This suggests that certain inappropriate test methods are being used to gather diagnostic information, sometimes resulting in overextended and inaccurate test interpretations.

Common Clinical Practices

Single Presentation

When obtaining a word recognition score, most audiologists present one word list at a single intensity level per ear. This WRS, assumed to be the best score obtainable by the listener, is also referred to as "PB-max." PB-max is the best expected score of a PI function, which is WRS score as a function of intensity. Kamm, Morgan, and Dirks (1983) found that the majority of listeners obtain their best score when the stimulus level is between 30 and 40 dB above the speech recognition threshold. It is likely that PB-max would be obtained at this presentation level for most listeners. Presenting a word list at a single sensation level can produce an estimate of PB-max, thereby avoiding lengthy and sometimes impractical testing at multiple intensity levels. However, the benefit of decreased test time often comes at the expense of accuracy and could possibly lead to misdiagnosis.

Partial Word Lists

Martin and colleagues' survey of practicing audiologists found that 56% of all respondents routinely administer just 25 words of a 50-item word list when obtaining word recognition scores (1998). If four seconds are allowed for each test item, it would

take six minutes and forty seconds to obtain a single word recognition score for each ear with a 50-item word list. A 25-item word list would require just three minutes and twenty seconds. If multiple tests are to be conducted in each ear, a substantial amount of time would be required. The use of partial word lists can cut testing time significantly. The audiologist calculates the percentage correct from the 25 words administered and records it as the word recognition score.

Monitored Live-voice Presentation

Martin and colleagues (1998) found that 82% of respondents used monitored-live voice testing to obtain word recognition scores. The use of monitored live-voice (MLV) involves the clinician speaking aloud into a microphone, i.e. reading the WRS word list. The signal is then passed through an audiometer, where the intensity of the signal is controlled and presented to the listener. This method requires that the clinician carefully monitor the intensity level of his or her voice. This is commonly accomplished through visual attention to a VU (volume units) meter. Live-voice testing allows greater flexibility than the use of conventional recordings; a clinician can easily change the rate of presentation, randomize word lists, and repeat test items when desired (Stach, Davis-Thaxton, & Jerger, 1995). The use of recorded speech via compact disc player does not provide that level of flexibility. Although MLV testing is more flexible than recorded speech, clinicians sometimes report that recorded word recognition procedures are not regularly practiced, because they “take too much time” when balancing the demands of a busy clinic schedule (Hurley & Sells, 2003).

Evidence-based Audiology

While the use of MLV, half lists, and a single presentation level are common in clinical audiology, these procedures are not consistent with evidence-based audiology (Dirks, Kamm, Bower, & Betsworth, 1977; Martin et al., 1998; Meyer & Mishler, 1985; Tillman & Olsen, 1973; Thornton & Raffin, 1978). WRS obtained via these "short cut" procedures may provide some useful information, but the results have questionable validity for the intended purpose. In other words, WRS obtained with poor methodology will yield results which have questionable use in the audiology diagnostic process.

Multiple Presentations (Levels and Lists)

PB-max refers to the maximum word recognition score the listener is capable of obtaining, regardless of the intensity level of the stimulus. Since PB-max typically occurs 30-40 dB above the SRT (Causey, Hermanson, Hood, & Bowling, 1983; Hirsh et al., 1952; Kamm, Morgan, & Dirks, 1983; Tillman & Carhart, 1966;), in order to save time, many clinicians simply conduct word recognition testing at 30-40 dB SL re: SRT only (Martin et al., 1998). The score obtained at this intensity level may have some clinical value, but it cannot be confidently labeled as PB-max without testing at additional intensities, unless a score of 100% is obtained. Kamm and colleagues (1983) found 30-40 dB re: SRT to be adequate for PB-max in only 60% of subjects. This implies that with 40% of listeners, using 30-40 dB SL will result in a score which is less than PB-max. Therefore, it has been recommended that speech recognition ability be tested at multiple intensity levels in order to properly identify PB-max (Dirks et al., 1977).

Although testing at multiple intensities is recommended when determining PB-max, it is required when testing for the presence of rollover. Rollover occurs when a listener has a substantial decrease (40% of PB-max) in word recognition ability as the intensity of the stimulus increases. The lower word recognition score, obtained at the higher intensity, is labeled "PB-min." The presence of rollover has been closely correlated with retrocochlear pathologies and can help identify the site of lesion (Meyer & Mishler, 1985). If a clinician records a WRS at 30-40 dB SL re: SRT, he must then present another word list at a higher intensity to see if there is a change in word recognition ability. If the score remains constant, the procedure requires that the intensity level is raised until the stimulus reaches the listener's level of discomfort (Beattie & Zipp, 1990; Meyer & Mishler, 1985). Meyer and Mishler (1985) suggested that when screening for the presence of rollover, intensities greater than 90 dB HL should be used whenever possible. If a listener has an SRT of 20 dB HL and the clinician first presents a word list at 60 dB HL (40 dB SL re: SRT), and then raises the intensity in 10 dB steps, four separate word lists are presented just to reach 90 dB HL. Obtaining reliable PB-max scores, and properly screening for rollover, demand a considerable amount of time, which clinicians appear unwilling to spend on this test procedure (Martin et al., 1998).

In addition to improving diagnostic accuracy, there are counseling benefits to presenting more than one word list when performing word recognition testing. Testing at average conversational levels can provide the listener and family with an estimation of the impact the hearing loss has on the listener's everyday communication. This can also be used to evaluate the need for accommodations in a school or work setting. When

presented under TDH-49/50 headphones, average conversational level is approximately 50 dB HL (re: ANSI S3.6 1996). For listeners with hearing loss, this would typically be below the 30-40 dB SL used to estimate PB-max and would require an alternative presentation level.

Testing word recognition ability with both male and female talkers, in addition to multiple intensities, can be of benefit in a counseling session following the audiological evaluation. The acoustic spectra of male and female speech vary substantially. Wilson and colleagues (1990) found that when word lists were presented by female talkers, due to the higher frequencies present in the female voice, word recognition scores were significantly lower than when presented by male speakers at the same intensity level. This was likely due to the higher frequencies present in the female voice. The presentation level of the female voices had to be increased by 10-13 dB to produce scores equivalent to those of the male voices. Providing a listener and family with a quantitative representation of ability to understand male versus female voices can be helpful in explaining a perceived inconsistency of communicative difficulty when listening to different speakers.

Number of Words Presented

Word lists used to test word recognition ability are typically phonemically balanced. In an effort to save time, clinicians often present the first 10, 20 or 25 words of a list and calculate the percentage of correct responses (Martin et al., 1998). Although the list as a whole is balanced, sections of the word lists may contain a disproportionate amount of certain speech sounds (Elkins, 1970). Arbitrarily chosen partial word lists are not phonemically balanced.

In addition, reliability and validity are sometimes sacrificed when partial word lists are used. Variability is inversely related to the number of test items (Hagerman, 1976; Thornton and Raffin, 1978). When an insufficient number of test items is administered, the resulting score can be greatly influenced by only a slight increase or decrease in performance. Since the word lists are designed to represent everyday speech, decreasing the sample size by administering an insufficient number of test items can reduce the validity of the test.

Thornton and Raffin (1978) found that a 50-item WRS of 72% has a confidence interval of 54-86%. When a partial word list of just 25 items was used, a 72% word recognition score had an enlarged confidence interval of 48-92%. Using the above confidence intervals and a 25 word list, a score of 72% in one ear and 48% in the other would not result in a significant difference between ears. However, it is unlikely that the average clinician would consider both ears equal, possibly diagnosing the poorer ear as having a significant deficit. This variability can be particularly detrimental when comparing scores between different listeners, testing conditions, ears, or test sessions. A partial word list may give the clinician an idea of how the listener can understand speech, but the test results must be interpreted with caution when they are used for diagnostic purposes.

Recorded Word Lists Versus Monitored Live-voice

In an editorial commenting on Martin and colleagues' audiometric practices survey (1998), James Jerger described the popularity of MLV as "perhaps the single most depressing aspect" of the survey (1998). Although audiometers provide the clinician with the option of presenting word lists using either MLV or a recording, live-

voice presentation methods have been shown to be unreliable (Brandy, 1966; Penrod, 1979; Tillman & Olsen, 1973). The difficulty of a word list has been shown to vary significantly depending on the presenter (Penrod, 1979; Wilson et al., 1990). Different talkers can produce significantly different word recognition scores with the same listener and word list. Brandy (1966) even observed differences in scores with the same talker and same word list when tested on different days. Although the textual word lists are standardized, the variability of MLV prevents accurate diagnostic comparison between tests. Tillman and Olsen (1973) indicated that "no standardized test is possible unless recorded tests are employed, that is, because of talker differences tests administered via monitored-live-voice defy standardization." Apparently, the sentiment that the use of recorded word lists is too time consuming is strong enough to overshadow the necessity for clinical standardization (Martin et al., 1998).

Existing Methods to Decrease Test Time

Decreased Inter-stimulus Interval Recording

While many audiologists incorporate certain techniques to decrease speech testing time, a few methods have been formally developed to maintain reliability. Auditec of St. Louis distributes a recording of the NU-6 word list with a reduced inter-stimulus interval (Hurley & Sells, 2003). For listeners who respond quickly to the stimulus, this would decrease excessive waiting between word presentations; however, this recording is not suitable for listeners who need more time to respond. Unfortunately, whether a listener will respond quickly to a reduced inter-stimulus interval is not always apparent. If a listener is unable to keep up with an accelerated presentation method, either the pause button of the CD player must be used or the

regular interval recording must be played. Therefore, the use of a decreased inter-stimulus recording may actually increase test time with some listeners.

Abbreviated Word List in Order of Difficulty

An abbreviated word recognition protocol, incorporating word lists arranged in order of difficulty, has been developed. The purpose is to decrease the need to present all 50 words of a word list without decreasing reliability (Hurley & Sells, 2003). It was hypothesized that the use of the 10 or 25 most difficult words in a list could predict the percentage obtained if the entire 50 word lists was used. Hurley and Sells administered the Auditec recorded NU-6 word lists to 493 listeners. The errors were tabulated, and the words were then rearranged in order of difficulty. A clinical decision analysis was performed for each individual to determine the percentage that would have been obtained, had only the most difficult 10 or 25 words been presented. The researchers concluded that all four of the 10 word and 25 word NU-6 lists were able to differentiate which listeners would require the full 50 word test to obtain reliable results.

The protocol suggests that the 10 most difficult words are presented first. If more than one word is repeated incorrectly, the next 15 most difficult words are presented. If more than three of the first 25 words are incorrect, the remaining 25 words must be presented. Hurley and Sells found that this method reduced the need for full-list administrations by 25%. They predicted an average of 24 minutes saved for every 10 listeners. They also concluded that had the Auditec reduced inter-stimulus recording been used with all 10 listeners, testing time could have been reduced by a full 60 minutes.

Computer-based Presentation Method

Stach, Davis-Thaxton, and Jerger (1995) developed and tested a computer-based approach to improving the efficiency of speech audiometry. With financial support from Apple Computer, a software program was written to present digitized speech in a manner simulating live-voice presentation. Four word lists from the PAL PB-50 test were digitally recorded from magnetized tape and then stored on the computer's hard drive. The computer system was programmed to automatically present each word from a pre-selected list in a random order. After each presentation, if the listener repeated the word, the clinician pressed a key or clicked the mouse to indicate a correct or incorrect response. The computer system then presented the next word from the same list, until the entire list was presented. The total percentage correct was automatically calculated after each response. The software also had the capability of presenting the words in a continuous manner, i.e. without clinician involvement, using an inter-stimulus interval fixed at 2 sec. This option simulated the traditional recorded speech presentation method.

Clinical testing was performed on 981 patients chosen from a caseload at The Methodist Hospital Audiology Service and the Neurosensory Center of Houston. Each participant was tested with the clinician-controlled computer presentation method in one ear and the computer-controlled (fixed, 2-second interval) presentation method in the other ear. The ears were randomly assigned to each method. The results indicated an average decrease in testing time of 22% when the clinician-controlled computer presentation method was used compared to the computer controlled-fixed interval. The researchers concluded that a computer-based presentation system was a promising way

to improve test efficiency, while maintaining the advantages of recorded speech. The software was never commercially distributed.

Future Computer-based Possibilities

The computer-based, clinician-controlled, method described above resulted in a significant decrease in test time, thus the concept of computer-based speech audiometry has at least one advantage compared with current techniques; however, the system Stach et al. investigated did not advance beyond the research stage. Given the advancement and increased availability of technology since that time, it seems appropriate to readdress the possible integration of computers and speech audiometry and to further develop the concept of computer-based speech audiometry.

One possible way to take advantage of the research accumulated on recordings of speech materials, along with the advances in digital technology, is to rip the digital code directly from the standardized CD recordings. The code could then be segmented into smaller parts for use in a PC presentation method. If the original recording is standardized, as long as the signal integrity is maintained, it is likely that the normative data of the original recording would also apply to the segmented digital recording. Assuming that a patient-directed reduction of inter-stimulus interval does not significantly affect the WRS, as suggested by Stach et al. (1995), then digitally segmented speech lists provide a way to broaden the clinical usefulness of the method without the need for extensive normative research.

Many of the methods used to increase the efficiency of speech audiometry that have been shown to be reliable in past research can also be applied to computer-presented digital speech. For example, the use of partial word lists presented in order of

difficulty and the use of a flexible inter-stimulus interval have been shown to increase efficiency without sacrificing reliability (Hurley & Sells, 2003, Stach et al., 1995). If speech-audiometry software is developed which includes pre-recorded lists which permit digitized words to be presented in order of difficulty with flexible inter-stimulus interval, this could provide a means to improve the overall efficiency of speech audiometry without sacrificing validity and reliability.

Statement of Purpose

The ability to present recorded word lists, on demand, one word at a time, has the potential to drastically improve speech testing efficiency. The use of a personal computer could provide the benefits of both recorded speech and monitored live voice presentation methods. PC based speech audiometry could permit the use a standardized recording while maintaining the speed and flexibility of MLV. The purpose of this study was to explore the use of a software program to present segmented versions of existing recorded word lists to be used in word recognition testing.

Chapter 3

Development and Technology

Development of Speech Presentation Software

A prototype software program was developed to explore whether recorded word recognition stimuli could be accurately presented with the increased speed and flexibility associated with MLV testing, using recorded word lists which have been studied extensively with regard to their validity, reliability, and homogeneity. The primary intent of the software was to permit the presentation of recorded words with each word being presented at the clinician's command. This would allow the clinician and client to set the presentation pace, thereby, in most cases, reducing the time required to obtain a word recognition score.

Microsoft Windows was chosen as the operating system due to its extreme popularity. A review of Windows compatible programming languages and platforms was conducted. Microsoft Visual Basic.NET (VB.NET) was chosen as the programming language for the development of the software, because of its ease of use, built-in multimedia functions, and object-based programming structure. A prototype program was developed to enable the clinician to present an NU-6 word list one word a time and to provide the clinician with the flexibility to skip words, repeat words, and compute the percentage of correct responses after each word. The main components of the software are the graphical user interface, the programming code, and the recorded media.

Graphical User Interface

The graphical user interface consists of a display of all the words in the chosen word list, a score display, and eight buttons to control the programming code (See Figure 1). Fifty text boxes allow for a maximum of 50 words to be displayed for a given word list. The score display area includes a count of the total number of correct responses, the total number of words presented, and an automatically calculated percentage of correct responses.

The buttons are labeled: Start/Restart, Calibrate, Correct, Incorrect, Repeat, Skip, Finish(Correct), and Finish(Incorrect). Each button performs a different function, when selected with the mouse cursor. "Start/Restart" presents the first word of the selected word list and resets the score display. "Calibrate" presents the calibration tone associated with the word list. "Correct" adds 1 to the total number of correct responses, adds 1 to the total number of words presented, and presents the next word in the list. "Incorrect" adds 1 to the total number of words presented and presents the next word in the list. "Repeat" presents the last word presented. "Skip" presents the next word in the list without changing the total number of words presented. "Finish(Correct)" adds 1 to the total number of correct responses and adds 1 to the total number of words presented, without presenting the next word in the list. "Finish(Incorrect)" adds 1 to the total number of words presented only, without presenting the next word in the list.

Programming Code

The programming code consists of textual representations of the binary instructions for the computer to follow. The Visual Basic code of the prototype software

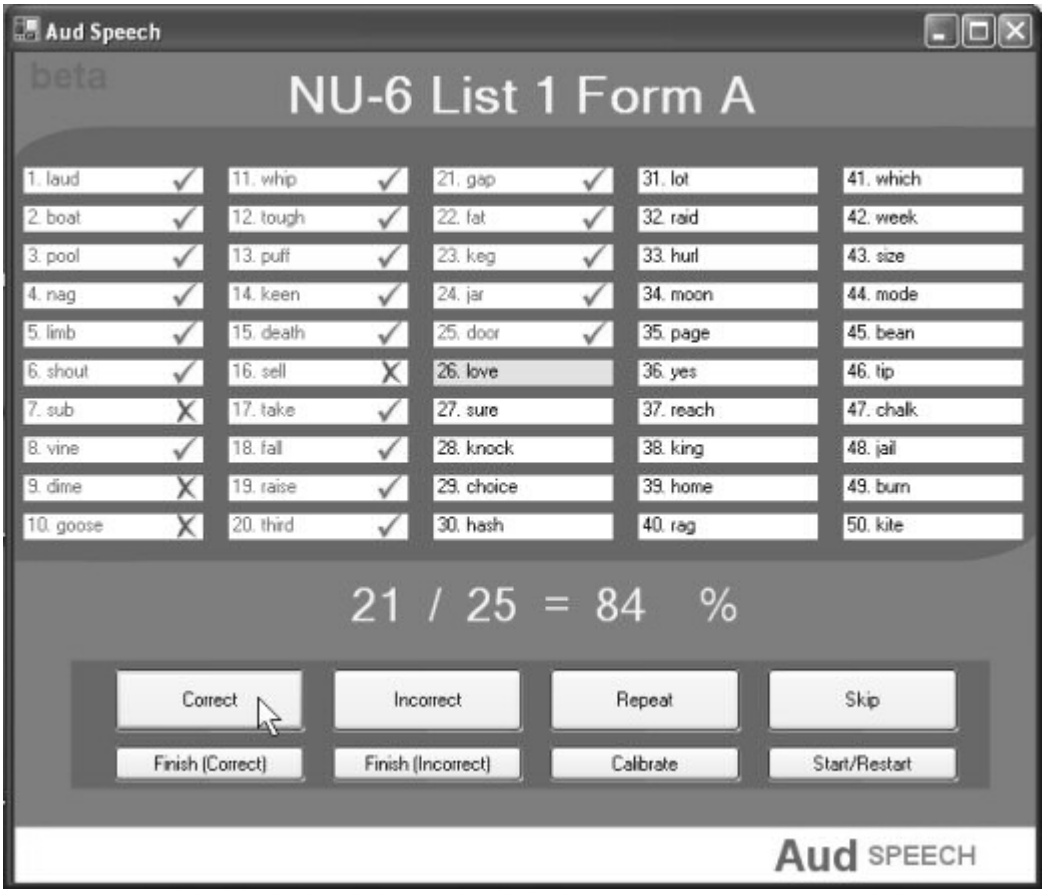


Figure 1. Prototype Software Program Graphical User Interface

program is included in Appendix A. Different sections of code are triggered by different events. For example, when the program is initially started, a certain section of code is run, which displays the graphical user interface and makes it available to the user. The user is then able to choose which sections of code are run, through the use of the interface buttons.

