

# **Implementation & Evaluation of Directional Sound System to Support Audio Navigation**

**Greg R. Bullock**  
gregbullock@bonstio.net

September 2003  
MSc in Human Communication and Computing



**Department of Computer Science**

## **Declaration**

This dissertation is submitted to the University of Bath in accordance with the requirements of the degree of Master of Science in Human Communication and Computing in the Department of Computer Science. No portion of the work in this thesis has been submitted in support of an application for any other degree or qualification of this or any other university or institution of learning. Except where specifically acknowledged, it is the work of the author.

**Author:** Greg R. Bullock

**Signature:**

<b>1. Introduction.....</b>	<b>5</b>
1.1 Abstract.....	5
1.2 Motivation.....	5
<b>2. Background.....</b>	<b>7</b>
2.1. 3D Sound.....	7
2.2. Alternative Stereo Formats.....	7
2.2.1. <i>Monophonic Sound</i> .....	7
2.2.2. <i>Stereophonic Sound</i> .....	7
2.2.3. <i>Quadraphonic Sound</i> .....	8
2.2.4. <i>Dolby Stereo</i> .....	8
2.2.5. <i>Dolby Digital</i> .....	9
2.2.6. <i>Head-Related Transfer Functions</i> .....	9
2.3. Problems with Spatial Sound.....	10
2.4. Auditory Cues.....	12
2.4.1. <i>Inter-aural Time Difference</i> .....	12
2.4.2. <i>Pinna response</i> .....	12
2.4.3. <i>Head shadow</i> .....	13
2.4.4. <i>Shoulder echo</i> .....	14
2.4.5. <i>Head Motion</i> .....	14
2.4.6. <i>Vision</i> .....	15
2.4.7. <i>Precision of Localization</i> .....	15
2.5. Synopsis of Relevant Papers.....	15
2.5.1. <i>"Head Motion and Latency Compensation on Localization of 3D Sound in Virtual Reality"</i> .....	15
2.5.2. <i>"Localization with non-individualized virtual acoustic display cues"</i> .....	16
<b>3. Project Proposal .....</b>	<b>18</b>
3.1. What?.....	18
3.2. Why?.....	18
3.3. How?.....	21
3.4. Resources Available.....	23
3.4.1. <i>Hardware</i> .....	23
3.4.2. <i>Software</i> .....	23
3.5. Research Question.....	24
<b>4. Implementation .....</b>	<b>25</b>
4.1. Implementation Choice.....	25
4.1.1. <i>Bryden 2D</i> .....	25
4.1.2. <i>Creative EAX</i> .....	25
4.1.3. <i>Aureal A3D</i> .....	25
4.1.4. <i>FMOD</i> .....	26
4.1.5. <i>KOAN</i> .....	26
4.2. The Development Phase.....	26
<b>5. Methodology .....</b>	<b>30</b>
5.1. Participants.....	30
5.2. Apparatus.....	30
5.3. Procedures.....	32
<b>6. Results.....</b>	<b>34</b>
6.1. Pilot Study.....	34
6.2. Observations & Amendments.....	35
6.3. Presentation of Results.....	36
6.4. Chart to Show Perceptual Inaccuracies of Ten Participants Lateralizing Five Different Sounds.....	38
6.5. Chart to Show Distribution of Perceptual Inaccuracy Measurements ..	39
6.6. Summarised Results.....	40
6.7. Discussion.....	41
6.8. Revised Presentation of Data.....	42

6.9.	Chart to Show Revised of Ten Participants Lateralizing Five Different Sounds.....	43
6.10.	Chart to Show Revised Distribution of Perceptual Inaccuracies.....	44
6.11.	Graph to Show Target Azimuth Versus Perceived Azimuth .....	45
6.12.	Graph to Show Relationship Between Compass Inaccuracy and Perceptual Inaccuracy .....	46
<b>7.</b>	<b>Conclusion .....</b>	<b>49</b>
7.1.	Summary of Results .....	49
7.2.	Evaluation and Future Work .....	49
7.2.1.	<i>Software</i> .....	50
7.2.2.	<i>Hardware</i> .....	51
7.2.3.	<i>Design</i> .....	52
7.3.	Implications .....	53
<b>8.</b>	<b>Appendices .....</b>	<b>55</b>
<b>A.</b>	<b>References .....</b>	<b>56</b>
A.1.	Books & Printed Journals.....	56
A.2.	Websites & Online Documents .....	56
<b>B.</b>	<b>Test Results – Raw Data .....</b>	<b>58</b>
B.1.	Participant Data Sheets.....	62
<b>C.</b>	<b>Source Code Listing .....</b>	<b>75</b>
C.1.	<i>Context.cpp</i> .....	76
C.2.	<i>Sensor.cpp</i> .....	84
C.3.	<i>Serial.cpp</i> .....	104
C.4.	<i>SettingsDlg.cpp</i> .....	110
C.5.	<i>StdAfx.cpp</i> .....	113
C.6.	<i>UStest.cpp</i> .....	114
C.7.	<i>UStestDlg.cpp</i> .....	117
<b>D.</b>	<b>Project Timetable.....</b>	<b>132</b>
<b>E.</b>	<b>Thanks and Acknowledgements .....</b>	<b>133</b>

## **1. Introduction**

### **1.1 Abstract**

This paper documents the implementation and evaluation of a prototypical directional sound system to support navigation, which responds to movements of the head. This is achieved through the use of a head mounted compass. Head movements have been cited as one of the most influential auditory cues which serve to assist the localization of sound in space. The paper then goes on to describe a methodology for assessing the extent to which the system supports sound lateralization.

Conclusions are drawn in section 7 whilst the raw data tables are presented in Appendix B.

The area of psychoacoustics is an interesting and multidisciplinary one, encompassing a wide range of skills such as programming, psychology and digital signal processing.

### **1.2 Motivation**

The Mobile Bristol project is collaboration between the University of Bristol and Hewlett Packard Laboratories which is concerned with wearable or ultra-portable computing.

The first event of the Mobile Bristol program was held at HP Labs, Bristol on 30<sup>th</sup> January 2002. The integration of the works of photographer Liz Millner and audio-artist Armin Elsaesser were exhibited in 'the street'. Visitors to the exhibition were invited to wear headphones with an integral ultrasonic sensor and a small shoulder bag while viewing the photographs. The leads from the sensor and headphones both made their way into the bag. Inside it was a HP Jornada 568 handheld computer with wireless network connectivity and an extension board handling the output from the sensor.

Equipped with this 'wearable client', the visitor then spent up to twenty minutes wandering around the exhibition, viewing the photographs and hearing music, woodland sounds and speech chosen to enhance their content. The particular sounds

heard by the visitor at any point were determined automatically by the system according to her location within the exhibition space. For example, as the visitor approached certain photographs, she might begin to hear atmospheric music appropriate to the scenes depicted. As she moved on to other images, the music might be replaced by natural woodland sounds, or by a spoken fragment of woodland mythology. The particular sounds heard by the visitor at any point were determined automatically by the system according to the location within the exhibition space.

It is envisaged that such an arrangement could be enhanced by using directionalized sound; rather than using only geographical location as an input, supplementary data about the users' orientation could be fed into the system allowing for implementation of a more advanced sound reproduction system. The new system would feed the sound to the user's headphones according to the direction they were facing. In order to ascertain this information, the introduction of a head mounted compass would be necessary.