Sometimes a section of code included in the software program may call another section of code already stored within the Windows environment. For example, when the speech software presents the sound file for a recorded word, it uses the "winmm.dll." This is a collection of code, built into Windows, which is used to present various forms of digital multimedia. The speech software uses this code to send the appropriate instructions to the computer's sound card. The sound card then converts the digital information into an analog signal. The analog signal is then sent through an audiometer, and it is eventually converted into acoustical energy via an earphone transducer.

Recorded Media

The speech software uses digital audio files created from existing word list recordings. Each file contains just one word. The prototype presents the NU-6 List 1 Form A from the Department of Veterans Affairs Speech Recognition and Identification Materials, Disc 1.1 (1991). A 16 bit, 44100 Hz, mono WAV copy of the NU-6 List 1 Form A track was created using Exact Audio Copy V0.95 beta 3. This single WAV file included all the NU-6 List 1 Form A words as well as the recorded silence between each word. Adobe Audition 1.5 was used to segment the WAV file. The inter-stimulus silence was removed, and each word was saved as an individual WAV file. The calibration tone was also saved as a separate WAV file. The WAV audio format was

chosen because there is theoretically no degradation of signal when ripping a WAV copy of an audio CD (Balo et al., 2004). Compared to other audio formats, the WAV file is larger, but its fidelity is superior to most compressed formats such as MP3 (Balo et al., 2004). The use of WAV files guarantees that the recordings used by the software are the same quality as the original CD. However, the software is reliant on both the computer's sound card and the audiometer system to generate the audible signal. Audiometer fidelity is assured using standards established by the American National Standards Institute (ANSI S3.6-1996), but the fidelity of the sound card is still in question.

Sound Card Technology

A personal computer uses a sound card to manipulate and translate sound signals. A sound card serves two primary functions. It converts analog signals into digital and digital signals into analog. Both functions are extremely important since humans cannot understand digital signals, and computers cannot understand analog signals. The sound card essentially acts as a sound signal interpreter.

Analog-to-digital Conversion

Most sound cards have a microphone/line-in port. This port typically leads to the Analog-to-digital converter (ADC). When someone speaks into a microphone, the acoustic signal is converted to an electrical signal. The electrical signal travels through the microphone cord and into the ADC. The ADC then converts the electrical analog signal into a digital signal that the computer can understand. The computer could then be used to display a spectral analysis of the sound, record the sound, or perform a variety of other functions.

Digital-to-analog Conversion

A computer's sound card is also used in the production of sound. A digital signal is converted into analog using the digital-to-analog converter (DAC). The electrical analog signal exits the speaker port and typically travels through a cord to a speaker. A transducer then converts the electrical analog signal into an audible acoustic signal. The acoustic signal can then be decoded by the human auditory system.

When sound signals are converted between analog and digital formats, there is always some degree of change in the signal. For example, if a signal went from analog to digital and back to analog again, the resulting signal would not be identical to the original. Any change in the original signal through the conversion process is called distortion. Sound card distortion presents itself in both the amplitude and time domain of sound. Analog signals are continuous waveforms, while digital signals are simply numerical representations of those waveforms. The digital signals store information at frequent intervals. In contrast, the analog signals contain information at infinite intervals. Various sound cards are able to convert digital signals to analog and visa versa with varying degrees of success. Generally, the less expensive the sound card is, the lower the sound quality.

Electro-acoustic Measurements

Two common measures of sound card quality are total harmonic distortion plus noise (THD+N) and signal to noise ratio (SNR). THD+N is a measurement of signal distortion and noise. A common formula for calculating THD+N is:

$$\%THD+n=\frac{\sqrt{H_2^2+H_3^2+\ldots+H_N^2+n^2}}{\sqrt{H_1^2+H_2^2+H_3^2+\ldots+H_N^2+n^2}}\times 100$$

where 2 thru N represent the power levels of the 2nd through Nth harmonics and term 1 represents the power level of the fundamental frequency (Guidorzi, 2005).

The SNR reflects the amplitude of the ratio of the power of a desired signal and the power of the noise floor. This is calculated in a variety of different ways, but often uses the same variables as the THD+N algorithm.

These measures, as well as others, are usually printed on the sound card packaging as part of the sound card specifications. However, once a sound card is installed in a computer, its sound quality can be affected by its operating environment (Audio Engineering Society Standards Committee, 2000). Neighboring components can have a degrading effect on sound card quality. Therefore, practical computer sound quality measurements must be taken after sound card installation.

As sound card technology rapidly changes and improves standards for sound quality measurements are constantly being developed. Sound card quality is generally measured as other audio equipment is measured, with a few notable differences. For example, when testing component audio equipment, the signal to noise ratio is measured with and without a signal present. For example, when testing a phonograph, the sound pressure level of a recorded signal is measured. Then the sound pressure level of the noise the phonograph itself generates is measured, with no recorded signal. This measurement is made when the turntable is spinning and the needle is raised. The calculated difference between the two measurements is the SNR.

This type of SNR measurement technique is impossible to use with computer sound cards. When a sound card is not actively generating sound, its DAC circuit is inactive. Therefore, no noise is generated by the sound card. However, the sound card does generate noise while presenting a desired signal. In order to measure this noise level, it has been suggested that the sound card generate an extremely quiet signal during the silent condition (Audio Engineering Society Standards Committee, 2000). As long as the generated signal is below the noise floor, it will not affect the calculated SNR.

Many sound card quality measurements can be made using a software program and a reference sound card. A reference sound card is assumed to have very high fidelity, and is used to capture the signal in question. The signal is then analyzed by the measurement software. If the reference sound card's fidelity is superior to those being measured, very accurate measurements can be made for a fraction of the cost of stand-alone acoustic analysis equipment.

Sound quality can be measured in a variety of ways, using a variety of methods and tools. While the human listener's perception of sound quality is often the ultimate quality measurement, it is extremely difficult to quantify and compare among different listeners. In addition, certain changes in sound quality may not be perceived consciously by all listeners. The ANSI standards require that certain audiometer sound quality measurements are objectively measured using a spectrum analyzer (American National Standards Institute, 1996).

Chapter 4

Research Methodology and Results

The present study incorporated two separate experiments involving the investigation of personal computer sound quality and one observation involving a cassette player and CD player. In Experiment 1, the sound quality of a group of personal computers was measured, and the computers were ranked in order of THD+N. The purpose of Experiment 1 was to identify a poor quality computer for use in Experiment 2. Experiment 2 involved using a computer with poor sound quality, as identified in Experiment 1, in combination with two audiometers clinically in use. Acoustic measurements were made to determine if the computer complied with current ANSI standards for speech stimuli.

Experiment 1 – Sound Quality Measurements

Method

Twenty computers were evaluated for this study. They were selected from a variety of settings including: personal homes (n=7), clerical and administrative offices (n=6), an audiology clinic (n=3), a school cafeteria (n=1), and a library (n=3). Both desktop computers (n=16) and laptop computers (n=4) were represented in the study. Six sound card manufacturers, seven sound card models/drivers, seven computer brands, and two desktop computers assembled from various components by their owner were included in the study (See Table 1). Most of the computers contained sound cards integrated into the motherboard, with only three having a separate PCI sound card.

The computer's sound cards were evaluated for total harmonic distortion plus noise (THD+N). All measurements were made without removing the test computers from their usual setting. Laptop computers were measured while plugged in to an AC outlet. A 1000 Hz pure tone was used as the test signal. The test tone was created using Adobe Audition 1.5 with the following parameters: 60 second duration, 1000 Hz base frequency, sine wave, 0 dB, 16 bit, mono, 44100 Hz. The test tone was transferred from a CD-ROM or USB flash memory drive onto the test computer's hard drive, and then presented using Windows Sound Recorder. The output levels of the sound card were adjusted using Windows Mixer. In most cases, the output of the WAV circuit was set to maximum, and the main volume control was adjusted so that the relative peak amplitude of the test tone was as close to -10 dB as the mixer would allow, as measured by the reference sound card and software. In the event that the output of the test computer was excessively noisy or distorted, as viewed on the reference computer, the main volume control was adjusted to maximum and the WAV output was attenuated until the appropriate level was obtained. If the computer had additional volume controls independent of Windows Mixer, such as multimedia controls on the keyboard, they were set to maximum.

The output from the test computer was connected to the input of reference computer sound card using a shielded mini-plug to mini-plug cable. An IBM ThinkPad R40 with a PCMCIA Sound Blaster Audigy 2 ZS Notebook sound card was used to collect the signal from the test computers. This sound card was chosen as the reference sound card because of its superior fidelity, with the assumption that it would have less noise and distortion than most of the sound cards being tested. Spectral measurements

and calculations for THD+N were made using SpectraPLUS FFT Spectral Analysis System version 2.32.04, with the following parameters: Mode: Real-Time, View: Spectrum, Amplitude axis: Logarithmic, Frequency axis: Linear, Standard Weighting: Type A, Mic compensation: None, Sampling rate: 44100Hz, FFT size: 8192, Decimation: 1, Averaging size: Infinite, Smoothing Window: Hanning, 16 bit, Mono. The signal was averaged for approximately 30 seconds, and the displayed THD+N value was recorded.

Results

The THD+N values and sound quality ranking of the test computers are shown in Table 1. The mean THD+N measurement obtained was 0.03058% (SD = 0.037527%). The computer with the best sound quality (Dell Optiplex G70, ID: GYHT851) had a THD+N value of 0.00443%. The poorest computer (Compaq Armada V300, ID: DM6458) had a THD+N value of 0.13515%, and the second poorest computer (Dell Dimension 4500, ID: GZGYJ11) had a THD+N value of 0.08202%. The frequency responses for these computers are represented in Figure 1 through Figure 3. (See Appendix for the frequency response of the other computers.)

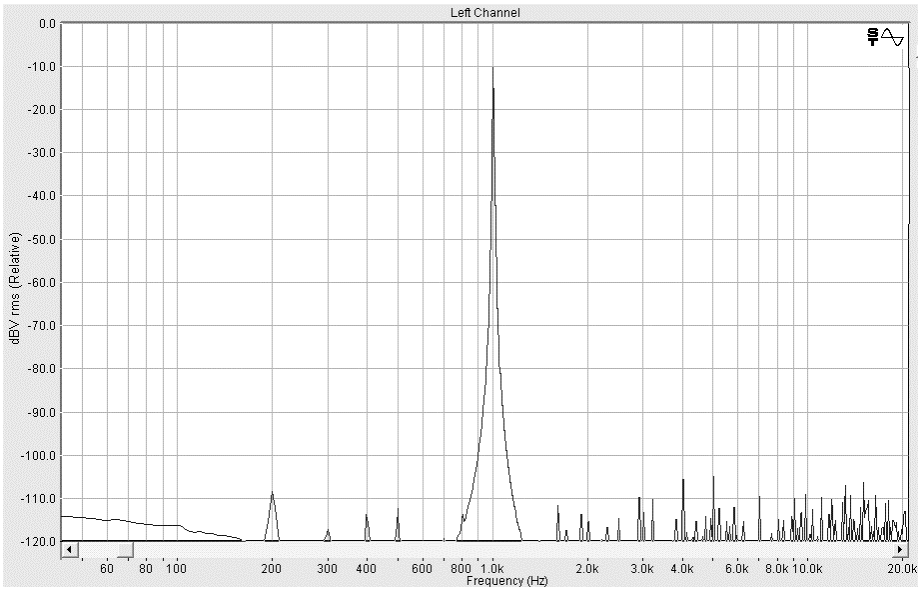


Figure 1. Frequency Response of Dell Optiplex G7, ID: GYHT851

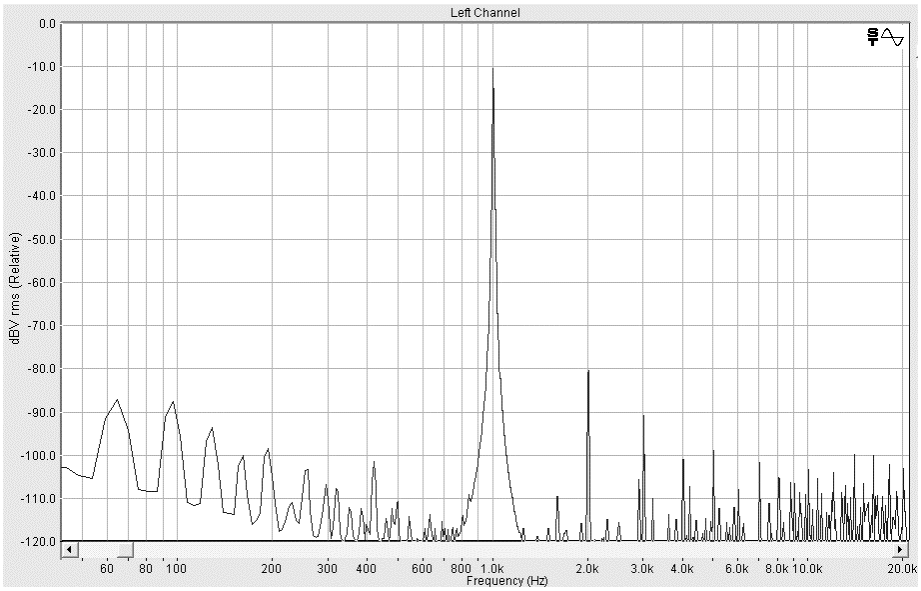


Figure 2. Frequency Response of Compaq Armada V300, ID: DM6458

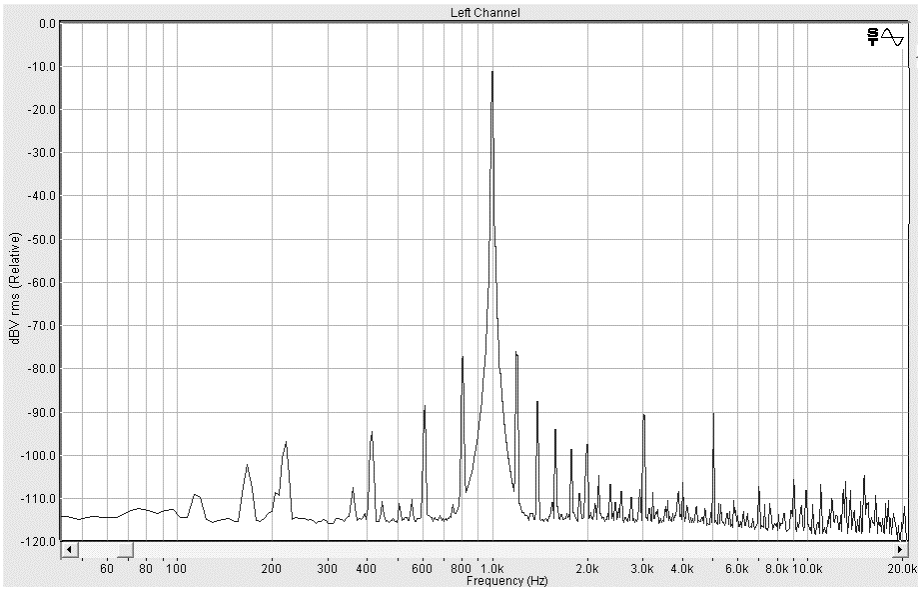


Figure 3. Frequency Response of Dell Dimension 4500 (ID: GZGYJ11)

Table 1

Computer Sound Quality (Total Harmonic Distortion Plus Noise)

Rank	ID	PC Brand	PC Model	Sound Card Manufacturer	Sound Card Model/Driver	THD+N
1	GYHT851	Dell	Optiplex GX270	Analog Devices, Inc.	SoundMAX Integrated Digital Audio	0.00443%
2	61JT851	Dell	Optiplex GX270	Analog Devices, Inc.	SoundMAX Integrated Digital Audio	0.00449%
3	BGX5431	Dell	Optiplex GX260	Analog Devices, Inc.	SoundMAX Integrated Digital Audio	0.00462%
4	GZHT851	Dell	Optiplex GX270	Analog Devices, Inc.	SoundMAX Integrated Digital Audio	0.00484%
5	44J9T71	Dell	Optiplex GX280	Analog Devices, Inc.	SoundMAX Integrated Digital Audio	0.00529%

Rank	ID	PC Brand	PC Model	Sound Card Manufacturer	Sound Card Model/Driver	THD+N
6	83VW091	Dell	Optiplex GX520	Analog Devices, Inc.	SoundMAX Integrated Digital Audio	0.00583%
7	7NBC451	Dell	Optiplex SX270	Analog Devices, Inc.	SoundMAX Integrated Digital Audio	0.00656%
8	4000206T	Insignia	D400	Realtek	Realtek AC'97 Audio	0.00696%
9	56N2889	Dell	Inspiron 600m (Laptop)	SigmaTel	SigmaTel C-Major Audio	0.00799%
10	5080MWK	HP	dx2000 MT	Analog Devices, Inc.	SoundMAX Integrated Digital Audio	0.01049%
11	40900DR	HP-Compaq	d220 MT	Analog Devices, Inc.	SoundMAX Integrated Digital Audio	0.01058%
12	402061D	HP-Compaq	d330 uT	Analog Devices, Inc.	SoundMAX Integrated Digital Audio	0.01096%

Rank	ID	PC Brand	PC Model	Sound Card Manufacturer	Sound Card Model/Driver	THD+N
13	4CK6014	Dell	Latitude D600 (Laptop)	SigmaTel	SigmaTel C-Major Audio	0.01309%
14	33999A	Toshiba	Satellite 325CDS (Laptop)	Yamaha	Yamaha OPL3-Sax WDM Driver	0.02828%
15	A99999	none	Epox Motherboard	Philips	PSC706 Audio	0.05374%
16	5500JG9	HP	dx5150 MT	Realtek	Realtek AC97 Audio	0.05650%
17	B99999	none	Mach Speed Motherboard	Realtek	Realtek AC'97 Audio for VIA ® Audio Controller	0.07984%
18	906085	Vsquared	None listed	Yamaha	Yamaha Native DS1 WDM Driver	0.08002%
19	GZGYJ11	Dell	Dimension 4500	Philips	Acoustic Edge Audio	0.08202%
20	DM6458	Compaq	Armada V300 (Laptop)	ESS Technology, Inc.	Maestro MPU401 Devices	0.13515%

Experiment 2 – Audiometric Calibration

Method

The test computer with the worst quality measurements (Compaq Armada V300, ID: DM6458) was used to replace the recorded speech stimulus device of a Madsen Aurical audiometer currently in use clinically. Due to the poor results found when this computer was assessed (See results section for Experiment 2), the second poorest computer (Dell Dimension 4500, ID: GZGYJ11) was also tested. This computer was tested in conjunction with the Madsen Aurical audiometer as well as a GSI-61 audiometer. For each computer and audiometer combination, the output of the computer's sound card was connected to the input of the audiometer using a shielded mini-plug to composite stereo cable. TDH-49 supra-aural earphones with MX-41/AR cushions were used. The computer and audiometer remained on the clinician's side of a double-walled, sound attenuating booth with the earphones routed to the listener's side. The left earphone was removed from the headband and coupled to a Brüel & Kjær Artificial Ear Type 4152 with a Brüel & Kjær Condenser Microphone Type 4144. A Brüel & Kjær Frequency Analyzer Type 2131 was used to measure the acoustic output from the earphone.

Overall distortion.

The output level of the computer was again adjusted, using the procedure in Experiment 1, to a relative peak amplitude of approximately -10 dB. The input sensitivity of the audiometer was then adjusted so a 1000 Hz test tone, presented by the test computer, peaked at 0 dB on the audiometer's VU meter. The test tone was a WAV file, created using Adobe Audition 1.5, with the following parameters: Sample Rate:

44100, Channels: Mono, Resolution: 16-bit, Base Frequency: 1000 Hz, Modulate by: 0 Hz, General Flavor: Sine, Duration: 60 seconds.

The WAV stimulus was presented via the test computer using Windows Recorder. The attenuation of the audiometer was adjusted to allow for an approximately 120 dB SPL output, as measured at the earphone. The intensities of the fundamental frequency (F1) and two harmonics (H1, H2) were measured using the digital frequency analyzer with a 1/3 bandwidth filter. F1, H1 and H2 measurements were also obtained using 250 Hz, 500 Hz, 2000 Hz, and 4000 Hz test tones. All tones were WAV files generated by Adobe Audition 1.5, using the same parameters as listed above, with the exception of the change in base frequency.

Noise.

The noise of the speech audiometry circuit was measured with and without a stimulus. A 1000 Hz tone (the same WAV file used to measure the overall distortion) was presented by the test computer using Windows Recorder. The audiometer attenuator was set to 100 dB HL. The A-weighted sound pressure level was measured at the earphone using the digital frequency analyzer with a 1 octave bandwidth filter. A silent WAV file was then presented by the test computer with the audiometer attenuation at 100 dB HL. The silent WAV file was created using Adobe Audition 1.5's silence generator with a duration of 60 seconds. The sound pressure level was again measured at the earphone in the same manner as described above. The difference in SPL between the two outputs was calculated.

Results

Overall Distortion

The overall distortion of the Compaq computer with the Aurical audiometer was acceptable under ANSI S3.6-1996 Section 5.6.4. The standard requires that the fundamental of each output signal is at least 25 dB above the level of all higher harmonics. The computer and audiometer produced no harmonics with an intensity greater than 25 dB below the fundamental (See Table 2). The smallest difference between any fundamental and harmonic was at 250 Hz, with a difference between F1 and H1 of 46.4 dB.

The overall distortion of the Dell Dimension 4500 with the Aurical audiometer was acceptable under the ANSI standard. The computer and audiometer produced no harmonics with an intensity level greater than 25 dB below the fundamental (See Table 3). The smallest difference between any fundamental and harmonic was at 4000 Hz, with a difference between F1 and H1 of 47.8 dB.

The overall distortion of the Dell Dimension 4500 with the GSI-61 audiometer was acceptable under the ANSI standard. The computer and audiometer produced no harmonics with an intensity level greater than 25 dB below the fundamental (See Table 4). The smallest difference between any fundamental and harmonic was at 4000 Hz, with a difference between F1 and H1 of 48.8 dB.

Noise

The noise levels obtained for each audiometer and computer combination are displayed in Table 5. The noise level of the Compaq computer and Aurical audiometer was not acceptable under ANSI S3.6-1996 Section 5.8. The standard requires that the

Table 2

Fundamental Frequency and Harmonic Intensity Levels for Aurical Audiometer and Compaq Computer (dB SPL)

Test Tone	F1	H1	H2
250 Hz	119.1	72.7	70.5
500 Hz	119.4	71.1	65.4
1000 Hz	119.9	67.6	61.1
2000 Hz	119.9	70.6	61.1
4000 Hz	117.9	71.1	< 60

Table 3

Fundamental Frequency and Harmonic Intensity Levels for Aurical Audiometer and Dell Computer (dB SPL)

Test Tone	F1	H1	H2
250 Hz	116.7	61.8	< 60
500 Hz	119.7	64.4	< 60
1000 Hz	119.9	64.8	< 60
2000 Hz	116.6	64.0	< 60
4000 Hz	117.7	69.9	< 60

Table 4

Fundamental Frequency and Harmonic Intensity Levels for GSI-61 Audiometer and Dell Computer (dB SPL)

Test Tone	F1	H1	H2
250 Hz	115.9	61.0	< 60
500 Hz	119.4	64.8	< 60
1000 Hz	119.9	65.1	< 60
2000 Hz	115.9	64.0	< 60
4000 Hz	116.0	67.2	< 60

sound pressure level measured at 100 dB HL, with a 1000 Hz pure-tone, be at least 50 dB greater than the sound pressure level obtained with a silent condition. With the audiometer attenuation dial set at 100 dB HL, a sound pressure level of 115.1 dB A was measured at the earphone. In the silent condition, with the computer presenting a silent WAV file, the output level at the earphone was 71.8 dB A. This represents a difference of 43.3 dB, which is 6.4 dB more than permitted by the standard.

The Dell Dimension 4500 with the Aurical Audiometer produced an acceptable noise level. With the audiometer attenuation dial set at 100 dB HL, a sound pressure level of 119.9 dB A was measured at the earphone (See Table 6). In the silent condition, the output level at the earphone was 48.7 dB A. This represents a difference of 71.2 dB, which is 21.2 dB greater than the 50 dB minimum difference acceptable under the ANSI standard.

The Dell Dimension 4500 with the GSI-61 Audiometer also produced an acceptable noise level. With the audiometer attenuation dial set at 100 dB HL, a sound pressure level of 119.9 dB A was measured at the earphone (See Table 7). In the silent condition the output level at the earphone was 35.9 dB A. This represents a difference of 84.0 dB, which is 34.0 dB greater than the 50 dB minimum difference required by the ANSI standard.

Table 5

Fundamental Frequency and Harmonic Intensity Levels for Audiometer and Computer Combinations

Audiometer	Computer	1000 Hz	Silence	Difference
Aurical	Compaq	115.1 dB A	71.8 dB A	43.3 dB
Aurical	Dell	119.9 dB A	48.7 dB A	71.2 dB
GSI-61	Dell	119.9 dB A	35.9 dB A	84.0 dB

Observation - Cassette and CD Player Quality Measurements

Method

The 1000 Hz calibration tone from the Auditec of St. Louis Spanish Trisyllables for SRT cassette tape was presented using a Panasonic 607 stereo cassette deck. A stereo composite to mini-plug adapter was used to connect the composite stereo output cable of the cassette player to the input of the reference sound card.

The 1000 Hz calibration tone from the NU-6 List 1 recording on the Department of Veterans Affairs Speech Recognition and Identification Materials, Disc 1.1 (1991) was used to measure the sound quality of a Sony CDP-CE535 compact disc player. A stereo composite to mini-plug adapter was used to connect the composite stereo output cable of the cassette player to the input of the reference sound card.

The output levels for both the cassette and CD player could not be adjusted. With the exception of adjusting the output level, the same procedure outlined in Experiment 1 for the sound card quality measurements was followed.

Results

The frequency responses for the Cassette and CD player are represented in Figure 4 and Figure 5, respectively. The line output of the cassette player initially produced a relative dB reading of less than 0 dB, with no attenuation of the reference sound card input. In other words, the output of the cassette player was at least 10 dB stronger than the output levels set on the computer sound cards in Experiment 1. To compensate for the difference and allow measurement without clipping, the input signal of the reference sound card was attenuated by 5 dB. The cassette player then produced a relative peak amplitude of -4.18 dB and a THD+N measurement of 2.11013%. The CD player

produced a relative peak amplitude of -2.86 dB and a THD+N measurement of 0.23716% (See Table 6).

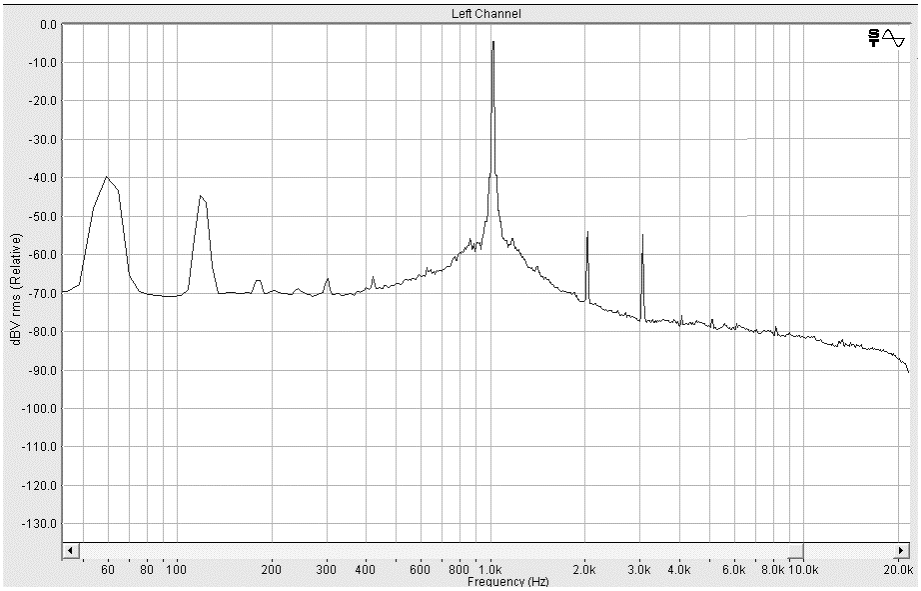


Figure 4. Frequency Response of Cassette Player

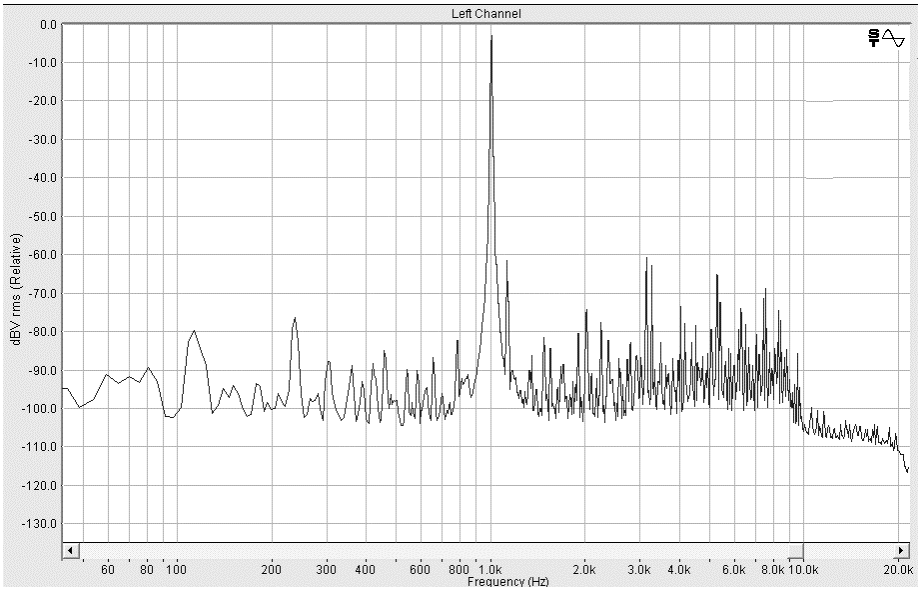


Figure 5: Frequency Response of CD Player

Table 6

Cassette and CD Player Sound Quality Measurements

Device	Relative Peak Amp.	THD+N
Cassette	-4.18 dB	2.11013%
CD Player	-2.86 dB	0.23716%

Chapter 5

Discussion

The primary purpose of this study was to determine if the sound quality of average computer sound cards was good enough to permit the use of computer-presented stimuli for the purpose of audiometric speech testing. Ninety five percent (19/20) of the computers tested in this study were adequate for clinical use based on the overall distortion and noise criteria. Based on the diversity of computer types used for this study (e.g. laptops and desktops, various sound cards) it is suggested that most personal computers would be acceptable for use for speech audiometry using computer-presented speech, providing that the computer's output level is carefully controlled.

Research Observations

Of the twenty computers tested, only one (Compaq Armada V300) produced a signal too poor to be used clinically under the ANSI standard. Although the level of distortion was well within the range permitted (See Table 2), the noise level was 3.3 dB above the maximum permissible noise level. This laptop computer produced a noise level of 71.8 dB, which presented itself as a clearly audible hissing sound to a listener with normal hearing. Compared to what an average listener would consider acoustically "quiet," the ANSI standard appears to be fairly liberal in the amount of noise that is permissible. This may be to accommodate the use of older, analog devices such as cassette players, which generally produce much more noise than digital devices.

Cassette and CD player quality measurements were made to compare the THD+N of traditional speech audiometry equipment with the THD+N produced by the computers. The cassette player's THD+N (2.11013%) and the CD player's THD+N (0.23716%) were both higher than that of even the worst computer (0.13515%). This suggests that an increase in distortion levels should not be a concern when considering the use of computer-based speech audiometry. The noise levels of the cassette and CD player in conjunction with an audiometer were not measured for this study. However, both units were being used with audiometers calibrated within the last year. In spite of its high level of THD+N, the cassette player likely qualified under the ANSI audiometer noise standard because the noise included in the recording and produced by the tape is not taken into account.

Prior to data collection, it was anticipated that the sound quality of the laptop computers would be substantially poorer when compared to the desktop computers. This was assumed because external sound controller manufacturers often promote their products as an upgrade for the low quality sound cards built into most laptop computers (e.g. Laptop Audio Upgrade Kit – Turtle Beach, n.d.). The results of the present study found eight desktop computers had poorer sound quality than the highest ranked laptop (See Table 1).

Another speculation was that the computers with aftermarket sound cards would have superior sound quality. This was based on the assumption that paying extra for a sound card would only be logical if it improved the quality of the system; however, this was not the case. Two of the desktop computers contained third-party Philips sound cards (Dell Dimension 4500 & Epox motherboard). The owners of the Dimension 4500

purchased the Philips sound card as an upgrade to the sound card built into the motherboard of their computer. The sound quality of the Philips sound card ranked the Dimension 4500 nineteenth out of twenty. Ironically, all seven of the other Dell desktop computers, equipped with their onboard sound cards, were ranked first through seventh in sound card quality. It should be noted that the sound quality rankings used in this study were based solely on the THD+N of the sound card's WAV output circuit. The owners of the Dimension 4500 purchased the Philips sound card primarily for its MIDI production capabilities. MIDI sound is produced using recorded instrumental sounds, often stored in the memory of the sound card. The Philips sound card was designed to produce more realistic instrumental sounds than the factory-included Dell sound card.

Three of the computers tested were of the same make and model (Dell Optiplex GX270, ID: GYHT851, 61JT851, GZHT851). These three "identical" computers produced different spectral outputs (See Appendix B). Two of them (ID: GYHT851, 61JT851) had very similar spectral outputs, with only slight variations in the intensity of the noise at certain frequencies. However, the third Optiplex GX270 (ID: GZHT851) produced a signal with much more low frequency noise, when compared to the other two PCs of the same model. A portion of this difference can be explained by the possibility of very subtle variations in the sound cards' structural composition. The majority of the difference, however, is likely explained by variations in secondary noise sources. The spectral output represents a combination of the desired signal as well as fan noise (case, CPU, power supply, video card), hard drive noise, 60 Hz AC noise, electrical interference from the environment and adjacent equipment, distortion from the original recording, and additional noise produced by the many components contained inside a computer case.

Although the three computers were designed identically, they were each plugged into separate wall outlets and were adjacent to different pieces of equipment.

A few of the computers produced a spectral output with an excessive amount of 60 Hz noise (Armada V300 & Epox Motherboard). Both of these computers produced a spectral output with a prominent peak at 60 Hz greater than 30 dB above the measurement floor. Each computer was retested in a second wall outlet. There was no substantial difference in the amount of noise. This suggests that the cause of the noise was not associated with the power strip or wall outlet, but rather that the computers were less able to isolate the 60 Hz signal from the sound card output compared with the other computers.

Options for Noisy Sound Cards

Based on the results of this study, it is likely most computers will have no difficulty producing a pure enough signal for clinical use. However, in the event that a computer's sound card is not capable of producing an adequate signal, the use of an alternative sound card or sound controller could be considered. Many aftermarket devices are available, which vary greatly in both quality and price. Desktop computers can use a secondary sound card installed in an available PCI slot. Several of these sound cards are available at retail stores for less than $50 (e.g. Creative Sound Blaster Audigy SE, Trust 5.1 Sound Expert Optical 514DX, Turtle Beach Riviera). Laptop computers can use a PCMCIA sound card. The PCMCIA slot technology found in most laptops is the laptop equivalent for a PCI slot. Recently, USB sound controllers and external sound cards have become more popular. USB sound controllers typically offer the audio production functions of a sound card, packaged into the size of a flash drive. They can be

easily installed into any computer with a USB port and can provide adequate sound quality for $20-$30.

Setting the Computer's Output Level

Although most computers are capable of producing reasonably quiet and undistorted sound, the volume level of the computer must be adjusted appropriately. There are several ways to adjust the output of the sound card. A single computer may contain several software-driven volume controls, a control on the keyboard and an external amplifier control. In addition, there may be volume controls built into amplified speakers, monitors, and CD-ROM or DVD-ROM drives. However, this last list of controls typically does not affect the signal traveling from the sound card to the audiometer. For example, adjusting the volume control of an amplified speaker will only affect the output of the speaker and not the signal traveling to the audiometer. In the case of a personal computer running Windows, most volume levels can be controlled by the Windows Mixer or Volume Control software found in Windows' Control Panel. Windows Mixer contains a variety of volume controls. The signal produced when playing a WAV file is typically manipulated by two separate controls. However, the names of these controls vary, depending on the sound card manufacturer. Typically, one controls the main volume level, while the other controls the attenuation or amplification of WAV files. Often, they are labeled "Wave" and "Main Volume" respectively. If both controls are set to maximum, clipping is likely to occur, producing unwanted noise and distortion.