## **2. Background**

### **2.1. 3D Sound**

Three dimensional sound or spatial sound as it is sometimes referred to as, is simply sound as we hear it in everyday life. It is possible for individual sounds to be distinguished by their tone, pitch, loudness, and by their location in space. It is this spatial location of the sound which imparts a three dimensional experience.

The continuous influx of sound from our surrounding environment provides us with a great deal of information about the world around us. Subtle reverberations and echoes in the immediate environment serve as cues to the brain which interprets them and translates them to specific information about the relative direction and distance of objects. The ability to identify both the distance and direction of a sound is known as localization. These cues also convey information about the size and nature of the surrounding environment. For example, a small room has far fewer echoes than a large hall or church.

### **2.2. Alternative Stereo Formats**

Many different audio formats have been developed over the years. What follows is a brief overview of some of the more well known ones.

#### **2.2.1. *Monophonic Sound***

Mono sound is delivered via one audio channel to one speaker. It is ideal for listening with just one ear but has few practical uses today as it has been surpassed by just about every other format. Some hi-fi's support the facility to use mono mode in place of stereo mode when, for example, the radio broadcast signal is weak. Under such circumstances mono can offer a better reception since the crackle and distortion is reduced.

#### **2.2.2. *Stereophonic Sound***

The standard format most widely in use today for sound recording and reproduction is two channel stereo sound. It is recorded with two microphones a few feet apart from one another and separated by an empty space. Most people should be familiar with stereo sound; it is heard commonly through personal stereos and was also predominant in cinemas until the advent of surround sound. During playback, the recording from one microphone goes into the left ear, while the recording from the other microphone is directed to the right ear. This imparts a sense of the sound's position as recorded by the microphones. However, stereo sound is frequently perceived as if the sounds' sources are located at a position inside the listener's head, rather than externalized as in reality. This is because humans do not usually hear sounds in the manner they are recorded in stereo, i.e. separated by empty space. Instead the human head sits in between and acts as a filter to the sound.

### 2.2.3. *Quadraphonic Sound*



This was an attempt to enhance stereo sound as became relatively popular during the 1970's. The idea behind quadraphonic sound was that by arranging four speakers around the listener spatialized sound could be delivered via four distinct sound channels – really just an extension of two-channel stereo – in such a way as to be able to project sound from anywhere in the horizontal plane. This technology ultimately failed due to poor marketing and because of the increased cost of additional hardware required.

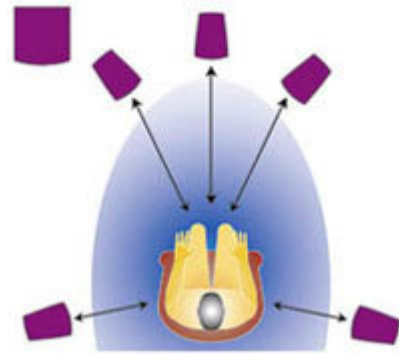
### 2.2.4. *Dolby Stereo*

Also known as *Dolby Surround*, this is an analogue sound system, again utilizing four distinct sound channels. With three speakers positioned in front of the listener (one to the left and to the right and one placed centrally) and one frequency limited speaker to the rear, this sound encoding standard was very

common until the emergence of superior digital films. The combination of these four speakers provided adequate ambient sounds but was not capable of supporting sound localization. The sound still relies upon encoding the sound into two channels. Special hardware is required to decode the sound from two channels in order that it can be played over four speakers as intended.

### 2.2.5. *Dolby Digital*

The standard most commonly in use today is known as Dolby digital. Like its predecessor, Dolby digital employs three speakers located to the front of the listener but instead of having one surround channel, there are now two full bandwidth speakers located to the rear. An optional omni-directional subwoofer provides extra bass. Due to the extra data involved in this setup, the data is compressed using an algorithm known as AC-3. Although this produces an impressive sound, in my opinion it remains insufficient to support listeners in accurately localizing sound.



### 2.2.6. *Head-Related Transfer Functions*

As has already been mentioned, head-related transfer functions (or HRTF's) refer to a set of functions which can be applied to a sound and which will affect the signal in such a way as to give the impression that the sound's source is located at a specific location in space. In short HRTF's provide the spectral cues required to localize a sound. HRTF's are now the focus of much engineering and psychoacoustic research since recent technological research has enabled the empirical analysis and evaluation of their use. Typically, HRTF's are computed by placing microphones inside the pinnae of a mannequin and sampling the impulse responses for the left and right ears as the sound source moves around a fixed radius from the head. HRTF's include ITD and IID data since the time delays are encoded in the filters.

HRTF's are applied by taking a mono sound and applying the left and the right components of filter which results in a two channel stereo signal. The result of having applied the left filter is played to the left ear and similarly the right. Applying a HRTF to a mono signal for a given azimuth will result in the sound appearing to originate from that same azimuth during playback.

The best means of delivery for this method of spatialization are to use stereo headphones since in that way two ears hear only what was intended for them to hear. Cross-talk is the term used to describe the sound being emitted from one speaker but perceived by the opposite ear with some small delay. This inevitably reduces the overall effectiveness of the spatial illusion and headphones address this minor problem.

### **2.3. Problems with Spatial Sound**

Most forms of audio processing which support sound localization tend to suffer the same set of drawbacks. Front-back / up-down reversals<sup>1</sup> are the most common misconception encountered when listening to spatialized sounds.<sup>2</sup> That is to say a person perceived a sound to be coming from above them when it is in fact coming from beneath them or vice versa. This error most often occurs when having not taken into account two important auditory cues; the subject's head movement and individualized pinnae response.

Additional synthetic cues can also be applied to a sound to help disambiguate its source – a simple high-pass filter applied to the sound when its azimuth falls between 90 and 270 degrees has the effect of muffling the sound. Although this is not strictly what happens in reality, the additional cue can serve to help localization.

The unnatural internalization of sounds (intracranially heard sounds), as perceived when listening through headphones also lessens the overall effect of realism. Although this is

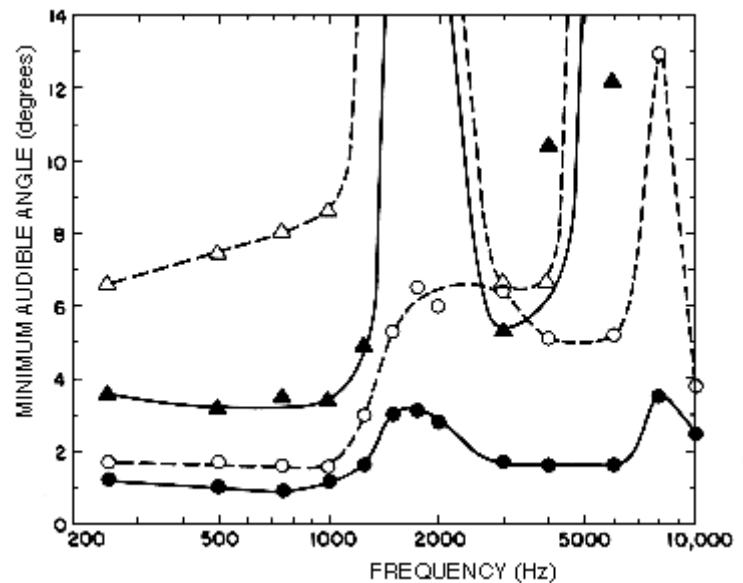
---

<sup>1</sup> Stevens & Newman, 1936

<sup>2</sup> [BURGESS92].

partly to do with the way the sound has been recorded / generated, it can be successfully addressed by adding reverberation effects.

The second aspect of localization is concerned with how well a person can detect a small change in the direction of a sound source. This is to do with the resolution of the person's auditory system. This resolution is dictated by how sensitive a person's auditory system is to detecting changes in the IID and ITD.



Minimum audible angle between successive pulses of tone as a function of the frequency and the direction of the source measured for angles (bottom to top at left hand side) 0°, 30°, 60° and 75° (source: Mills)

The smallest detectable change in the angle of a sound source, relative to the listener, is referred to the minimum audible angle (MAA). The MAA is greatly dependant upon the sound's azimuth and frequency. It should be no surprise that the MAA is at its smallest for sounds originating from a zero degree azimuth i.e. the sound originating directly from the front.

For sounds having frequency of less than 1000Hz, a shift in angle of only one degree is perceptible. Consistent with the duplex theory however, performance worsens between 1500 and 1800Hz. This is because at frequencies of above 1500Hz, ITD's are ambiguous cues for localization. Furthermore, up to the 1800Hz frequency intensity differences perceived between the two ears are insignificant and change little with azimuth. Again, performance deteriorates rapidly as the angle of sound projection moves away from zero degrees. Indeed, for angles of 60 and 75 degrees the MAA was indeterminably large for frequencies around the 1800Hz range.

## 2.4. Auditory Cues

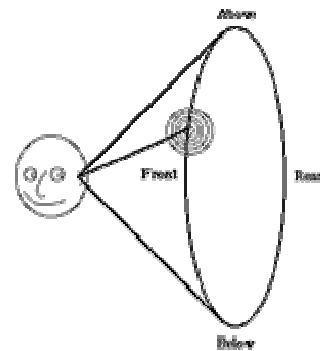
A number of auditory cues assist us in accurately localizing sounds. These cues are a result of the nature of sound and how it travels through the air before reaching the ear.

### 2.4.1. *Inter-aural Time Difference*

The Inter-aural time difference (ITD) is the subtle time delay between sounds arriving at the left and right ears. This is a primary cue in helping us determine the lateral position (or azimuth) of a sound source. The inter-aural time delay of a sound source directly in front or behind a person is zero since the distance between the sound source and both ears is equal. The frequency of a sound also has a bearing upon the inter-aural time difference.

### 2.4.2. *Pinna response*

The pinnae, or the external visible portion of the ears, also exert an effect on incoming sounds. Similar to a radar dish, the pinnae reflect and focus incoming sound waves upon the opening of the external auditory canal. Higher frequencies are affected by the pinnae so as to alter the perceived azimuth and elevation of the source of a sound. The effect exerted by the pinnae is highly dependent upon the actual direction of the sound source. Furthermore, pinnae response among individuals differs greatly.



*The so called cone of confusion highlights the ambiguity in localizing a sound*

Batteau<sup>3</sup> investigated the role of the pinnae (see figure 1) in sound localization and conducted experiments to determine the role of the pinnae in sound localization.

---

<sup>3</sup> Batteau, D. W. (1967). "The role of the pinna in human localization," *Proc. Royal Society London*, Vol. 168 (series B), pp. 158-180. The first work to report pinna echoes as an explanation for vertical (and also horizontal) localization

He placed microphones in casts of real pinnae which were collocated on a bar without a human head in between. The sounds picked up by these microphones were then played to participants (via high fidelity head phones) who tried to identify the direction of the sound's source. Whilst the artificial pinnae were in place, participants were able to make reasonably accurate judgements about the azimuth (direction in the horizontal plane) and the elevation. Without the pinnae, participants' judgements were quite erratic. It is interesting to note that when the artificial pinnae were in place, participants found the sound to be 'out in space' and not simply lateralized inside the head as is usually the case with headphones.

The pinnae modify the spectra of incoming sounds in a way which depends on the angle of incidence of the sounds relative to the head and in so doing provided a complex direction-dependant filter. In characterizing this filtering action, measurements can be taken of the spectrum at the sound source and of the spectrum as it reaches the ear. The ratio between these two measurements is known as the *head-related transfer function (HRTF)*. Head-related transfer functions can be used to describe the acoustic interactions that a sound wave has with the listener's body, head, pinnae and ear canals. The complexity of these interactions makes the HRTF at each ear strongly dependent on the direction of the sound.

#### **2.4.3.            *Head shadow***

Head shadow is the term which refers to a sound having to travel through or around the head in order to reach an ear. This effect can significantly reduce the volume of the sound arriving at the contra-lateral ear and can also exert a filtering effect, adjusting the frequency components of the perceived sound. The head shadow effect can often give rise to ambiguity of sound localization. The difference between the volume perceived at each ear is known as the inter-aural intensity difference (IID). The relevance of the IID together with the ITD for spatial hearing is known as the *duplex theory*.

Using only IID's and ITD's, ambiguity still exists in accurately localizing a sound.

*“With only ITD and IID, a person cannot judge whether an acoustic event is in front, above, behind, or below. This ambiguity of location at a given degree of lateralization has been called the "cone of confusion" (Woodworth 1954) (depicted in Figure 2). It is now commonly accepted that the seeming uncertainty of spatial location on the cone of confusion is disambiguated by the complex acoustic profiles of the HRTF's”- Gary Kendall 1995.*

#### **2.4.4.            *Shoulder echo***

This is a more subtle cue which can be used to supplement auditory information about the elevation of a sound source. Certain frequencies are reflected from the upper torso of the body. Generally, the result of this is additional echoes which the ears perceive as a time delay.

#### **2.4.5.            *Head Motion***

The movement of the head in determining a location of a sound source is of great importance to sound localization. With the head in a static position, the auditory system has trouble resolving some of the directional ambiguities which are still present even having taken into account other auditory cues. Moving the head through an arbitrary number of degrees, either by rotating or by tilting, provides additional information about the direction of a sound source and can help resolve the ambiguity. As the frequency of a given sound increases, more head movements tend to be required. This is due to the fact that higher frequencies tend to not curve around surrounding obstructing objects as much and are therefore harder to localize.

#### **2.4.6. Vision**

Our sense of sight serves as an important confirmatory cue with which we can identify the physical location of a sound source. It is this feedback mechanism by which we confirm the direction of a sound source and experiments have demonstrated that restricted a person's field of vision for a period of time has dramatic knock on effects for their sound localization ability.

In practice, a combination of all of these cues contributes to the ability to locate the source of a sound in three dimensions. It follows, therefore, that to accurately reproduce three dimensional sound as we may hear it in the real world, it is necessary to understand and replicate these aforementioned cues, a task which is far from trivial.