In this study, most of the computers tested had the best sound quality when the WAV control was set to maximum and the main volume was set no higher than 75-85%.

However, the Vsquared computer and the Compaq Armada V300 produced the best quality with the opposite configuration; the main volume level was set to maximum and the WAV level was decreased. This reflects a difference in the amplification circuit design employed by the sound card manufacturer. It appears that most of the sound cards' output circuits utilized an attenuator, controlled by the Mixer WAV control, situated before the master amplifier. In contrast, the design of the Vsquared computer and Compaq Armada V300 likely utilized the Mixer WAV control to amplify the signal before being attenuated by the main volume control.

When setting up a computer for clinical use, it is important to determine which of the two Windows Mixer controls attenuates and which amplifies. The following procedure is suggested: With the WAV control set to maximum and the main volume control set to 50%, observe the spectral output on a digital frequency analyzer. Note the peak amplitude and the presence of low-frequency noise and sidebands. Then set the main volume control to maximum and adjust the WAV control until the same peak amplitude is produced. If the spectral output improves in the second condition, producing fewer sidebands and less low-frequency noise, then the master volume control is likely an attenuator and should be set to maximum. Otherwise, the WAV control should be set to maximum.

If this latter condition is assumed, and the WAV control is at 100%, the master volume control must then be adjusted to a level appropriate for the audiometer. Since most audiometers can accommodate a broad range of input intensity levels, it is suggested that the Windows Mixer master volume level be set to approximately 50%. The audiometer's input sensitivity can then be adjusted to accommodate for differences

between sound card output levels. If a calibration tone cannot produce a 0 dB reading on the audiometer's VU meter, in spite of adjusting the input sensitivity of the audiometer, the Mixer master volume should then be adjusted to more or less than 50%.

Once the computer and audiometer are adjusted to the appropriate output intensity level, a digital frequency analyzer should be used to verify the noise and distortion levels are permissible under the ANSI standard for audiometry (ANSI s3.36-1996).

Maintaining and Verifying Output Levels

One potential problem of using software based speech stimuli stems from the ease at which a personal computer's volume can be adjusted. Most manufacturer versions of Windows Mixer do not display numerical volume levels. The volume is adjusted using a slider with very limited visual markings. This makes it extremely difficult to verify that the volume setting has not changed since it was originally set and calibrated. In addition, some software applications can actually adjust a computer's volume levels, to accommodate the use of that particular piece of software, without even notifying the user. For example, during this study, it was found that Microsoft Office XP can change the microphone input levels for Windows. The voice recognition calibration procedure automatically adjusted the microphone input level to maximize voice recognition ability. Although this change affected the entire machine, it was only visible in a sub-window of Windows Mixer.

Accidental changes of output levels can be avoided in a number of ways. The following suggestions should be used in combination when applicable. The software of some sound cards provides numerical volume levels. The numerical volume level could be easily noted at calibration and then verified daily. Alternatively, if numerical volume

levels are not provided, the input sensitivity of the audiometer could be noted or marked at calibration. During clinical use, if the calibration tone does not produce a 0 dB VU meter reading, and the input sensitivity has not been changed, it suggests that the computer's volume levels have been adjusted. Of course, daily biological calibration checks could also prove useful in thwarting unintentional volume adjustments.

Future Research

The results of this study suggest that commonly available computer sound cards are capable of producing an acceptable sound quality for speech audiometry based on ANSI standards. However, in a clinical setting, the adequacy of each individual computer must still be evaluated.

The traditional recorded speech method differs from the computer-based presentation method in multiple ways, including the amount of time required, the sound processing, and the variable inter-stimulus interval produced by a client-controlled presentation pace. Stach and colleagues (1995) conducted research highlighting the time-saving benefits of the computer-based method. The present study results suggest that the difference in sound processing does not disqualify most computers for clinical use; however, the effects of a variable presentation pace and inter-stimulus interval have not been thoroughly researched. Anecdotal evidence suggests that this change in pace does not significantly affect word recognition scores (Stach, et al., 1995). However, formal research is required prior to the use of the current normative data to the new presentation format. Additional research and development in the area of computer-based word recognition presentation methods has the potential to increase clinical efficiency, and

provide audiologists with flexible, standardized, and evidence-based speech recognition test protocols.