#### **2.4.7. Precision of Localization**

The theoretic accuracy with which sounds may be localized is largely dependant upon the direction and frequency of the incoming sound. Different age groups have differing sensitivities to audio frequencies but more importantly is the direction. Varying degrees of azimuth can be localized more precisely than differences in elevation owing principally to the lack of acoustic cues for elevation.

### **2.5. Synopsis of Relevant Papers**

What follows are extremely succinct synopses of two papers found, which have a direct relevance to the study in hand.

#### **2.5.1. "Head Motion and Latency Compensation on Localization of 3D Sound in Virtual Reality"**

*Authors: Jiann-Rong Wu, Cha-Dong Duh & Ming Ouhyoung*

This paper, published in 1997, describes a study with two main hypotheses, one of which is of particular relevance to this study and that is that “through the computer simulation of 3D sound, dynamic head movement can help in the localization of sound in space, as compared to fixed head position.”

The authors make full use of HRTF's in the implementation of their 3D sound system and aim not only to investigate whether head movements assist in sound localization but also, if it does, to what extent it assists. This was done by testing participants in two sessions; one where head movement was permitted and one where it was not.

There findings demonstrated an astounding 90% improvement in sound localization when head movements were permitted. With head movement allowed, an average inaccuracy of  $9.5^\circ$  was observed compared to an average of  $18^\circ$  where participants were denied head movements.

**2.5.2.        *“Localization with non-individualized virtual acoustic display cues”***

*Authors: Elizabeth M. Wenzel, Frederic L. Wightman & Doris J. Kistler*

This is a study, published in 1991, which assessed the ability of sixteen participants to localize sounds in terms of both azimuth and elevation. The sounds were projected from a random direction and the sounds were heard both over loud speakers and via headphones. In the case of the headphones, HRTF's were used to spatialize the sound.

Their findings indicate that spatialized sound can be used to great effect without the need to individually tailor the HRTF's to the participants. However, high error rates and reversals were observed among a few participants and this remains problematic for the authors. Although not completely understood, they assert that these confusions are a result of the static nature of the stimulus being presented and this allows the cone of confusion to have its ambiguous influence.

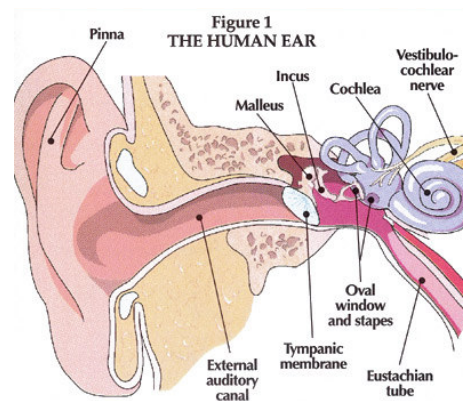
They go on to postulate that front / back reversals may be down to the role of the visual sense; given an ambiguously projected sound and in the absence of any visual confirmation as to the source of the noise, the authors suggest that the perceptual system may apply a heuristic which assumes the sound source is located to the rear since it a likely source cannot be seen to the front. They suggest, amongst other things, that a dynamic cue which correlates with head movement – the very essence of this study – might help resolve this problematic issue.

### 3. Project Proposal

#### 3.1. What?

The development and implementation of a real-time 3D audio spatialization mechanism may enhance the overall participant experience, making for a more compelling and immersive encounter.

Based on general work in the area and specific research currently on-going at The University of Bristol, I intend to introduce directionalized audio into the existing application framework by integrating a head mounted compass. By emulating the way in which noise normally reaches our ear, sound could be delivered to users in such a way as to create the illusion of a *virtual sound source* whose perceived position remains static. For example, when facing towards the source of a sound, roughly equal sounds levels reach each ear. Having turned clockwise through 90 degrees, more sound is picked up by the left ear than by the right. This is a dramatic over simplification of the way in which we locate the direction and distance of a sound source but it serves to highlight the fact that using only traditional headphones, such a mechanism is simply not feasible.



#### 3.2. Why?

The motivation for implementing such a system stems from the ideology of being able to supplement our sense of hearing with information relevant to our geographic location / orientation in real-time. Provision of such a system would permit investigation into the possibility of using spatialized sound reproduction either as a navigational aid or simply as a means of enhancing any kind of digital experience by adding to the perceived realism.

Sound has not been as important as graphics in computer game development. Game developers tend to spend a great deal more time and effort on new features and effects for three dimensional graphics and it has been a struggle to persuade people to spend time and money on high-quality sound in games. Similarly, most end-users would prioritize a new 3D graphics accelerator over a new sound card.

However, the situation is now changing - both end-users and games developers are now paying more attention to sound. Audio chip makers and 3D sound developers have done their best to convince users and application developers that good 3D sound is an integral part of a modern computer. Recent developments, such as integrated hardware-based 3D sound processing hardware have made it easier to include high quality 3D sound in new developments. Such a system provides programmers with a simple application programming interface (API) with which they may specify the location, orientation and speed of sound emitting objects and the rest of the processing is undertaken by the inbuilt hardware which computes the signal to be sent to the left and right sound channels.

The motivation for implementing 3D sound systems is gain is clearly driven by the increasing amount of revenue returned by the sales of millions of copies of games each year but applications of equal, if not greater importance emerge as the technology matures. AudioDoom<sup>4</sup> is the name of a take of the classic computer game Doom which has been adapted especially for visually impaired children. The study looked at the utility of a system in assisting the generation of mental maps by exposing the children to a virtual acoustic environment before inviting them to recreate their mental model using Lego bricks.

---

<sup>4</sup> <http://www.dcc.uchile.cl/~jsanchez/Pages/papers/virtualenviorment.pdf>

The results of the prototypical system point to the fact that virtual acoustic environments can serve not only to entertain and amuse but also as a means of delivering educational materials. It is widely accepted fact that blind children require some level of assistance in building up a mental model of their school, local neighbourhoods and places in the locality. This new technology<sup>5</sup> provides us with an opportunity to support the blind in carrying out these tasks by going beyond models of the fantastic and imaginary and replacing them with actual virtual representations of real world places.

Sound has many other potential applications including in the domain of collaborative virtual environments. As discussed, directionalized sound could help to increase the sense of presence or immersion in virtual environments by simulating what would be heard in the real world. It could conceivably be used to relay information about the surrounding environment and the objects within it. Such a mechanism could be invaluable in supplementing the users' sense of orientation in virtual environments.

Sound can also be used as a substitute for other forms of sensory feedback. Consider the case pressing a soft-button (i.e. not an actual physical button, but an onscreen button) on a touch screen computer. Due to the lack of haptic feedback, users can find it hard to know when the button has been successfully depressed. Sound cues, especially accurately spatialized ones could be used to help alleviate this problem and others like it by projecting an appropriate confirmatory sound from the location of the button.

An excellent example of sound being used to supplement or replace typical visual cues is *Mercator*. This project is an attempt to provide spatialized sound as an alternative non-visual interface to the X Window system, primarily for use by the visually impaired. The project aims to translate the behaviour of window-based application into a virtual auditory space. In such a way, audio data is used to convey information about the layout and organization of objects on the user's screen.