APPENDICES

Appendix A

Visual Basic Code for Protype Software Program

```
Public Class frmAudSpeech
   Inherits System.Windows.Forms.Form

#Region " Windows Form Designer generated code "

   Public Sub New()
      MyBase.New()

      'This call is required by the Windows Form Designer.
      InitializeComponent()

      'Add any initialization after the InitializeComponent() call

   End Sub

   'Form overrides dispose to clean up the component list.
   Protected Overloads Overrides Sub Dispose(ByVal disposing As Boolean)
      If disposing Then
         If Not (components Is Nothing) Then
            components.Dispose()
         End If
      End If
      MyBase.Dispose(disposing)
   End Sub

   'Required by the Windows Form Designer
   Private components As System.ComponentModel.IContainer

   'NOTE: The following procedure is required by the Windows Form Designer
   'It can be modified using the Windows Form Designer.
   'Do not modify it using the code editor.
   Friend WithEvents btnFinishCorrect As System.Windows.Forms.Button
   Friend WithEvents btnIncorrect As System.Windows.Forms.Button
   Friend WithEvents btnRepeat As System.Windows.Forms.Button
   Friend WithEvents btnSkip As System.Windows.Forms.Button
   Friend WithEvents btnFinishIncorrect As System.Windows.Forms.Button
   Friend WithEvents lblTotalCorrect As System.Windows.Forms.Label
```

```
Friend WithEvents lblTotalOutOf As System.Windows.Forms.Label
Friend WithEvents lblSlash As System.Windows.Forms.Label
Friend WithEvents lblEquals As System.Windows.Forms.Label
Friend WithEvents lblPercentCorrect As System.Windows.Forms.Label
Friend WithEvents lblPercentSign As System.Windows.Forms.Label
Friend WithEvents btnStartRestart As System.Windows.Forms.Button
Friend WithEvents btnCalibrate As System.Windows.Forms.Button
Friend WithEvents lblWord8 As System.Windows.Forms.Label
Friend WithEvents lblWord9 As System.Windows.Forms.Label
Friend WithEvents lblWord10 As System.Windows.Forms.Label
Friend WithEvents lblWord17 As System.Windows.Forms.Label
Friend WithEvents lblWord18 As System.Windows.Forms.Label
Friend WithEvents lblWord19 As System.Windows.Forms.Label
Friend WithEvents lblWord20 As System.Windows.Forms.Label
Friend WithEvents lblWord16 As System.Windows.Forms.Label
Friend WithEvents lblWord12 As System.Windows.Forms.Label
Friend WithEvents lblWord13 As System.Windows.Forms.Label
Friend WithEvents lblWord14 As System.Windows.Forms.Label
Friend WithEvents lblWord15 As System.Windows.Forms.Label
Friend WithEvents lblWord11 As System.Windows.Forms.Label
Friend WithEvents lblWord37 As System.Windows.Forms.Label
Friend WithEvents lblWord38 As System.Windows.Forms.Label
Friend WithEvents lblWord39 As System.Windows.Forms.Label
Friend WithEvents lblWord40 As System.Windows.Forms.Label
Friend WithEvents lblWord36 As System.Windows.Forms.Label
Friend WithEvents lblWord32 As System.Windows.Forms.Label
Friend WithEvents lblWord33 As System.Windows.Forms.Label
Friend WithEvents lblWord34 As System.Windows.Forms.Label
Friend WithEvents lblWord35 As System.Windows.Forms.Label
Friend WithEvents lblWord31 As System.Windows.Forms.Label
Friend WithEvents lblWord27 As System.Windows.Forms.Label
Friend WithEvents lblWord28 As System.Windows.Forms.Label
Friend WithEvents lblWord29 As System.Windows.Forms.Label
Friend WithEvents lblWord30 As System.Windows.Forms.Label
Friend WithEvents lblWord26 As System.Windows.Forms.Label
Friend WithEvents lblWord22 As System.Windows.Forms.Label
Friend WithEvents lblWord23 As System.Windows.Forms.Label
Friend WithEvents lblWord24 As System.Windows.Forms.Label
Friend WithEvents lblWord25 As System.Windows.Forms.Label
Friend WithEvents lblWord21 As System.Windows.Forms.Label
Friend WithEvents lblWord47 As System.Windows.Forms.Label
Friend WithEvents lblWord48 As System.Windows.Forms.Label
Friend WithEvents lblWord49 As System.Windows.Forms.Label
Friend WithEvents lblWord46 As System.Windows.Forms.Label
Friend WithEvents lblWord42 As System.Windows.Forms.Label
Friend WithEvents lblWord43 As System.Windows.Forms.Label
```

```
    Friend WithEvents lblWord44 As System.Windows.Forms.Label
    Friend WithEvents lblWord45 As System.Windows.Forms.Label
    Friend WithEvents lblWord41 As System.Windows.Forms.Label
    Friend WithEvents lblWord50 As System.Windows.Forms.Label
    Friend WithEvents lblTitle As System.Windows.Forms.Label
    Friend WithEvents lblWord7 As System.Windows.Forms.Label
    Friend WithEvents lblWord6 As System.Windows.Forms.Label
    Friend WithEvents lblWord5 As System.Windows.Forms.Label
    Friend WithEvents lblWord4 As System.Windows.Forms.Label
    Friend WithEvents lblWord3 As System.Windows.Forms.Label
    Friend WithEvents lblWord2 As System.Windows.Forms.Label
    Friend WithEvents lblWord1 As System.Windows.Forms.Label
    Friend WithEvents btnCorrect As System.Windows.Forms.Button
    <System.Diagnostics.DebuggerStepThrough()> Private Sub InitializeComponent()
        Dim resources As System.Resources.ResourceManager = New
System.Resources.ResourceManager(GetType(frmAudSpeech))
        Me.lblWord8 = New System.Windows.Forms.Label
        Me.lblWord9 = New System.Windows.Forms.Label
        Me.lblWord10 = New System.Windows.Forms.Label
        Me.lblWord17 = New System.Windows.Forms.Label
        Me.lblWord18 = New System.Windows.Forms.Label
        Me.lblWord19 = New System.Windows.Forms.Label
        Me.lblWord20 = New System.Windows.Forms.Label
        Me.lblWord16 = New System.Windows.Forms.Label
        Me.lblWord12 = New System.Windows.Forms.Label
        Me.lblWord13 = New System.Windows.Forms.Label
        Me.lblWord14 = New System.Windows.Forms.Label
        Me.lblWord15 = New System.Windows.Forms.Label
        Me.lblWord11 = New System.Windows.Forms.Label
        Me.lblWord37 = New System.Windows.Forms.Label
        Me.lblWord38 = New System.Windows.Forms.Label
        Me.lblWord39 = New System.Windows.Forms.Label
        Me.lblWord40 = New System.Windows.Forms.Label
        Me.lblWord36 = New System.Windows.Forms.Label
        Me.lblWord32 = New System.Windows.Forms.Label
        Me.lblWord33 = New System.Windows.Forms.Label
        Me.lblWord34 = New System.Windows.Forms.Label
        Me.lblWord35 = New System.Windows.Forms.Label
        Me.lblWord31 = New System.Windows.Forms.Label
        Me.lblWord27 = New System.Windows.Forms.Label
        Me.lblWord28 = New System.Windows.Forms.Label
        Me.lblWord29 = New System.Windows.Forms.Label
        Me.lblWord30 = New System.Windows.Forms.Label
        Me.lblWord26 = New System.Windows.Forms.Label
        Me.lblWord22 = New System.Windows.Forms.Label
        Me.lblWord23 = New System.Windows.Forms.Label
```

```
Me.lblWord24 = New System.Windows.Forms.Label
Me.lblWord25 = New System.Windows.Forms.Label
Me.lblWord21 = New System.Windows.Forms.Label
Me.lblWord47 = New System.Windows.Forms.Label
Me.lblWord48 = New System.Windows.Forms.Label
Me.lblWord49 = New System.Windows.Forms.Label
Me.lblWord50 = New System.Windows.Forms.Label
Me.lblWord46 = New System.Windows.Forms.Label
Me.lblWord42 = New System.Windows.Forms.Label
Me.lblWord43 = New System.Windows.Forms.Label
Me.lblWord44 = New System.Windows.Forms.Label
Me.lblWord45 = New System.Windows.Forms.Label
Me.lblWord41 = New System.Windows.Forms.Label
Me.btnCorrect = New System.Windows.Forms.Button
Me.btnFinishCorrect = New System.Windows.Forms.Button
Me.btnIncorrect = New System.Windows.Forms.Button
Me.btnRepeat = New System.Windows.Forms.Button
Me.btnSkip = New System.Windows.Forms.Button
Me.btnFinishIncorrect = New System.Windows.Forms.Button
Me.lblTotalCorrect = New System.Windows.Forms.Label
Me.lblTotalOutOf = New System.Windows.Forms.Label
Me.lblSlash = New System.Windows.Forms.Label
Me.lblEquals = New System.Windows.Forms.Label
Me.lblPercentCorrect = New System.Windows.Forms.Label
Me.lblPercentSign = New System.Windows.Forms.Label
Me.btnStartRestart = New System.Windows.Forms.Button
Me.btnCalibrate = New System.Windows.Forms.Button
Me.lblTitle = New System.Windows.Forms.Label
Me.lblWord7 = New System.Windows.Forms.Label
Me.lblWord6 = New System.Windows.Forms.Label
Me.lblWord5 = New System.Windows.Forms.Label
Me.lblWord4 = New System.Windows.Forms.Label
Me.lblWord3 = New System.Windows.Forms.Label
Me.lblWord2 = New System.Windows.Forms.Label
Me.lblWord1 = New System.Windows.Forms.Label
Me.SuspendLayout()
'
'lblWord8
'
Me.lblWord8.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord8.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord8.Location = New System.Drawing.Point(8, 240)
Me.lblWord8.Name = "lblWord8"
Me.lblWord8.Size = New System.Drawing.Size(112, 16)
Me.lblWord8.TabIndex = 15
'
```

```
'lblWord9
'
Me.lblWord9.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord9.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord9.Location = New System.Drawing.Point(8, 264)
Me.lblWord9.Name = "lblWord9"
Me.lblWord9.Size = New System.Drawing.Size(112, 16)
Me.lblWord9.TabIndex = 14
'
'lblWord10
'
Me.lblWord10.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord10.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord10.Location = New System.Drawing.Point(8, 288)
Me.lblWord10.Name = "lblWord10"
Me.lblWord10.Size = New System.Drawing.Size(112, 16)
Me.lblWord10.TabIndex = 13
'
'lblWord17
'
Me.lblWord17.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord17.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord17.Location = New System.Drawing.Point(136, 216)
Me.lblWord17.Name = "lblWord17"
Me.lblWord17.Size = New System.Drawing.Size(112, 16)
Me.lblWord17.TabIndex = 26
'
'lblWord18
'
Me.lblWord18.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord18.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord18.Location = New System.Drawing.Point(136, 240)
Me.lblWord18.Name = "lblWord18"
Me.lblWord18.Size = New System.Drawing.Size(112, 16)
Me.lblWord18.TabIndex = 25
'
'lblWord19
'
Me.lblWord19.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord19.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord19.Location = New System.Drawing.Point(136, 264)
Me.lblWord19.Name = "lblWord19"
Me.lblWord19.Size = New System.Drawing.Size(112, 16)
Me.lblWord19.TabIndex = 24
'
'lblWord20
```

```
'
Me.lblWord20.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord20.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord20.Location = New System.Drawing.Point(136, 288)
Me.lblWord20.Name = "lblWord20"
Me.lblWord20.Size = New System.Drawing.Size(112, 16)
Me.lblWord20.TabIndex = 23
'
'lblWord16
'
Me.lblWord16.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord16.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord16.Location = New System.Drawing.Point(136, 192)
Me.lblWord16.Name = "lblWord16"
Me.lblWord16.Size = New System.Drawing.Size(112, 16)
Me.lblWord16.TabIndex = 22
'
'lblWord12
'
Me.lblWord12.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord12.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord12.Location = New System.Drawing.Point(136, 96)
Me.lblWord12.Name = "lblWord12"
Me.lblWord12.Size = New System.Drawing.Size(112, 16)
Me.lblWord12.TabIndex = 21
'
'lblWord13
'
Me.lblWord13.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord13.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord13.Location = New System.Drawing.Point(136, 120)
Me.lblWord13.Name = "lblWord13"
Me.lblWord13.Size = New System.Drawing.Size(112, 16)
Me.lblWord13.TabIndex = 20
'
'lblWord14
'
Me.lblWord14.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord14.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord14.Location = New System.Drawing.Point(136, 144)
Me.lblWord14.Name = "lblWord14"
Me.lblWord14.Size = New System.Drawing.Size(112, 16)
Me.lblWord14.TabIndex = 19
'
'lblWord15
'
```

```
Me.lblWord15.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord15.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord15.Location = New System.Drawing.Point(136, 168)
Me.lblWord15.Name = "lblWord15"
Me.lblWord15.Size = New System.Drawing.Size(112, 16)
Me.lblWord15.TabIndex = 18
'
'lblWord11
'
Me.lblWord11.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord11.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord11.Location = New System.Drawing.Point(136, 72)
Me.lblWord11.Name = "lblWord11"
Me.lblWord11.Size = New System.Drawing.Size(112, 16)
Me.lblWord11.TabIndex = 17
'
'lblWord37
'
Me.lblWord37.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord37.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord37.Location = New System.Drawing.Point(392, 216)
Me.lblWord37.Name = "lblWord37"
Me.lblWord37.Size = New System.Drawing.Size(112, 16)
Me.lblWord37.TabIndex = 46
'
'lblWord38
'
Me.lblWord38.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord38.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord38.Location = New System.Drawing.Point(392, 240)
Me.lblWord38.Name = "lblWord38"
Me.lblWord38.Size = New System.Drawing.Size(112, 16)
Me.lblWord38.TabIndex = 45
'
'lblWord39
'
Me.lblWord39.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord39.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord39.Location = New System.Drawing.Point(392, 264)
Me.lblWord39.Name = "lblWord39"
Me.lblWord39.Size = New System.Drawing.Size(112, 16)
Me.lblWord39.TabIndex = 44
'
'lblWord40
'
Me.lblWord40.BackColor = System.Drawing.SystemColors.Menu
```

```
Me.lblWord40.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord40.Location = New System.Drawing.Point(392, 288)
Me.lblWord40.Name = "lblWord40"
Me.lblWord40.Size = New System.Drawing.Size(112, 16)
Me.lblWord40.TabIndex = 43
'
'lblWord36
'
Me.lblWord36.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord36.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord36.Location = New System.Drawing.Point(392, 192)
Me.lblWord36.Name = "lblWord36"
Me.lblWord36.Size = New System.Drawing.Size(112, 16)
Me.lblWord36.TabIndex = 42
'
'lblWord32
'
Me.lblWord32.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord32.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord32.Location = New System.Drawing.Point(392, 96)
Me.lblWord32.Name = "lblWord32"
Me.lblWord32.Size = New System.Drawing.Size(112, 16)
Me.lblWord32.TabIndex = 41
'
'lblWord33
'
Me.lblWord33.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord33.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord33.Location = New System.Drawing.Point(392, 120)
Me.lblWord33.Name = "lblWord33"
Me.lblWord33.Size = New System.Drawing.Size(112, 16)
Me.lblWord33.TabIndex = 40
'
'lblWord34
'
Me.lblWord34.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord34.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord34.Location = New System.Drawing.Point(392, 144)
Me.lblWord34.Name = "lblWord34"
Me.lblWord34.Size = New System.Drawing.Size(112, 16)
Me.lblWord34.TabIndex = 39
'
'lblWord35
'
Me.lblWord35.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord35.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
```

```
Me.lblWord35.Location = New System.Drawing.Point(392, 168)
Me.lblWord35.Name = "lblWord35"
Me.lblWord35.Size = New System.Drawing.Size(112, 16)
Me.lblWord35.TabIndex = 38
'
'lblWord31
'
Me.lblWord31.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord31.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord31.Location = New System.Drawing.Point(392, 72)
Me.lblWord31.Name = "lblWord31"
Me.lblWord31.Size = New System.Drawing.Size(112, 16)
Me.lblWord31.TabIndex = 37
'
'lblWord27
'
Me.lblWord27.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord27.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord27.Location = New System.Drawing.Point(264, 216)
Me.lblWord27.Name = "lblWord27"
Me.lblWord27.Size = New System.Drawing.Size(112, 16)
Me.lblWord27.TabIndex = 36
'
'lblWord28
'
Me.lblWord28.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord28.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord28.Location = New System.Drawing.Point(264, 240)
Me.lblWord28.Name = "lblWord28"
Me.lblWord28.Size = New System.Drawing.Size(112, 16)
Me.lblWord28.TabIndex = 35
'
'lblWord29
'
Me.lblWord29.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord29.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord29.Location = New System.Drawing.Point(264, 264)
Me.lblWord29.Name = "lblWord29"
Me.lblWord29.Size = New System.Drawing.Size(112, 16)
Me.lblWord29.TabIndex = 34
'
'lblWord30
'
Me.lblWord30.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord30.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord30.Location = New System.Drawing.Point(264, 288)
```

```
Me.lblWord30.Name = "lblWord30"
Me.lblWord30.Size = New System.Drawing.Size(112, 16)
Me.lblWord30.TabIndex = 33
'
'lblWord26
'
Me.lblWord26.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord26.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord26.Location = New System.Drawing.Point(264, 192)
Me.lblWord26.Name = "lblWord26"
Me.lblWord26.Size = New System.Drawing.Size(112, 16)
Me.lblWord26.TabIndex = 32
'
'lblWord22
'
Me.lblWord22.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord22.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord22.Location = New System.Drawing.Point(264, 96)
Me.lblWord22.Name = "lblWord22"
Me.lblWord22.Size = New System.Drawing.Size(112, 16)
Me.lblWord22.TabIndex = 31
'
'lblWord23
'
Me.lblWord23.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord23.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord23.Location = New System.Drawing.Point(264, 120)
Me.lblWord23.Name = "lblWord23"
Me.lblWord23.Size = New System.Drawing.Size(112, 16)
Me.lblWord23.TabIndex = 30
'
'lblWord24
'
Me.lblWord24.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord24.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord24.Location = New System.Drawing.Point(264, 144)
Me.lblWord24.Name = "lblWord24"
Me.lblWord24.Size = New System.Drawing.Size(112, 16)
Me.lblWord24.TabIndex = 29
'
'lblWord25
'
Me.lblWord25.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord25.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord25.Location = New System.Drawing.Point(264, 168)
Me.lblWord25.Name = "lblWord25"
```

```
Me.lblWord25.Size = New System.Drawing.Size(112, 16)
Me.lblWord25.TabIndex = 28
'
'lblWord21
'
Me.lblWord21.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord21.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord21.Location = New System.Drawing.Point(264, 72)
Me.lblWord21.Name = "lblWord21"
Me.lblWord21.Size = New System.Drawing.Size(112, 16)
Me.lblWord21.TabIndex = 27
'
'lblWord47
'
Me.lblWord47.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord47.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord47.Location = New System.Drawing.Point(520, 216)
Me.lblWord47.Name = "lblWord47"
Me.lblWord47.Size = New System.Drawing.Size(112, 16)
Me.lblWord47.TabIndex = 56
'
'lblWord48
'
Me.lblWord48.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord48.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord48.Location = New System.Drawing.Point(520, 240)
Me.lblWord48.Name = "lblWord48"
Me.lblWord48.Size = New System.Drawing.Size(112, 16)
Me.lblWord48.TabIndex = 55
'
'lblWord49
'
Me.lblWord49.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord49.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord49.Location = New System.Drawing.Point(520, 264)
Me.lblWord49.Name = "lblWord49"
Me.lblWord49.Size = New System.Drawing.Size(112, 16)
Me.lblWord49.TabIndex = 54
'
'lblWord50
'
Me.lblWord50.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord50.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord50.Location = New System.Drawing.Point(520, 288)
Me.lblWord50.Name = "lblWord50"
Me.lblWord50.Size = New System.Drawing.Size(112, 16)
```

```
Me.lblWord50.TabIndex = 53
'
'lblWord46
'
Me.lblWord46.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord46.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord46.Location = New System.Drawing.Point(520, 192)
Me.lblWord46.Name = "lblWord46"
Me.lblWord46.Size = New System.Drawing.Size(112, 16)
Me.lblWord46.TabIndex = 52
'
'lblWord42
'
Me.lblWord42.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord42.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord42.Location = New System.Drawing.Point(520, 96)
Me.lblWord42.Name = "lblWord42"
Me.lblWord42.Size = New System.Drawing.Size(112, 16)
Me.lblWord42.TabIndex = 51
'
'lblWord43
'
Me.lblWord43.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord43.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord43.Location = New System.Drawing.Point(520, 120)
Me.lblWord43.Name = "lblWord43"
Me.lblWord43.Size = New System.Drawing.Size(112, 16)
Me.lblWord43.TabIndex = 50
'
'lblWord44
'
Me.lblWord44.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord44.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord44.Location = New System.Drawing.Point(520, 144)
Me.lblWord44.Name = "lblWord44"
Me.lblWord44.Size = New System.Drawing.Size(112, 16)
Me.lblWord44.TabIndex = 49
'
'lblWord45
'
Me.lblWord45.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord45.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord45.Location = New System.Drawing.Point(520, 168)
Me.lblWord45.Name = "lblWord45"
Me.lblWord45.Size = New System.Drawing.Size(112, 16)
Me.lblWord45.TabIndex = 48
```

```
'
'lblWord41
'
Me.lblWord41.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord41.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord41.Location = New System.Drawing.Point(520, 72)