Spatialized sound is an exciting new means of implementing a *generic* non-visual interface and is currently an active area of new research. The advantages of auditory

---

<sup>5</sup> <http://www.c5.cl/blind/>

interfaces are many and varied; of primary interest is the ability to monitor and identify informational sources from all directions whilst leaving the primary input sense (vision) free. Equally attractive is the ability of 3D sound to support spatial segregation of a multitude of simultaneous noises in a way which naturally maximises positive transfer from the real world. In short, 3D sound has the ability to reinforce visual information and be used to provide a far superior sense of realism and immersion.

### **3.3. How?**

The human auditory system is capable of localizing sounds in three dimensional space, comprising lateral resolution of a sound's direction, distance resolution and also elevation resolution. The predominant cue for determining a sound's source is the difference of intensity of the signals arriving at the left and the right ears (known as the interaural intensity difference or IID). Secondary to this is the perceived delay between the sounds arriving at each ear (known as the interaural time delay or ITD). ITD is also the predominant cue for localizing wavelengths of less than 1500Hz (or ~23cm) since the head acts as a barrier for such frequencies. In such cases, sounds waves propagate around to the contra-lateral ear (the ear opposite the sound's source).

Batteau<sup>6</sup> investigated the role of the pinnae (see figure 1) in sound localization and conducted experiments to determine the role of the pinnae in sound localization. He placed microphones in casts of real pinnae which were collocated on a bar without a human head in between. The sounds picked up by these microphones were then played to participants (via high fidelity head phones) who tried to identify the direction of the sound's source. Whilst the artificial pinnae were in place, participants were able to make reasonably accurate judgements about the azimuth (direction in the horizontal plane) and the elevation. Without the pinnae, participants' judgements were quite erratic. It is interesting to note that when the artificial pinnae were in place, participants

---

<sup>6</sup> Batteau, D. W. (1967). "The role of the pinna in human localization," *Proc. Royal Society London*, Vol. 168 (series B), pp. 158-180. The first work to report pinna echoes as an explanation for vertical (and also horizontal) localization

found the sound to be ‘out in space’ and not simply lateralized inside the head as is usually the case with headphones.

The pinnae modify the spectra of incoming sounds in a way which depends on the angle of incidence of the sounds relative to the head and in so doing provided a complex direction-dependant filter. In characterizing this filtering action, measurements can be taken of the spectrum at the sound source and of the spectrum as it reaches the ear. The ratio between these two measurements is known as the *head-related transfer function (HRTF)*. Head-related transfer functions can be used to describe the acoustic interactions that a sound wave has with the listener’s body, head, pinnae and ear canals. The complexity of these interactions makes the HRTF at each ear strongly dependent on the direction of the sound.

Using only IID’s and ITD’s ambiguity still exists in accurately localizing a sound.

*“With only ITD and IID, a person cannot judge whether an acoustic event is in front, above, behind, or below. This ambiguity of location at a given degree of lateralization has been called the "cone of confusion" (Woodworth 1954) (depicted in Figure 2). It is now commonly accepted that the seeming uncertainty of spatial location on the cone of confusion is disambiguated by the complex acoustic profiles of the HRTF’s” - Gary Kendall 1995.*

Since pinnae differ in shape and size between different individuals, HRTF’s also differ across individuals. Wenzel *et al*<sup>7</sup> investigated the effect of using non-individualized HRTF’s to address the question of whether participants could make use of information provided by another person’s pinnae. Their results suggest that localization in the horizontal plane was not significantly effected but that resolution of the cone of confusion is affected and this resulted in some participants confusing front with back and up with down.

---

<sup>7</sup> Localization Using Nonindividualized Head-Related Transfer Functions - Wenzel, Elizabeth M., Arruda, Marianne, Kistler, Doris J., and Wightman, Frederic L. JASA Vol. 94, No. 1 (July 1993)

Both humans and animals have evolved to greatly depend upon their ability to localize the source of a given sound. Acoustic data supplements the constant stream of visual data received by the eye, serving as a cue to determine the direction of objects to be avoided or sought out.

‘Localization’ refers to judgements regarding both the direction and the distance of a sound source. When wearing stereo headphones, the sounds are sometimes perceived to come from within the head. Akin to localization is the term ‘lateralization’ which is used to describe the apparent direction of the sound within the head.

### **3.4. Resources Available**

#### **3.4.1. Hardware**

The development environment in use throughout the project will be based on a Toshiba laptop with a 1500MHz Intel Celeron processor and 256MB of RAM. The target hardware is to be the HP iPAQ 5450 which, despite having a 400MHz ARM processor, actually runs at speeds of



closer to 200MHz. A wireless network (802.11b) connects the two machines.

The compass to be used is described in further detail in [section](#).

#### **3.4.2. Software**

The software will be written and compiled under Microsoft Embedded Visual C (EVC) under the Microsoft Windows XP Professional operating system. This is a derivative of the more familiar Visual Studio development environment. Whilst the OS should provided some stability throughout development, EVC is known to be highly unstable and could potentially present difficulties during development.

The iPAQ utilises Microsoft's Pocket PC 2002 and this target operating system for compilation. The test application is to be written in C and C++.

Cool Edit 2 Professional is the sound software used to tailor to the samples to the correct sample rate.

### **3.5. Research Question**

As has been stated, the investigation will focus upon the extent to which this sound system can support audio navigation. Specifically, we are interested in how accurately (i.e. to within how many degrees) a person can localize a sound using, as a primary cue, movements of the head.

This is of interest since head movements offer one of the most important acoustic cues used in localizing a sound. It is an absolutely typical response for people to turn their head towards an unexpected sound. If there is still perceived ambiguity in the sound's source, it is not unusual for a person to continue moving their head around until the sound has been localized to their satisfaction. This technique is carried out almost subconsciously and is useful in helping since when the head is turning, the brain receives additional information defining the sound's position in space. This is the exact process that a person undergoes when, having bought a brand new car, they are trying to locate the source of that irritating rattle.

For this task, headphones help to make the task somewhat easier as it solves the problem of delivering one signal to one ear and another signal to another ear. The use of headphones instead of a pair of speakers also removes the presence of so-called sweet spots. These are where sound waves from the two speakers meet and constructively interfere with one another. In one of these sweet spots, the 3D effects become audibly more perceptible whereas in locations where deconstructive interference occurs, it is tangibly harder to perceive the sound source's correct direction.

## 4. Implementation

### 4.1. Implementation Choice

There are innumerable means of spatializing sound in either two or three dimensions. Some of the potentials to be considered will now be outlined.

#### 4.1.1. *Bryden 2D*

Karen Bryden's 2D sound spatialization library works extremely well. Taking as an input a 16-bit 11025 samples/second sound, the software uses HRTF's to spatialize the sound anywhere in the lateral plane whilst taking into account room size as well as first order reflections from each of the room's four walls. This software relies heavily upon the use of floating point arithmetic and hence porting it to the Pocket PC 2002 platform would require extensive and time consuming code refactoring which would fall beyond the scope of this project.

#### 4.1.2. *Creative EAX*

EAX is a API developed by Creative Labs designed to create environmental audio (in conjunction with Microsoft's Direct Sound) in games and other applications. It encompasses obstruction and occlusion effects which describe the acoustic response when an audio stimulus is either blocked from the listener's direct 'line of sight' or when the audio stimulus' echoes are blocked from the listener's direct 'line of sight'.