Me.lblWord41.Name = "lblWord41"
Me.lblWord41.Size = New System.Drawing.Size(112, 16)
Me.lblWord41.TabIndex = 47
'
'btnCorrect
'
Me.btnCorrect.Enabled = False
Me.btnCorrect.FlatStyle = System.Windows.Forms.FlatStyle.System
Me.btnCorrect.Location = New System.Drawing.Point(64, 384)
Me.btnCorrect.Name = "btnCorrect"
Me.btnCorrect.Size = New System.Drawing.Size(120, 40)
Me.btnCorrect.TabIndex = 57
Me.btnCorrect.TabStop = False
Me.btnCorrect.Text = "Correct"
'
'btnFinishCorrect
'
Me.btnFinishCorrect.Enabled = False
Me.btnFinishCorrect.FlatStyle = System.Windows.Forms.FlatStyle.System
Me.btnFinishCorrect.Location = New System.Drawing.Point(64, 432)
Me.btnFinishCorrect.Name = "btnFinishCorrect"
Me.btnFinishCorrect.Size = New System.Drawing.Size(120, 23)
Me.btnFinishCorrect.TabIndex = 59
Me.btnFinishCorrect.Text = "Finish (Correct)"
'
'btnIncorrect
'
Me.btnIncorrect.Enabled = False
Me.btnIncorrect.FlatStyle = System.Windows.Forms.FlatStyle.System
Me.btnIncorrect.Location = New System.Drawing.Point(200, 384)
Me.btnIncorrect.Name = "btnIncorrect"
Me.btnIncorrect.Size = New System.Drawing.Size(120, 40)
Me.btnIncorrect.TabIndex = 62
Me.btnIncorrect.Text = "Incorrect"
'
'btnRepeat
'
Me.btnRepeat.Enabled = False
Me.btnRepeat.FlatStyle = System.Windows.Forms.FlatStyle.System
Me.btnRepeat.Location = New System.Drawing.Point(336, 384)
```

```
        Me.btnRepeat.Name = "btnRepeat"
        Me.btnRepeat.Size = New System.Drawing.Size(120, 40)
        Me.btnRepeat.TabIndex = 63
        Me.btnRepeat.Text = "Repeat"
        '
        'btnSkip
        '
        Me.btnSkip.Enabled = False
        Me.btnSkip.FlatStyle = System.Windows.Forms.FlatStyle.System
        Me.btnSkip.Location = New System.Drawing.Point(472, 384)
        Me.btnSkip.Name = "btnSkip"
        Me.btnSkip.Size = New System.Drawing.Size(120, 40)
        Me.btnSkip.TabIndex = 64
        Me.btnSkip.Text = "Skip"
        '
        'btnFinishIncorrect
        '
        Me.btnFinishIncorrect.Enabled = False
        Me.btnFinishIncorrect.FlatStyle = System.Windows.Forms.FlatStyle.System
        Me.btnFinishIncorrect.Location = New System.Drawing.Point(200, 432)
        Me.btnFinishIncorrect.Name = "btnFinishIncorrect"
        Me.btnFinishIncorrect.Size = New System.Drawing.Size(120, 23)
        Me.btnFinishIncorrect.TabIndex = 65
        Me.btnFinishIncorrect.Text = "Finish (Incorrect)"
        '
        'lblTotalCorrect
        '
        Me.lblTotalCorrect.BackColor = System.Drawing.Color.Transparent
        Me.lblTotalCorrect.Font = New System.Drawing.Font("Microsoft Sans Serif",
21.75!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.lblTotalCorrect.ForeColor = System.Drawing.Color.WhiteSmoke
        Me.lblTotalCorrect.Location = New System.Drawing.Point(208, 328)
        Me.lblTotalCorrect.Name = "lblTotalCorrect"
        Me.lblTotalCorrect.Size = New System.Drawing.Size(48, 32)
        Me.lblTotalCorrect.TabIndex = 66
        Me.lblTotalCorrect.Text = "0"
        '
        'lblTotalOutOf
        '
        Me.lblTotalOutOf.BackColor = System.Drawing.Color.Transparent
        Me.lblTotalOutOf.Font = New System.Drawing.Font("Microsoft Sans Serif",
21.75!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.lblTotalOutOf.ForeColor = System.Drawing.Color.WhiteSmoke
        Me.lblTotalOutOf.Location = New System.Drawing.Point(280, 328)
```

```
        Me.lblTotalOutOf.Name = "lblTotalOutOf"
        Me.lblTotalOutOf.Size = New System.Drawing.Size(48, 32)
        Me.lblTotalOutOf.TabIndex = 67
        Me.lblTotalOutOf.Text = "0"
        '
        'lblSlash
        '
        Me.lblSlash.BackColor = System.Drawing.Color.Transparent
        Me.lblSlash.Font = New System.Drawing.Font("Microsoft Sans Serif", 21.75!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.lblSlash.ForeColor = System.Drawing.Color.WhiteSmoke
        Me.lblSlash.Location = New System.Drawing.Point(256, 328)
        Me.lblSlash.Name = "lblSlash"
        Me.lblSlash.Size = New System.Drawing.Size(24, 32)
        Me.lblSlash.TabIndex = 68
        Me.lblSlash.Text = "/"
        '
        'lblEquals
        '
        Me.lblEquals.BackColor = System.Drawing.Color.Transparent
        Me.lblEquals.Font = New System.Drawing.Font("Microsoft Sans Serif", 21.75!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.lblEquals.ForeColor = System.Drawing.Color.WhiteSmoke
        Me.lblEquals.Location = New System.Drawing.Point(328, 328)
        Me.lblEquals.Name = "lblEquals"
        Me.lblEquals.Size = New System.Drawing.Size(32, 32)
        Me.lblEquals.TabIndex = 69
        Me.lblEquals.Text = "="
        '
        'lblPercentCorrect
        '
        Me.lblPercentCorrect.BackColor = System.Drawing.Color.Transparent
        Me.lblPercentCorrect.Font = New System.Drawing.Font("Microsoft Sans Serif",
21.75!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.lblPercentCorrect.ForeColor = System.Drawing.Color.WhiteSmoke
        Me.lblPercentCorrect.Location = New System.Drawing.Point(360, 328)
        Me.lblPercentCorrect.Name = "lblPercentCorrect"
        Me.lblPercentCorrect.Size = New System.Drawing.Size(64, 32)
        Me.lblPercentCorrect.TabIndex = 70
        Me.lblPercentCorrect.Text = "0"
        '
        'lblPercentSign
        '
```

```
        Me.lblPercentSign.BackColor = System.Drawing.Color.Transparent
        Me.lblPercentSign.Font = New System.Drawing.Font("Microsoft Sans Serif",
21.75!, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point,
CType(0, Byte))
        Me.lblPercentSign.ForeColor = System.Drawing.Color.WhiteSmoke
        Me.lblPercentSign.Location = New System.Drawing.Point(424, 328)
        Me.lblPercentSign.Name = "lblPercentSign"
        Me.lblPercentSign.Size = New System.Drawing.Size(40, 32)
        Me.lblPercentSign.TabIndex = 71
        Me.lblPercentSign.Text = "%"
        '
        'btnStartRestart
        '
        Me.btnStartRestart.FlatStyle = System.Windows.Forms.FlatStyle.System
        Me.btnStartRestart.Location = New System.Drawing.Point(472, 432)
        Me.btnStartRestart.Name = "btnStartRestart"
        Me.btnStartRestart.Size = New System.Drawing.Size(120, 23)
        Me.btnStartRestart.TabIndex = 72
        Me.btnStartRestart.Text = "Start/Restart"
        '
        'btnCalibrate
        '
        Me.btnCalibrate.FlatStyle = System.Windows.Forms.FlatStyle.System
        Me.btnCalibrate.Location = New System.Drawing.Point(336, 432)
        Me.btnCalibrate.Name = "btnCalibrate"
        Me.btnCalibrate.Size = New System.Drawing.Size(120, 23)
        Me.btnCalibrate.TabIndex = 73
        Me.btnCalibrate.Text = "Calibrate"
        '
        'lblTitle
        '
        Me.lblTitle.BackColor = System.Drawing.Color.Transparent
        Me.lblTitle.Font = New System.Drawing.Font("Microsoft Sans Serif", 24.0!,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, CType(0, Byte))
        Me.lblTitle.ForeColor = System.Drawing.Color.WhiteSmoke
        Me.lblTitle.Location = New System.Drawing.Point(168, 8)
        Me.lblTitle.Name = "lblTitle"
        Me.lblTitle.Size = New System.Drawing.Size(304, 40)
        Me.lblTitle.TabIndex = 74
        Me.lblTitle.Text = "NU-6 List 1 Form A"
        '
        'lblWord7
        '
        Me.lblWord7.BackColor = System.Drawing.SystemColors.Menu
        Me.lblWord7.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
        Me.lblWord7.Location = New System.Drawing.Point(8, 216)
```

```
Me.lblWord7.Name = "lblWord7"
Me.lblWord7.Size = New System.Drawing.Size(112, 16)
Me.lblWord7.TabIndex = 16
'
'lblWord6
'
Me.lblWord6.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord6.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord6.Location = New System.Drawing.Point(8, 192)
Me.lblWord6.Name = "lblWord6"
Me.lblWord6.Size = New System.Drawing.Size(112, 16)
Me.lblWord6.TabIndex = 12
'
'lblWord5
'
Me.lblWord5.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord5.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord5.Location = New System.Drawing.Point(8, 168)
Me.lblWord5.Name = "lblWord5"
Me.lblWord5.Size = New System.Drawing.Size(112, 16)
Me.lblWord5.TabIndex = 8
'
'lblWord4
'
Me.lblWord4.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord4.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord4.Location = New System.Drawing.Point(8, 144)
Me.lblWord4.Name = "lblWord4"
Me.lblWord4.Size = New System.Drawing.Size(112, 16)
Me.lblWord4.TabIndex = 9
'
'lblWord3
'
Me.lblWord3.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord3.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord3.Location = New System.Drawing.Point(8, 120)
Me.lblWord3.Name = "lblWord3"
Me.lblWord3.Size = New System.Drawing.Size(112, 16)
Me.lblWord3.TabIndex = 10
'
'lblWord2
'
Me.lblWord2.BackColor = System.Drawing.SystemColors.Menu
Me.lblWord2.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
Me.lblWord2.Location = New System.Drawing.Point(8, 96)
Me.lblWord2.Name = "lblWord2"
```

```
        Me.lblWord2.Size = New System.Drawing.Size(112, 16)
        Me.lblWord2.TabIndex = 11
        '
        'lblWord1
        '
        Me.lblWord1.BackColor = System.Drawing.SystemColors.Menu
        Me.lblWord1.Font = New System.Drawing.Font("Microsoft Sans Serif", 8.25!,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.lblWord1.ImageAlign = System.Drawing.ContentAlignment.MiddleRight
        Me.lblWord1.Location = New System.Drawing.Point(8, 72)
        Me.lblWord1.Name = "lblWord1"
        Me.lblWord1.Size = New System.Drawing.Size(112, 16)
        Me.lblWord1.TabIndex = 0
        '
        'frmAudSpeech
        '
        Me.AutoScaleBaseSize = New System.Drawing.Size(5, 13)
        Me.BackColor = System.Drawing.SystemColors.Control
        Me.BackgroundImage = CType(resources.GetObject("$this.BackgroundImage"),
System.Drawing.Image)
        Me.ClientSize = New System.Drawing.Size(640, 518)
        Me.Controls.Add(Me.lblTitle)
        Me.Controls.Add(Me.btnCalibrate)
        Me.Controls.Add(Me.btnStartRestart)
        Me.Controls.Add(Me.lblPercentSign)
        Me.Controls.Add(Me.lblPercentCorrect)
        Me.Controls.Add(Me.lblEquals)
        Me.Controls.Add(Me.lblSlash)
        Me.Controls.Add(Me.lblTotalOutOf)
        Me.Controls.Add(Me.lblTotalCorrect)
        Me.Controls.Add(Me.btnFinishIncorrect)
        Me.Controls.Add(Me.btnSkip)
        Me.Controls.Add(Me.btnRepeat)
        Me.Controls.Add(Me.btnIncorrect)
        Me.Controls.Add(Me.btnFinishCorrect)
        Me.Controls.Add(Me.btnCorrect)
        Me.Controls.Add(Me.lblWord47)
        Me.Controls.Add(Me.lblWord48)
        Me.Controls.Add(Me.lblWord49)
        Me.Controls.Add(Me.lblWord50)
        Me.Controls.Add(Me.lblWord46)
        Me.Controls.Add(Me.lblWord42)
        Me.Controls.Add(Me.lblWord43)
        Me.Controls.Add(Me.lblWord44)
        Me.Controls.Add(Me.lblWord45)
```

```
    Me.Controls.Add(Me.lblWord41)
    Me.Controls.Add(Me.lblWord37)
    Me.Controls.Add(Me.lblWord38)
    Me.Controls.Add(Me.lblWord39)
    Me.Controls.Add(Me.lblWord40)
    Me.Controls.Add(Me.lblWord36)
    Me.Controls.Add(Me.lblWord32)
    Me.Controls.Add(Me.lblWord33)
    Me.Controls.Add(Me.lblWord34)
    Me.Controls.Add(Me.lblWord35)
    Me.Controls.Add(Me.lblWord31)
    Me.Controls.Add(Me.lblWord27)
    Me.Controls.Add(Me.lblWord28)
    Me.Controls.Add(Me.lblWord29)
    Me.Controls.Add(Me.lblWord30)
    Me.Controls.Add(Me.lblWord26)
    Me.Controls.Add(Me.lblWord22)
    Me.Controls.Add(Me.lblWord23)
    Me.Controls.Add(Me.lblWord24)
    Me.Controls.Add(Me.lblWord25)
    Me.Controls.Add(Me.lblWord21)
    Me.Controls.Add(Me.lblWord17)
    Me.Controls.Add(Me.lblWord18)
    Me.Controls.Add(Me.lblWord19)
    Me.Controls.Add(Me.lblWord20)
    Me.Controls.Add(Me.lblWord16)
    Me.Controls.Add(Me.lblWord12)
    Me.Controls.Add(Me.lblWord13)
    Me.Controls.Add(Me.lblWord14)
    Me.Controls.Add(Me.lblWord15)
    Me.Controls.Add(Me.lblWord11)
    Me.Controls.Add(Me.lblWord7)
    Me.Controls.Add(Me.lblWord8)
    Me.Controls.Add(Me.lblWord9)
    Me.Controls.Add(Me.lblWord10)
    Me.Controls.Add(Me.lblWord6)
    Me.Controls.Add(Me.lblWord2)
    Me.Controls.Add(Me.lblWord3)
    Me.Controls.Add(Me.lblWord4)
    Me.Controls.Add(Me.lblWord5)
    Me.Controls.Add(Me.lblWord1)
    Me.Name = "frmAudSpeech"
    Me.Text = "Aud Speech"
    Me.ResumeLayout(False)

End Sub
```

```
#End Region

    Public Class SoundClass
        Declare Auto Function PlaySound Lib "winmm.dll" (ByVal name _
            As String, ByVal hmod As Integer, ByVal flags As Integer) As Integer

        Public Const SND_SYNC = &H0
        Public Const SND_ASYNC = &H1
        Public Const SND_FILENAME = &H20000
        Public Const SND_RESOURCE = &H40004

        Public Sub PlaySoundFile(ByVal filename As String)
            PlaySound(filename, Nothing, SND_FILENAME Or SND_ASYNC)
        End Sub

    End Class
    Dim arrWord() As String = {"laud", "boat", "pool", "nag", "limb", "shout", "sub",
"vine", "dime", "goose", _
                               "whip", "tough", "puff", "keen", "death", "sell", "take", "fall",
"raise", "third", _
                               "gap", "fat", "keg", "jar", "door", "love", "sure", "knock", "choice",
"hash", _
                               "lot", "raid", "hurl", "moon", "page", "yes", "reach", "king",
"home", "rag", _
                               "which", "week", "size", "mode", "bean", "tip", "chalk", "jail",
"burn", "kite"}
    Dim currentIndex As Integer = 0
    Dim arrLabel()



    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

        ReDim arrLabel(49)

        arrLabel(0) = lblWord1
        arrLabel(1) = lblWord2
        arrLabel(2) = lblWord3
        arrLabel(3) = lblWord4
        arrLabel(4) = lblWord5
        arrLabel(5) = lblWord6
        arrLabel(6) = lblWord7
        arrLabel(7) = lblWord8
        arrLabel(8) = lblWord9
```

```
arrLabel(9) = lblWord10
arrLabel(10) = lblWord11
arrLabel(11) = lblWord12
arrLabel(12) = lblWord13
arrLabel(13) = lblWord14
arrLabel(14) = lblWord15
arrLabel(15) = lblWord16
arrLabel(16) = lblWord17
arrLabel(17) = lblWord18
arrLabel(18) = lblWord19
arrLabel(19) = lblWord20
arrLabel(20) = lblWord21
arrLabel(21) = lblWord22
arrLabel(22) = lblWord23
arrLabel(23) = lblWord24
arrLabel(24) = lblWord25
arrLabel(25) = lblWord26
arrLabel(26) = lblWord27
arrLabel(27) = lblWord28
arrLabel(28) = lblWord29
arrLabel(29) = lblWord30
arrLabel(30) = lblWord31
arrLabel(31) = lblWord32
arrLabel(32) = lblWord33
arrLabel(33) = lblWord34
arrLabel(34) = lblWord35
arrLabel(35) = lblWord36
arrLabel(36) = lblWord37
arrLabel(37) = lblWord38
arrLabel(38) = lblWord39
arrLabel(39) = lblWord40
arrLabel(40) = lblWord41
arrLabel(41) = lblWord42
arrLabel(42) = lblWord43
arrLabel(43) = lblWord44
arrLabel(44) = lblWord45
arrLabel(45) = lblWord46
arrLabel(46) = lblWord47
arrLabel(47) = lblWord48
arrLabel(48) = lblWord49
arrLabel(49) = lblWord50

'Fill all 50 slots with the words from an array
For i As Integer = 0 To arrLabel.Length - 1
    arrLabel(i).Text = i + 1 & ". " & arrWord(i)
Next
```

```
    End Sub

    Private Sub btnCorrect_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCorrect.Click
        Dim SoundInst As New SoundClass
        SoundInst.PlaySoundFile("C:\NU-6wavs\" & arrWord(currentIndex) & ".wav")
        arrLabel(currentIndex - 1).BackColor = System.Drawing.SystemColors.Menu
        arrLabel(currentIndex - 1).ForeColor = System.Drawing.SystemColors.Highlight
        arrLabel(currentIndex).BackColor = System.Drawing.Color.Yellow
        arrLabel(currentIndex - 1).Image = Image.FromFile("C:\NU-6wavs\correct.jpg")
        lblTotalCorrect.Text = lblTotalCorrect.Text + 1
        lblTotalOutOf.Text = lblTotalOutOf.Text + 1
        lblPercentCorrect.Text = CInt(lblTotalCorrect.Text / lblTotalOutOf.Text * 100)
        currentIndex = currentIndex + 1
        If currentIndex = arrWord.Length Then 'End of word list
            btnCorrect.Enabled = False
            btnIncorrect.Enabled = False
        End If
    End Sub

    Private Sub btnStartRestart_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnStartRestart.Click

        For i As Integer = 0 To arrLabel.Length - 1
            arrLabel(i).BackColor = System.Drawing.SystemColors.Menu
            arrLabel(i).ForeColor = System.Drawing.SystemColors.ControlText
            arrLabel(i).Image = Nothing
            'Remove all the images
        Next
        lblTotalCorrect.Text = 0
        lblTotalOutOf.Text = 0
        lblPercentCorrect.Text = 0
        currentIndex = 0

        Dim SoundInst As New SoundClass
        SoundInst.PlaySoundFile("C:\NU-6wavs\" & arrWord(currentIndex) & ".wav")
        lblWord1.BackColor = System.Drawing.Color.Yellow
        currentIndex = currentIndex + 1
        btnCorrect.Enabled = True
        btnIncorrect.Enabled = True
        btnSkip.Enabled = True
        btnRepeat.Enabled = True
        btnFinishCorrect.Enabled = True
```

```
        btnFinishIncorrect.Enabled = True
    End Sub

    Private Sub btnIncorrect_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnIncorrect.Click
        Dim SoundInst As New SoundClass
        SoundInst.PlaySoundFile("C:\NU-6wavs\" & arrWord(currentIndex) & ".wav")
        arrLabel(currentIndex - 1).BackColor = System.Drawing.SystemColors.Menu
        arrLabel(currentIndex - 1).ForeColor = System.Drawing.SystemColors.Highlight
        arrLabel(currentIndex).BackColor = System.Drawing.Color.Yellow
        arrLabel(currentIndex - 1).Image = Image.FromFile("C:\NU-6wavs\incorrect.jpg")
        lblTotalOutOf.Text = lblTotalOutOf.Text + 1
        lblPercentCorrect.Text = CInt(lblTotalCorrect.Text / lblTotalOutOf.Text * 100)
        currentIndex = currentIndex + 1
        If currentIndex = arrWord.Length Then 'End of word list
            btnCorrect.Enabled = False
            btnIncorrect.Enabled = False
        End If
    End Sub

    Private Sub btnCalibrate_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnCalibrate.Click
        Dim SoundInst As New SoundClass
        SoundInst.PlaySoundFile("C:\NU-6wavs\calibrate.wav")
    End Sub
    Private Sub btnRepeat_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnRepeat.Click
        Dim SoundInst As New SoundClass
        SoundInst.PlaySoundFile("C:\NU-6wavs\" & arrWord(currentIndex - 1) & ".wav")
    End Sub

    Private Sub btnSkip_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnSkip.Click
        If currentIndex >= arrWord.Length Then
            arrLabel(currentIndex - 1).BackColor = System.Drawing.SystemColors.Menu
            arrLabel(currentIndex - 1).ForeColor = System.Drawing.SystemColors.Highlight
            lblPercentCorrect.Text = CInt(lblTotalCorrect.Text / lblTotalOutOf.Text * 100)
            btnCorrect.Enabled = False
            btnIncorrect.Enabled = False
            btnSkip.Enabled = False
            btnRepeat.Enabled = False
            btnFinishCorrect.Enabled = False
            btnFinishIncorrect.Enabled = False
            If currentIndex = arrWord.Length - 1 Then 'End of word list
                btnCorrect.Enabled = False
                btnIncorrect.Enabled = False
```

```
            End If
        Else
            Dim SoundInst As New SoundClass
            SoundInst.PlaySoundFile("C:\NU-6wavs\" & arrWord(currentIndex) & ".wav")
            arrLabel(currentIndex - 1).BackColor = System.Drawing.SystemColors.Menu
            arrLabel(currentIndex - 1).ForeColor = System.Drawing.SystemColors.Highlight
            arrLabel(currentIndex).BackColor = System.Drawing.Color.Yellow
            currentIndex = currentIndex + 1
            If currentIndex = arrWord.Length Then 'End of word list
                btnCorrect.Enabled = False
                btnIncorrect.Enabled = False
            End If
        End If
    End Sub

    Private Sub btnFinishCorrect_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnFinishCorrect.Click
        arrLabel(currentIndex - 1).BackColor = System.Drawing.SystemColors.Menu
        arrLabel(currentIndex - 1).ForeColor = System.Drawing.SystemColors.Highlight
        lblTotalCorrect.Text = lblTotalCorrect.Text + 1
        lblTotalOutOf.Text = lblTotalOutOf.Text + 1
        lblPercentCorrect.Text = CInt(lblTotalCorrect.Text / lblTotalOutOf.Text * 100)
        arrLabel(currentIndex - 1).Image = Image.FromFile("C:\NU-6wavs\correct.jpg")
        btnCorrect.Enabled = False
        btnIncorrect.Enabled = False
        btnSkip.Enabled = False
        btnRepeat.Enabled = False
        btnFinishCorrect.Enabled = False
        btnFinishIncorrect.Enabled = False
    End Sub

    Private Sub btnFinishIncorrect_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnFinishIncorrect.Click
        arrLabel(currentIndex - 1).BackColor = System.Drawing.SystemColors.Menu
        arrLabel(currentIndex - 1).ForeColor = System.Drawing.SystemColors.Highlight
        lblTotalOutOf.Text = lblTotalOutOf.Text + 1
        lblPercentCorrect.Text = CInt(lblTotalCorrect.Text / lblTotalOutOf.Text * 100)
        arrLabel(currentIndex - 1).Image = Image.FromFile("C:\NU-6wavs\incorrect.jpg")
        btnCorrect.Enabled = False
        btnIncorrect.Enabled = False
        btnSkip.Enabled = False
        btnRepeat.Enabled = False
        btnFinishCorrect.Enabled = False
        btnFinishIncorrect.Enabled = False
    End Sub
                                        End Class
```

Appendix B

Frequency Responses for Experiment 1

Figure B1. Frequency Response of Dell Optiplex GX270, ID: 61JT851

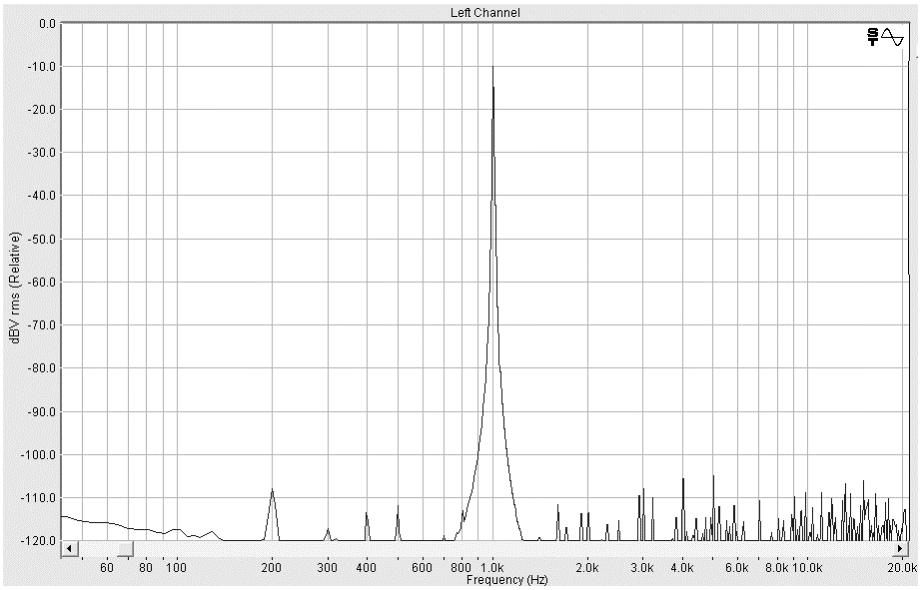


Figure B2. Frequency Response of Dell Optiplex GX260, ID: BGX5431

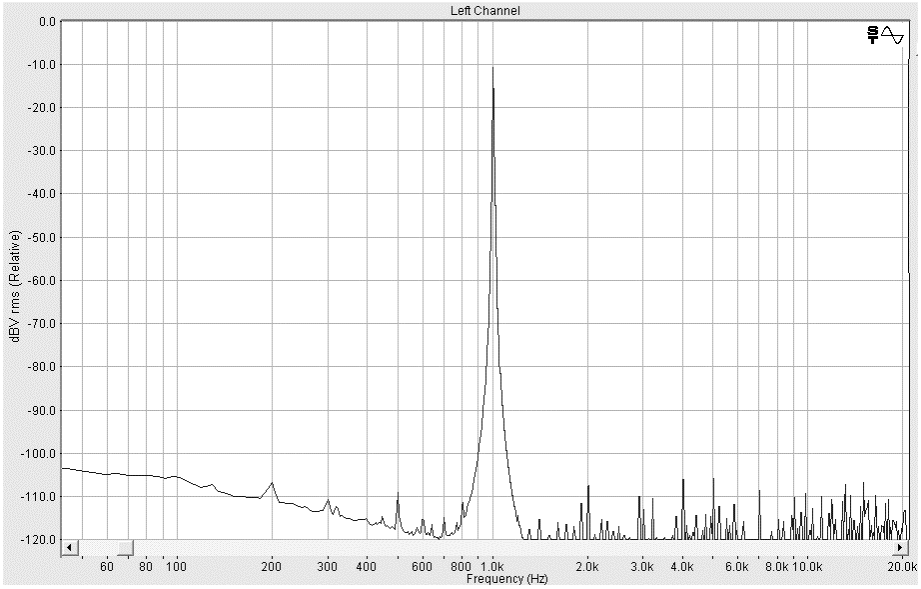


Figure B3. Frequency Response of Dell Optiplex GX270, ID: GZHT851

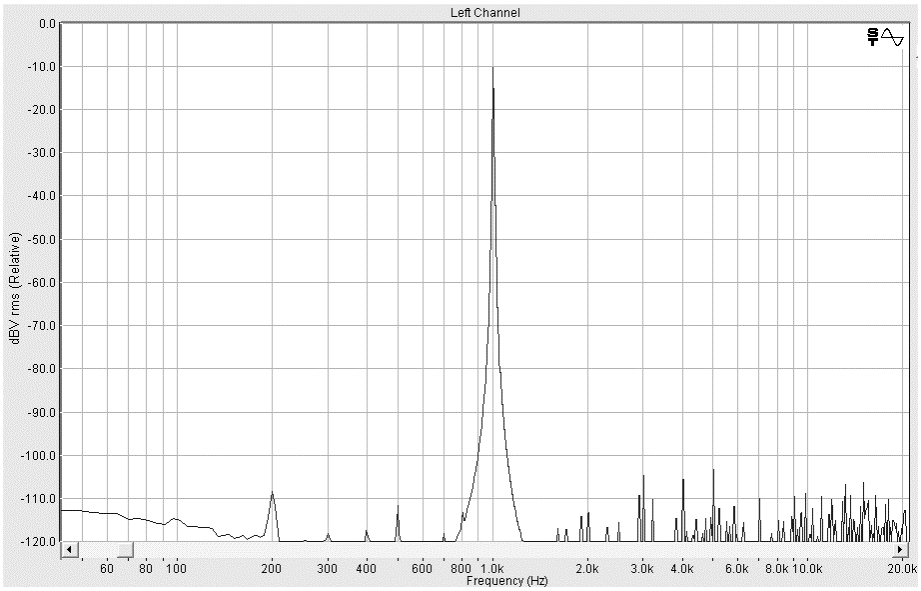


Figure B4. Frequency Response of Dell Optiplex GX280, ID: 44J9T71

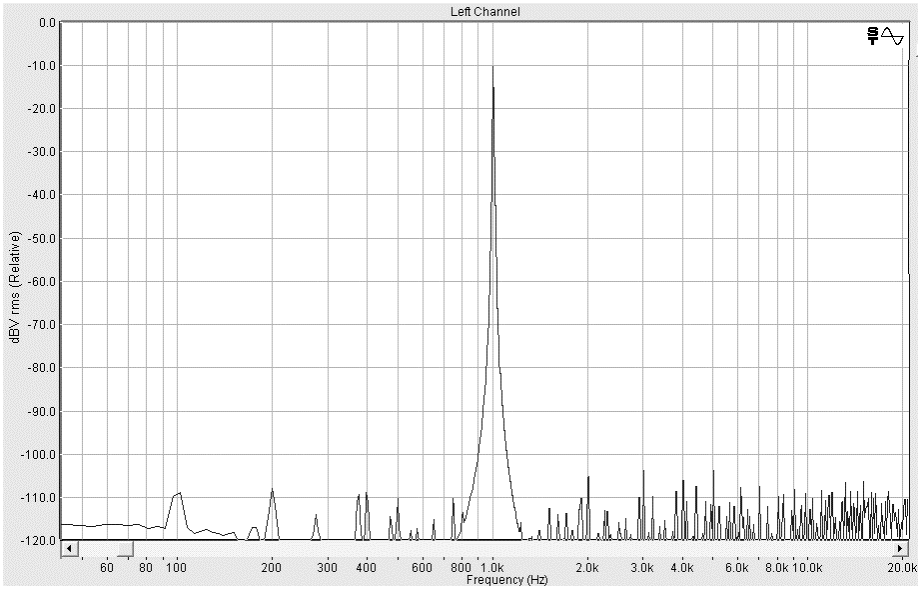


Figure B5. Frequency Response of Dell Optiplex GX520, ID: 83VW091

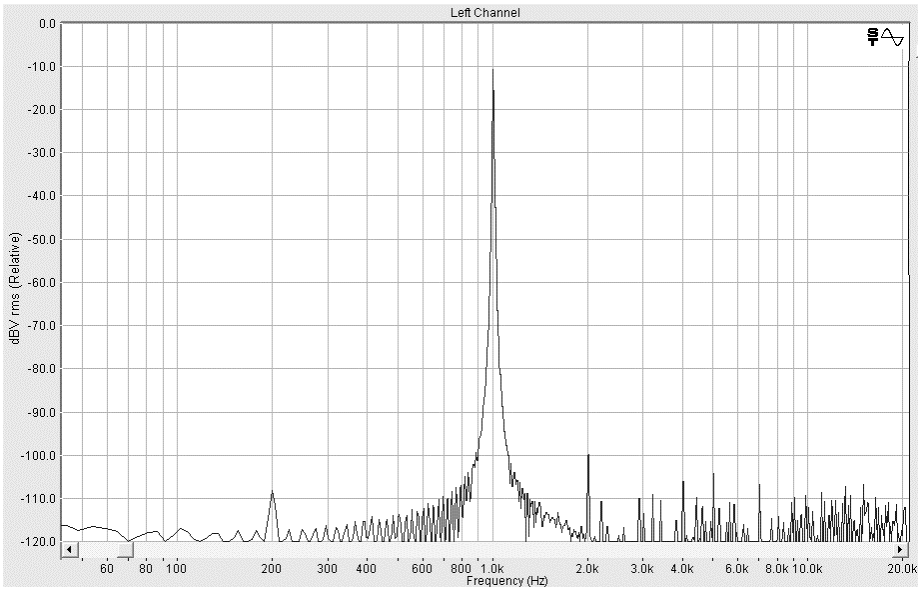


Figure B6. Frequency Response of Dell Optiplex GX270, ID: 7NBC451

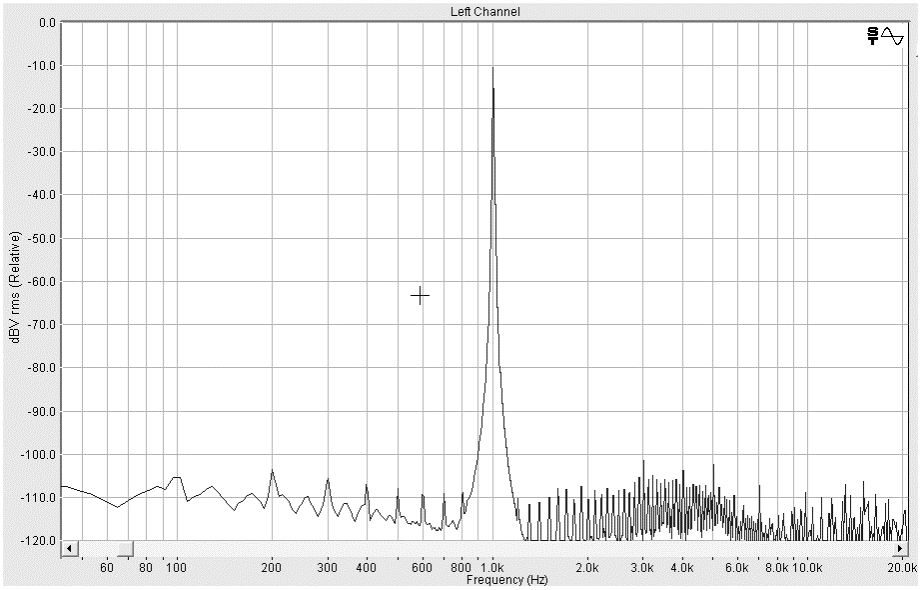


Figure B7. Frequency Response of Insignia D400, ID: 4000206T

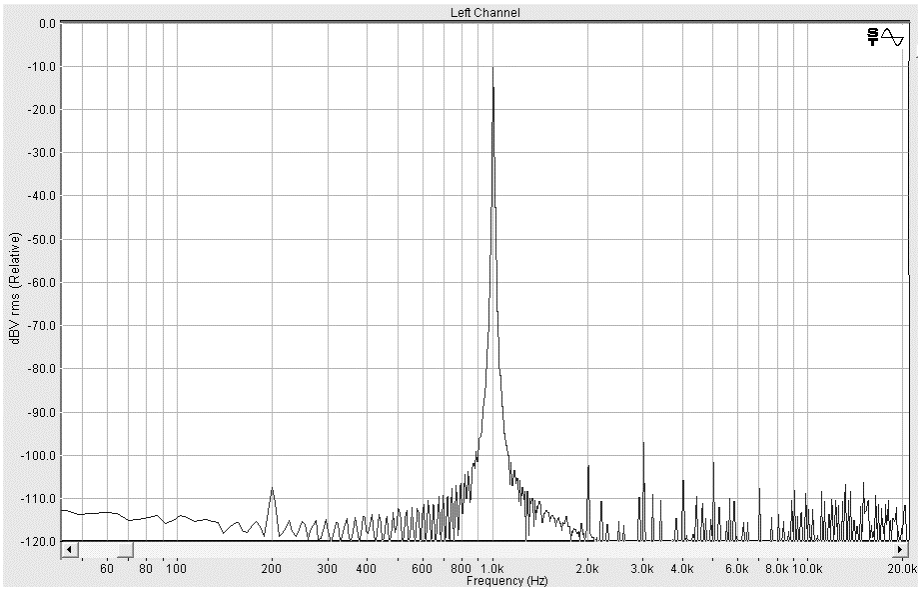


Figure B8. Frequency Response of Dell Inspiron 600m (Laptop), ID: 56N2889

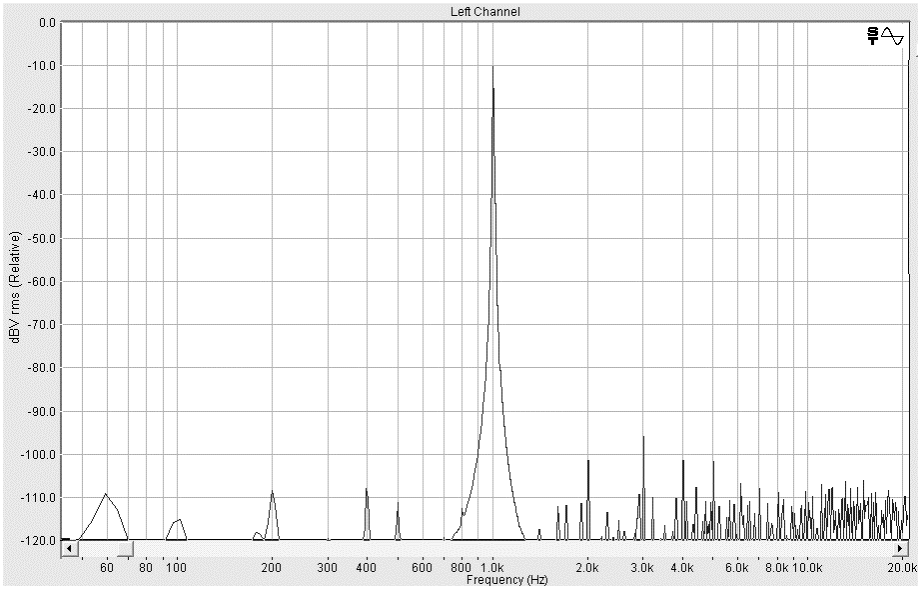


Figure B9. Frequency Response of HP dx2000 MT, ID: 5080MWK

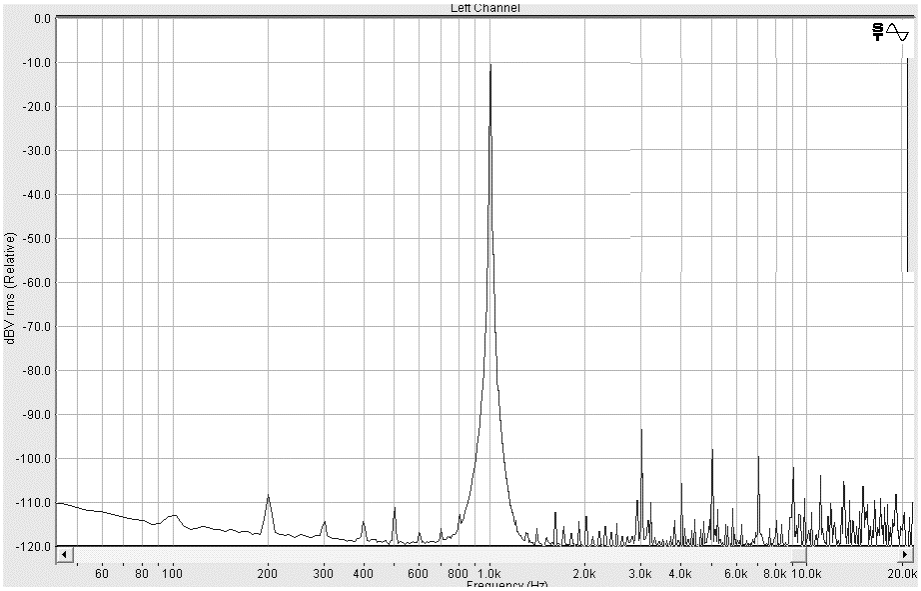


Figure B10. Frequency Response of HP-Compaq d220 MT, ID: 40900DR

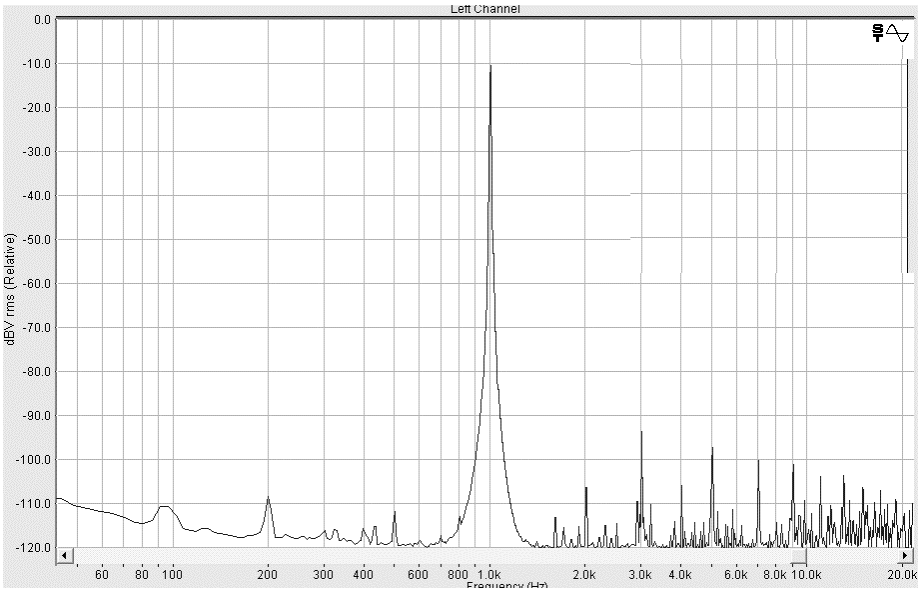


Figure B11. Frequency Response of HP-Compaq d330 uT, ID: 402061D

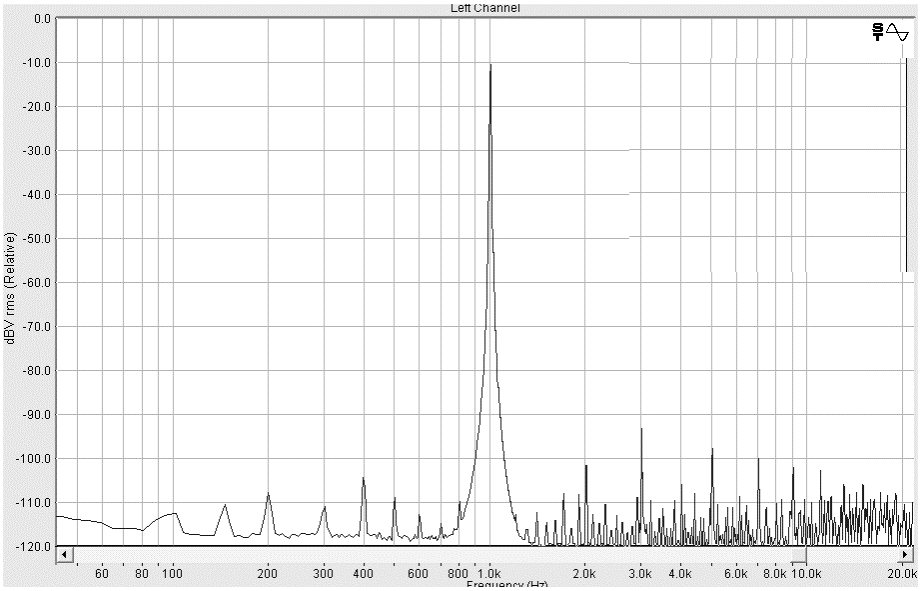


Figure B12. Frequency Response of Dell Latitude D600 (Laptop), ID: 4CK6014

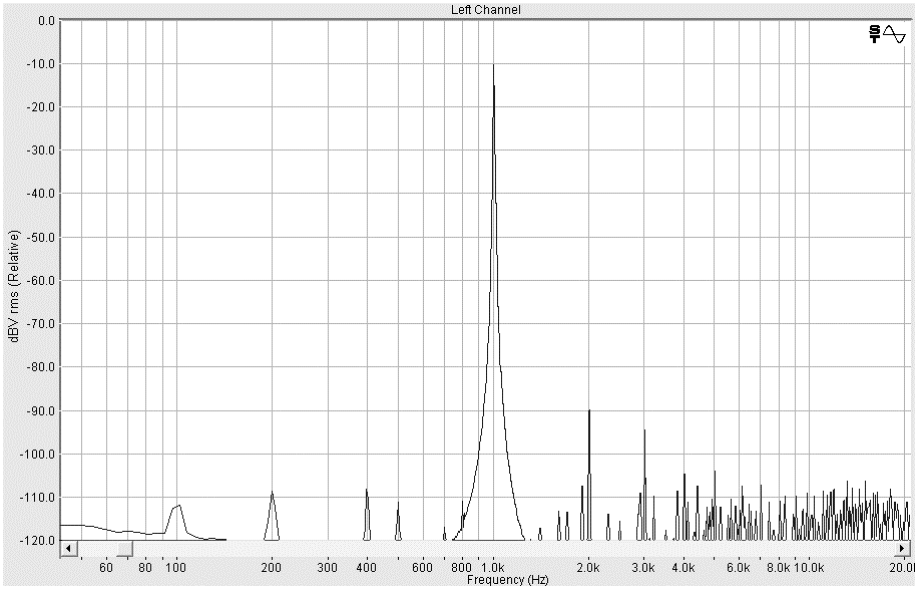


Figure B13. Frequency Response of Toshiba Satellite 325CDS (Laptop), ID: 33999A

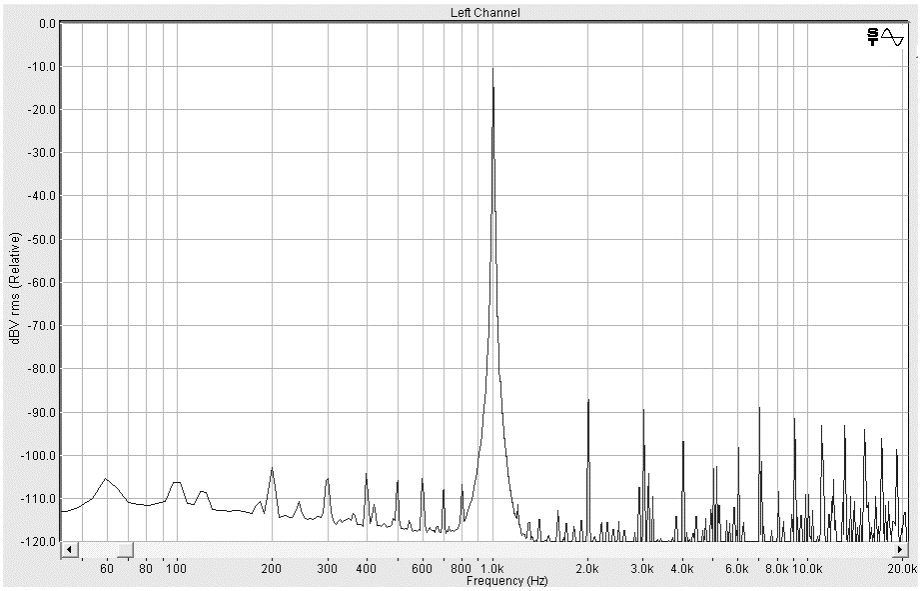


Figure B14. Frequency Response of PC with Epox Motherboard, ID: A99999

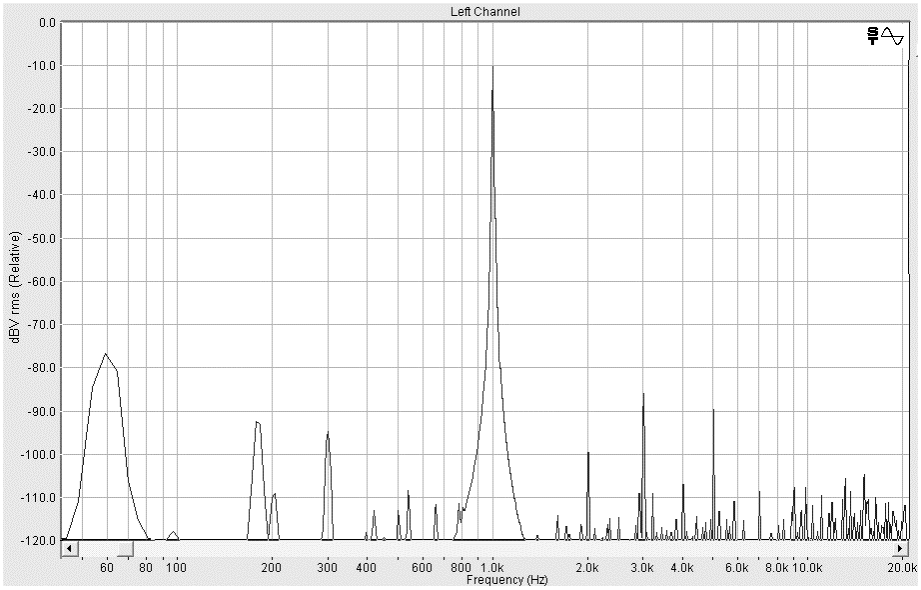


Figure B15. Frequency Response of HP dx5150 MT, ID: 5500JG9

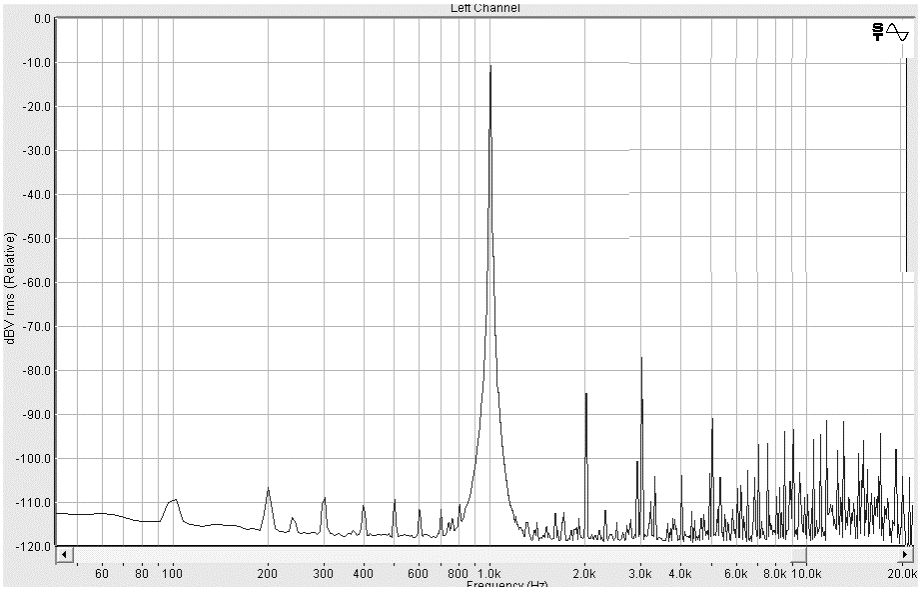


Figure B16. Frequency Response of PC with Mach Speed Motherboard, ID: B99999

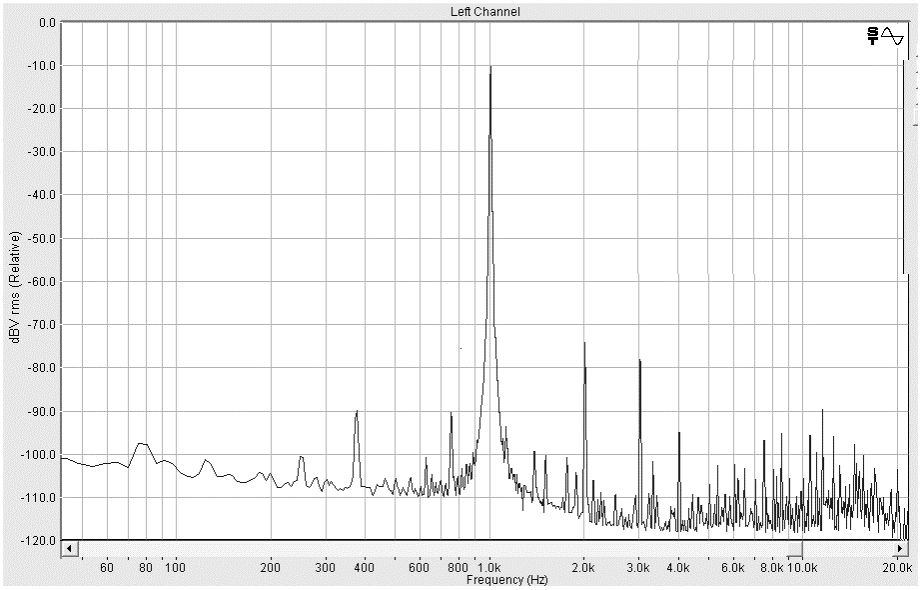
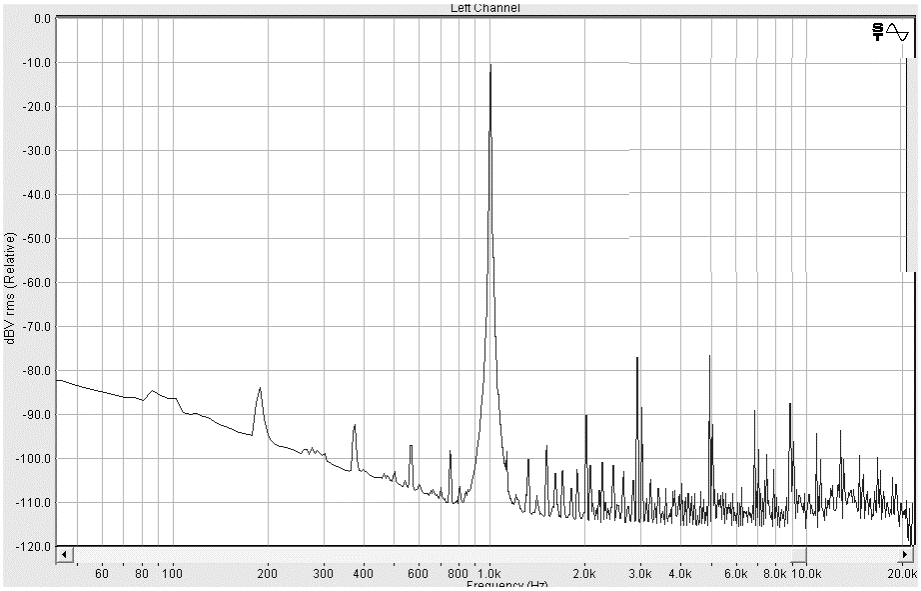


Figure B17. Frequency Response of Vsquared Computer, ID: 906085



References

American National Standards Institute. (1996). American national standard specification for audiometers. (ANSI S3.6-1996). New York: American National Standards Institute.

ASHA. (1988). Guidelines for determining threshold level for speech. *Asha, 30*(3), 85-89.

Audio Engineering Society Standards Committee. (2000). AES-6id-2000: AES information document for digital audio – Personal computer audio quality measurements. Audio Engineering Society, Inc.

Balo, S., Bangalore, N., Chavez, M., Day, R., Deyell, S., Ellis, P., et al. (2004). *Adobe Audition 1.5 User Guide for Windows* [Computer software manual].

Beattie, R. C., & Zipp, J. A. (1990). Range of intensities yielding PB max and the threshold for monosyllabic words for hearing-impaired subjects. *Journal of Speech and Hearing Disorders, 55*(3), 417-426.

Bentler, R. A., Tubbs, J. L., Egge, J. L., Flamme, G. A., & Dittberner, A. B. (2004). Evaluation of an adaptive directional system in a DSP hearing aid. *American Journal of Audiology, 13*(1), 73-79.

Brandy, W. T. (1966). Reliability of voice tests of speech discrimination. *Journal of Speech and Hearing Research, 9*(3), 461-465.

Carhart, R. (1965). Problems in the measurement of speech discrimination. *Archives of Otolaryngology—Head & Neck Surgery, 82*, 253–260.

Causey, G. D., Hermanson, C. L., Hood, L. J., & Bowling, L. S. (1983). A comparative evaluation of the Maryland NU

6 Auditory Test. *Journal of Speech and Hearing Disorders, 48*, 62-69.

Clark J. E. (1981). Four PB word lists for Australian English. *Australian Journal of Audiology 3*(1), 21-31.

Cullen, R. D., Higgins, C., Buss, E., Clark, M., Pillsbury, H. C., & Buckman, C. A. (2004). Cochlear implantation in patients with substantial residual hearing. *The Laryngoscope 114*(12), 2218-2223.

Department of Veterans Affairs. (1991). Speech recognition and identification materials. Version 1.1. [CD]. Long Beach, CA: VA Medical Center.

Dirks, D. D., Kamm, C., Bower, D., & Betsworth, A. (1977). Use of performance-intensity functions for diagnosis. *Journal of Speech and Hearing Disorders, 42*(3), 408-415.

Dubno, J. R., Lee, F. S., Klein, A. J., Matthews, L. J., & Lam, C. F. (1995). Confidence limits for maximum word-recognition scores. *Journal of Speech and Hearing Research, 38*(2), 490-502.

Dubno, J. R., Lee, F. S., Matthews, L. J., & Mills, J. H. (1997). Age-related and gender-related changes in monaural speech recognition. *Journal of Speech, Language, and Hearing Research, 40*(2), 444-452.

Egan, J. (1948). Articulation testing methods. *The Laryngoscope, 58*, 955-991.

Elkins, E. F. (1970). Analyses of the phonetic composition and word familiarity attributes of CNC intelligibility word lists. *Journal of Speech and Hearing Disorders, 35*(2), 156-160.

Epstein, A. (1978). Speech audiometry. *Otolaryngologic Clinics of North America, 11*(3), 667-676.

Fletcher, H. (1929). *Speech and hearing* [Electronic version]. Brooklyn, NY: D.Van Nostrand Company, Inc.

Flynn, M. C., Dowell, R. C., & Clark, G. M. (1998). Aided speech recognition abilities of adults with a severe or severe-to-profound hearing loss. *Journal of Speech, Language, and Hearing Research, 41*(2), 285-299.

Gatehouse, S., Naylor, G., & Elberling, C. (2003). Benefits from hearing aids in relation to the interaction between the user and the environment. *International Journal of Audiology, 42 Supplement 1*, S77-S85.

Gates, G. A., Feeney, M. P., & Higdon, R. J. (2003). Word recognition and the articulation index in older listeners with probable age-related auditory neuropathy. *Journal of the American Academy of Audiology, 14*(10), 574-581.

Guidorzi, P. (2005). *Sample Companion PRO 3.8 – User Manual* [Computer software manual].