Sadly, in order to take advantage of these reputedly impressive features, the machine's sound card must feature explicit support for EAX functionality, sadly something the iPAQ does not currently do.

#### 4.1.3. *Aureal A3D*

The groundbreaking Aureal A3D is another positional sound system which comes in two parts; the set of API for programmers to use and the unique

algorithms which spatialize the sound. Whilst the game or application utilises the API to specify whereabouts a particular sound will be placed, the sound card's drivers and algorithms are responsible for actually making it appear as if it were coming from a particular direction. As effective as it is, it again sadly requires specific sound card support.

#### **4.1.4. FMOD**

FMOD is a programming API which support MIDI, DirectSound and Creative EAX and is available for all major platforms including Linux and Macintosh. It is a powerful and fast audio engine which is used widely by games programmers across the industry. Furthermore, FMOD is the sound engine currently integrated with the Mobile Bristol framework. It is understood that FMOD includes EAX support so long as the sound card also has provision for it.

#### **4.1.5. KOAN**

KOAN is another software API which interfaces with inbuilt hardware to provide a range of audio effects including spatialization. Although seemingly very effective and happily targeted to the Pocket PC platform, it lends itself better to adding DSP effects to music and sound effects rather than being able to support accurate real-time sound localization.

The implementation choice hinged upon speed of sound processing and compatibility with the iPAQ operating system, Pocket PC 2002. Due to the inability of the target hardware to undertake floating point calculations with suitable speed, and other limitations, it has been necessary to use the FMOD system to control the sound.

## **4.2. The Development Phase**

What follows is an informal account of the path taken through the development phase.

*The initial couple of weeks were spent familiarising myself with the FMOD API.*

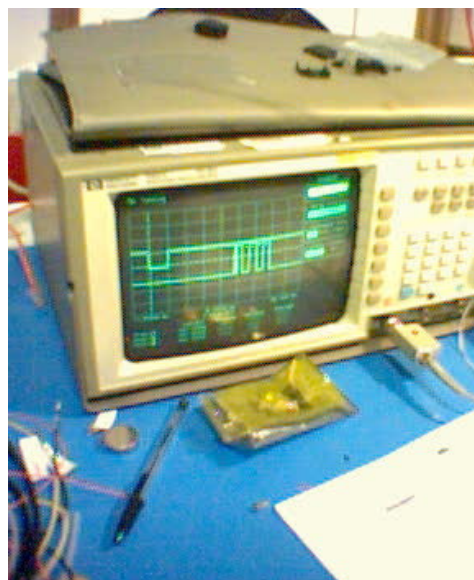


*Having a fair amount of programming experience this was a fairly straight forward exercise. The API functionality extends to allocating a particular sample to one of sixteen sound channels, specifying the listener position as well as the positions of sound sources in three dimensional co-ordinates.*

*The subsequent decision to attempt to integrate the Bryden 2D software with the FMOD framework was a poor one. The lack of floating point arithmetic support on the iPAQ meant that actually running the software as intended caused the machine to virtually grind to a halt as it struggled to undertake the required computations at the necessary rate.*

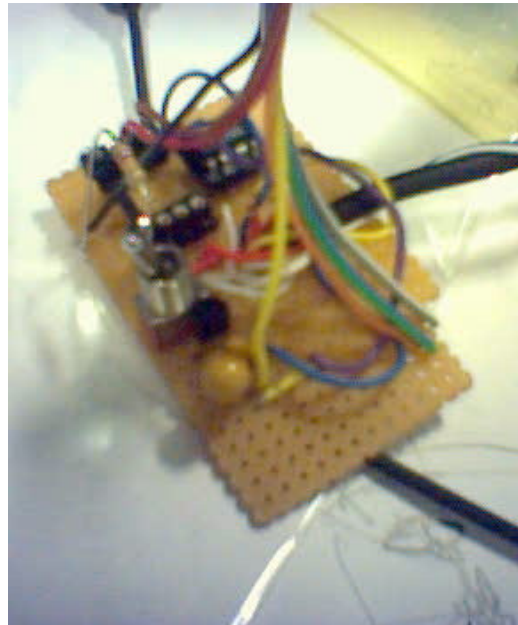
*Knowing that the inclusion of hardware into the project was likely to be the most time-consuming and troublesome, I made this the next step. Having obtained a Vector 2X compass, on loan from Bristol University, I went about soliciting a suitable person to help wire it up to the iPAQ. The kindly supplied compass was a two axis magnetometer one with a ten channel wire trailing out of it. At the other end was no plug or adaptor so it would have to be custom wired.*

*The Vector compass is known to be very sensitive to tilting; a few degrees yaw or pitch is sufficient to confound the compass' reported bearing by up to 180 degrees. With this in mind I sought an alternative compass, more resilient to tilting as this would have been entirely more appropriate to mount on a person's head.*



*By chance, I managed to source an alternative compass produced by Honeywell. Being a three-axis compass made it an attractive prospect as inaccuracies as a result of tilting would not be an issue. And so began the arduous process of coaxing the compass into talking to the iPAQ. The first challenge was to find a suitable means of powering the compass. A voltage regulator and 3.5v button battery proved sufficient for this task.*

*The specifications provided with the Honeywell compass as well as being lacking in details were also woefully inaccurate in the details they did provide. This led to further difficulties in getting the two devices to communicate. Although the Honeywell compass purported to be RS232 (a standard serial communication standard), spending two weeks investigating its refusal to return a compass reading of any kind led us to the conclusion that it was not in fact RS232 as claimed by the manufacturers. This was confirmed by oscilloscope readings which revealed that it was sending an inverted signal back to the iPAQ.*



*Having overcome all of these difficulties and innumerable others, much to my dismay the compass failed to provide any kind of heading data even when it really ought to have. We resolved to placing the Honeywell compass in the bin and reverting to the very tilt sensitive Vector compass.*

*This unfortunate set of circumstances would have serious knock on effects since the sensitivity of the compass would no doubt affect the iPAQ's ability to spatialize sound as intended. Yet more soldering and tinkering was required to find an alternative suitable power supply for the Vector compass (eventually using a 9v alkaline battery) as well as completely changing the circuit board layout to be compatible.*

*The original intention of this project was to integrate the existing locational data from the ultrasound sensors with the directional data from the new compass. Due to circumstance beyond my control however, this was not possible. Although the iPAQ has only one serial port into which sensors can be plugged, at the time of commencing the project it was my belief that an adaptor bus would have been developed allowing multiple sensors to be simultaneously plugged in. This was not the case but the sensor bus is still in progress and would provide an interesting extension to this project.*

## 5. Methodology

It is now necessary to define exactly how the experimental session will be conducted. The aim of the study will be to gather quantitative data regarding the accuracy of participants' ability to gauge the direction of a sound source.

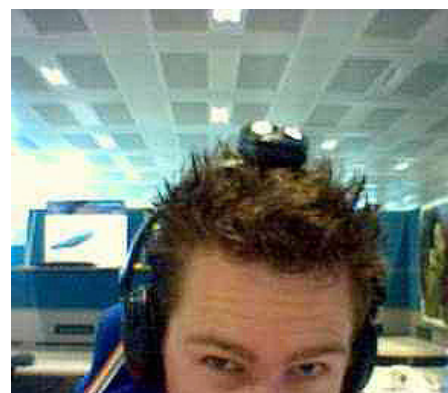
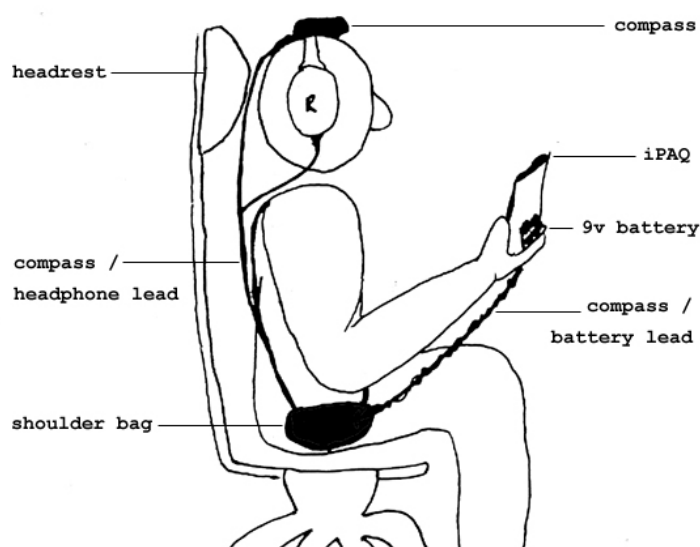
### 5.1. Participants

Ten participants were recruited for the experiment on a largely ad-hoc basis. No monetary incentive was offered to them. The participants all have a background in computers and are also accustomed to partaking in experiments of a technological nature. All of the participants are directly or indirectly Hewlett Packard employees.

All participants are educated to at least a graduate level and fall into the 25-45 age bracket, with no reported hearing problems.

### 5.2. Apparatus

Participants were to use a HP iPAQ 5450 handheld computer with 802.11b wireless networking connectivity used for retrieve test session data. Worn over the shoulder in a small bag was the circuit board interfacing between the compass and the iPAQ.








The head-mounted compass was fitted to the top headphone strap using a small custom built plastic mould produced by a 3D printer. Once in place on a participants head, the emerging wires were arranged so as to run down behind the head and under one arm into a black, shoulder-mounted bag. Re-emerging from the bag was the wire which connected to the serial port of the iPAQ. The 9v battery was to be held by the participants during the experimental session.

Participants were seated on a sturdy swivel chair with a comfortable headrest to support their head throughout the course of the session.

Centred directly underneath the swivel chair was a large print out of a compass measuring approximately 2 x 2 metres. This compass has 1 degree graduations from 0 to 360 degrees.

The five sounds were chosen so as to offer a range of varied timbres and frequencies. All were the same format; monophonic PCM WAV files at a sample rate of 11025 samples/second. All five sounds were set to loop indefinitely. The waveform depicted for each sample is an analysis of the frequency makeup with frequency running along the  $x$  axis and dB against the  $y$  axis.

Sample #	Description	Length (s)	Waveform
1	Helicopter noise	1.4	
2	Low Synth String	9	
3	The Flintstones theme	35	

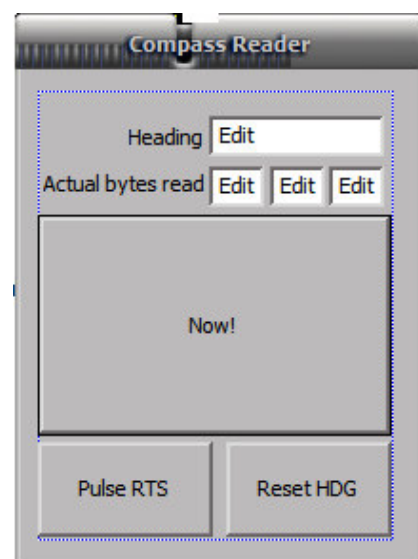
4	Andy C drums	11	
5	Homer Simpson's 32 'Doh's	14	

The headphones in use were Sony studio monitor headphones with an over-head strap (to which the compass was affixed)

### 5.3. Procedures

Upon commencement of an experimental session, each participant had the task explained to them. They were told that they would hear five different sounds, each of which would be projected from a randomly allocated direction. Elevation was not being considered; participants were instructed that their ability only to judge sound azimuth (i.e. in the lateral plane) was being assessed. With permission, a photograph was taken of the participant in situ.

Their aim was to rotate their swivel chair, whilst keeping their head on the chair's head-rest, until they thought the sound was in front of them i.e. the sound was balanced equally in the left and the right ear. When they were happy that they were facing the sound as accurately as they could, they were instructed to press the large and "Now!" button which occupies the majority of the iPAQ's screen. Upon depressing this button the sound would change to the next in the sequence of five. The depression of this button would also trigger the software to record, to a text file, the randomly allocated direction from which the sound was being projected as well as the



actual direction as read from the head mounted compass. Simultaneously the experimental supervisor would note down the real world bearing as indicated by the large compass printout below the swivel chair. This would also reveal the accuracy of the head mounted compass.

Participants were reminded that there were no time constraints since all five sounds were set to loop indefinitely but at the same time a session would need to last no longer than ten or fifteen minutes in total.

At this stage, participants were invited to don the headphones and head mounted compass. They were helped into the equipment and invited to adjust the chair's height and tilt settings and to seat themselves so as to be as comfortable as possible. The compass was confirmed to be level before commencement. When the participant indicated that they were ready, the experimental supervisor started the test program on the iPAQ by selecting 'USTEST' from the start menu at which point the participants started hearing the directionalized sound. Since the software was written in such a way as to automatically calibrate itself to zero degrees when the program started, it was necessary to align the participant's chair with zero degrees on the large print out compass below the swivel chair.

## 6. Results

Taking into account all of the above, it is now high time to actually present some results. Having accrued a selection of text files following each test session, their contents were transcribed into a spreadsheet for analysis.

### 6.1. Pilot Study

Prior to commencing the formal experimental sessions, two pilot study sessions were informally conducted to ensure that the software and hardware were performing as required to iron out any other problems that may affect the overall outcome.

The results of these two initial sessions are hereby presented. The column headed *actual azimuth* contains the randomly generated angles from which the sound was projected by the iPAQ.

The *compass reading* indicates what heading the compass returned at the time the participant indicated they had centred the sound in front of them.

The *real world bearing* was the reading made at the same time by the experimental supervisor and is derived by reading off the bearing indicated by the large printed compass beneath the participant. This figure is accurate only to the nearest five degrees due to the difficulties in manually interpreting the person's heading with any great degree of precision.

*Compass inaccuracy* data was computed by taking the difference between the heading returned by the digital compass and the real world bearing as noted by the experimental supervisor.

*Perceptual inaccuracy* data was obtained by computing the difference between the actual azimuth and the compass reading. Where this difference was found to be greater than  $180^\circ$ , the complement of this angle was taken i.e. the difference was the acute angle between the two headings rather than the obtuse.