Hagerman, B. (1976). Reliability in the determination of speech discrimination. *Scandinavian Audiology, 5*(4), 219-228.

Hirsh, I. J., Davis, H., Silverman, S. R., Reynolds, E. G., Eldert, E., & Benson, R. W. (1952). Development of materials for speech audiometry. *Journal of Speech and Hearing Disorders, 17,* 321-337.

Hnath-Chisolm, T. (1997). Context effects in auditory training with children. *Scandinavian Audiology Supplementum, 47*, 64-69.

Holden, L. K., Vandali, A. E., Skinner, M. W., Fourakis, M. S., & Holden, T. A. (2005). Speech recognition with the advanced combination encoder and transient emphasis spectral maxima strategies in nucleus 24 recipients. *Journal of Speech, Language, and Hearing Research, 48*(3), 681-701.

Hood, J. D., & Poole, J. P. (1977). Improving the reliability of speech audiometry. *British Journal of Audiology, 11*(4), 93-101.

Hudgins, C. V., Hawkins, J. E., Jr., Karlin, J. E., & Stevens, S. S. (1947). The development of recorded auditory tests for measuring hearing loss for speech. *The Laryngoscope, 57*, 57-89.

Hurley, R. M., & Sells, J. P. (2003). An abbreviated word recognition protocol based on item difficulty. *Ear and Hearing, 24*(2), 111-118.

Jerger, J. (1998). Editorial: Audiometric practices. *Journal of the American Academy of Audiology, 9*(2).

Jerger, J., Jerger, S., & Pirozzolo, F. (1991). Correlational analysis of speech audiometric scores, hearing loss, age, and cognitive abilities in the elderly. *Ear and Hearing, 12*(2), 103-109.

Kamm, C. A., Morgan, D. E., & Dirks, D. D. (1983). Accuracy of adaptive procedure estimates of PB-max level. *Journal of Speech and Hearing Disorders, 48*(2), 202-209.

Laptop Audio Upgrade Kit – Turtle Beach. (n.d.) Retrieved May 5, 2006 from http://www.turtlebeach.com/site/products/lapupkit/.

Lehiste, I., & Peterson, G. E. (1959). Linguistic considerations in the study of speech intelligibility. *Journal of the Acoustical Society of America, 31*, 280-286.

Martin, F. N., Champlin, C. A., & Chambers, J. A. (1998). Seventh survey of audiometric practices in the United States. *Journal of the American Academy of Audiology, 9*(2), 95-104.

Martin, F. N., & Jansen, R. M. (1985). Speech reception thresholds using conventional vs high-frequency spondees in normals and in subjects with marked high-frequency sensorineural loss. *Journal of Auditory Research, 25*(2), 133-142.

Mendel, L. L., & Danhauer, J. L. (1997). *Audiologic evaluation and management and speech perception assessment*. San Diego, CA: Singular Publishing Group, Inc.

Meyer, D. H., & Mishler, E. T. (1985). Rollover measurements with Auditec NU-6 word lists. *Journal of Speech and Hearing Disorders, 50*(4), 356-360.

Penrod, J. P. (1979). Talker effects on word discrimination scores of adults with sensoineural hearing impairment. *Journal of Speech and Hearing Disorders, 44*(3), 340-349.

Ptok, M. (2000). Otoacoustic emissions, auditory evoked potentials, pure tone thresholds and speech intelligibility in cases of auditory neuropathy. HNO, 48(1), 28-32. Abstract retrieved January 27, 2006, from PubMed database site http://www.pubmed.gov.

Schwartz, D. M., & Surr, R. K. (1979). Three experiments on the California Consonant Test. *Journal of Speech and Hearing Disorders, 44*(1), 61-72.

Silverman, S. R. (1983). Historical foundations of speech audiometry. In D. F. Konkle & W. F. Rintelmann (Eds.), *Principles of Speech Audiometry* (pp. 11-24). Baltimore, MD: University Park Press.

Stach, B. A., Davis-Thaxton, M. L., & Jerger, J. (1995). Improving the efficiency of speech audiometry: computer-based approach. *Journal of the American Academy of Audiology, 6*(4), 330-333.

Svirsky, M. A., Teoh, S. W., & Neuburger, H. (2004). Development of language and speech perception in congenitally, profoundly deaf children as a function of age at cochlear implantation. *Audiology & Neuro-otology, 9*(4), 224-233.

Teoh, S. W., Pisoni, D. B., & Miyamoto, R. T. (2004). Cochlear implantation in adults with prelingual deafness. Part I. Clinical results. *The Laryngoscope, 114*(9), 1536-1540.

Thornton, A. R., & Raffin, M. J. (1978). Speech-discrimination scores modeled as a binomial variable. *Journal of Speech and Hearing Research, 21*(3), 507-518.

Tillman, T. W., & Carhart, R. (1966). An expanded test for speech discrimination utilizing CNC monosyllabic words. Northwestern University Auditory Test No. 6 (Tech. Rep. No. SAM-TDR-66-55). Brooks Air Force Base, TX: USAF School of Aerospace Medicine.

Tillman, T. W., & Olsen, W. O. (1973). Speech audiometry. In J. Jerger (Ed.), *Modern developments in audiology* (2nd ed., pp. 37-74). New York: Academic Press.

Tobias, J. V., (1964). On phonemic analysis of speech discrimination tests. *Journal of Speech and Hearing Research, 8*, 363-369.

UK Cochlear Implant Study Group. (2004). Criteria of candidacy for unilateral cochlear implantation in postlingually deafened adults II: cost-effectiveness analysis. *Ear and Hearing, 25*(4), 336-360.

Wilson, R. H., Preece, J. P., & Thornton, A. R. (1990). Clinical use of the compact disc in speech audiometry. *ASHA, 32*(6-7), 47,51.

CURRICULUM VITA

NAME: Matthew H. Perry

PERMANENT ADDRESS: 333 Spenceola Pkwy, Forest Hill, Maryland 21050

PROGRAM OF STUDY: Audiology

DEGREE AND DATE TO BE CONFERRED: Doctorate of Audiology, 2007

SECONDARY EDUCATION: Fallston High School, 2301 Carrs Mill Road, Fallston, Maryland 21047, Graduation: 1996

Collegiate Institution	Dates	Degree	Date of Degree
Ricks College	1996-1997, 2000	Associate of Arts	Aug 2000
University of Utah	2000-2002	Bachelor of Arts	May 2001
Towson University	2003-2007	Doctorate of Audiology	May 2007

Major: Speech and Hearing Science (University of Utah)