Participant Name	Actual Azimuth (°)	Compass Reading (°)	Real		Mean Compass Inaccuracy (°)	Perceptual Inaccuracy(Δ)	Mean perceptual inaccuracy(ΣΔ/n)
			World Bearing (°)	Approximate Compass Inaccuracy (°)			
Greg Bullock	285	291	310	19		6	
	254	240	270	30		14	
	216	208	240	32		8	
	296	304	325	21		8	
	48	78	90	12	<b>22.8</b>	30	<b>13.2</b>
Howard Stredwick	71	37	45	8		34	
	0	22	35	13		22	
	202	20	5	15		178	
	69	49	50	1		20	
	291	333	355	22	<b>11.8</b>	42	<b>59.2</b>

## 6.2. Observations & Amendments

The first problem arising from the pilot study (although not at all apparent in this initial set of data) was to do with the allegedly randomly generated azimuths from which direction the sound was being projected. A software bug meant that the bearings were not being generated in the range [0...359] degrees and furthermore, the same numbers were being generated for each test session. This problem emerged as being a problem with the pseudo random number generator (PRNG). Despite seeding the PRNG with a number returned by the iPAQ system clock's milliseconds count, the sequence being returned was not at all random. After much puzzlement, it was discovered that this was a result of the fact that the iPAQ seems not to support the milliseconds count in the same way that other platforms do so seeding the PRNG with a tick count did indeed provided the solution.

Equally startling and also clearly apparent in these results is the marked inconsistency of the compass. Although the real world bearings are not exactly accurate, it was expected that the compass would return heading significantly closer to the real world bearing than it was doing. In an attempt to address this, any large metal objects such as computer monitors and whiteboards were removed from the vicinity in case these were

affecting the magnetometers inside the digital compass. Slightly reassuringly however, a greater compass inaccuracy does not seem to affect the sound lateralization. This is because the sound is being projected with reference to the bearing returned by the (seemingly inaccurate and jumpy) compass. Under laboratory conditions this has no real ill-effects but should this be outside being used as a navigation aid, this would present a significant problem.

A further problem at this stage was down the compass battery; having checked its charge after these reading were taken revealed that it was nearly dead. It is not clear what, if any, effect this had upon these results.

Looking now at the perceptual inaccuracy results, it is not reassuring to see that one participant achieved only a mean score of as much as  $59^\circ$ . Were this system to be implemented with this sort of inaccuracy still present, it would be near useless as an audio navigation aid. However upon closer inspection, we can see that one perceptual inaccuracy reading is greater than  $180^\circ$ . This would suggest that the participant failed to disambiguate the front-back confusion problem. This one error is pushing the mean up by a great deal. Had it not been for the front-back mix up, the inaccuracy would of course have been a mere  $2^\circ$  and this would have given a participant mean of  $24^\circ$  in place of  $59^\circ$ . It is hoped that this is a problem that will not manifest itself in the main study.

It is worth noting at this point that one of the participants in the pilot study was in fact the person who wrote the software and devised the experiment. This is not an ideal situation since it is expected that the person who has a deep understanding how the system works will achieve a much better score. This may be as a result of practice or because a specific strategy has been devised to help lateralization. In the case of a pilot study though, this is not too grievous an oversight.

### **6.3. Presentation of Results**

The results from the ten main tests are presented in the manner in which they were collected, by participant and with each test in order i.e. for each participant, there are

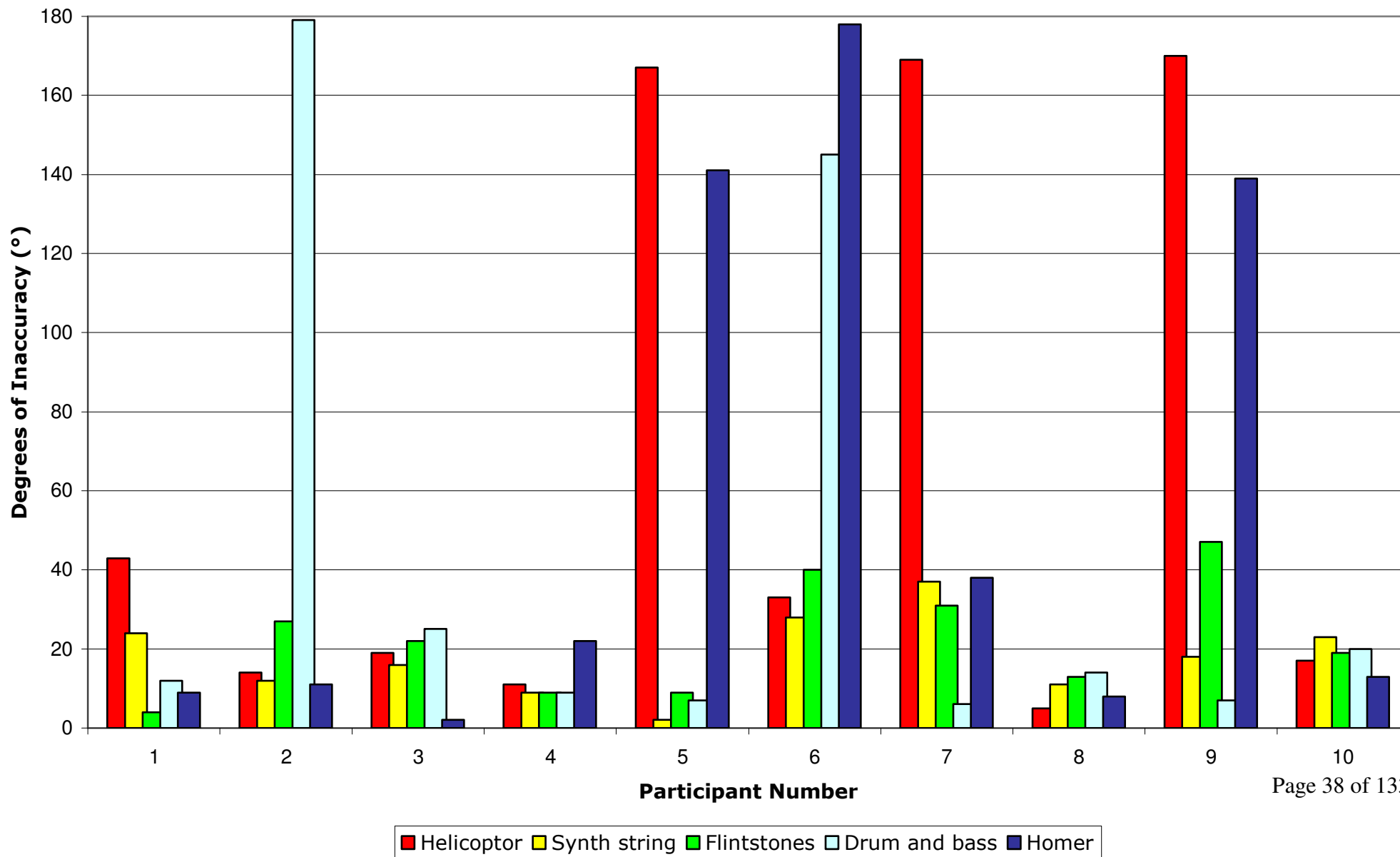
five rows of data corresponding to the five different sounds in the order they were presented to the participant.

The data has been analysed using a spreadsheet for convenience as depicted below. In presenting the data graphically, it is the intention to display the data so as to preserve the individual performance of different participants. It will also be of use to present a frequency graph. See Appendix B for the full set of results.

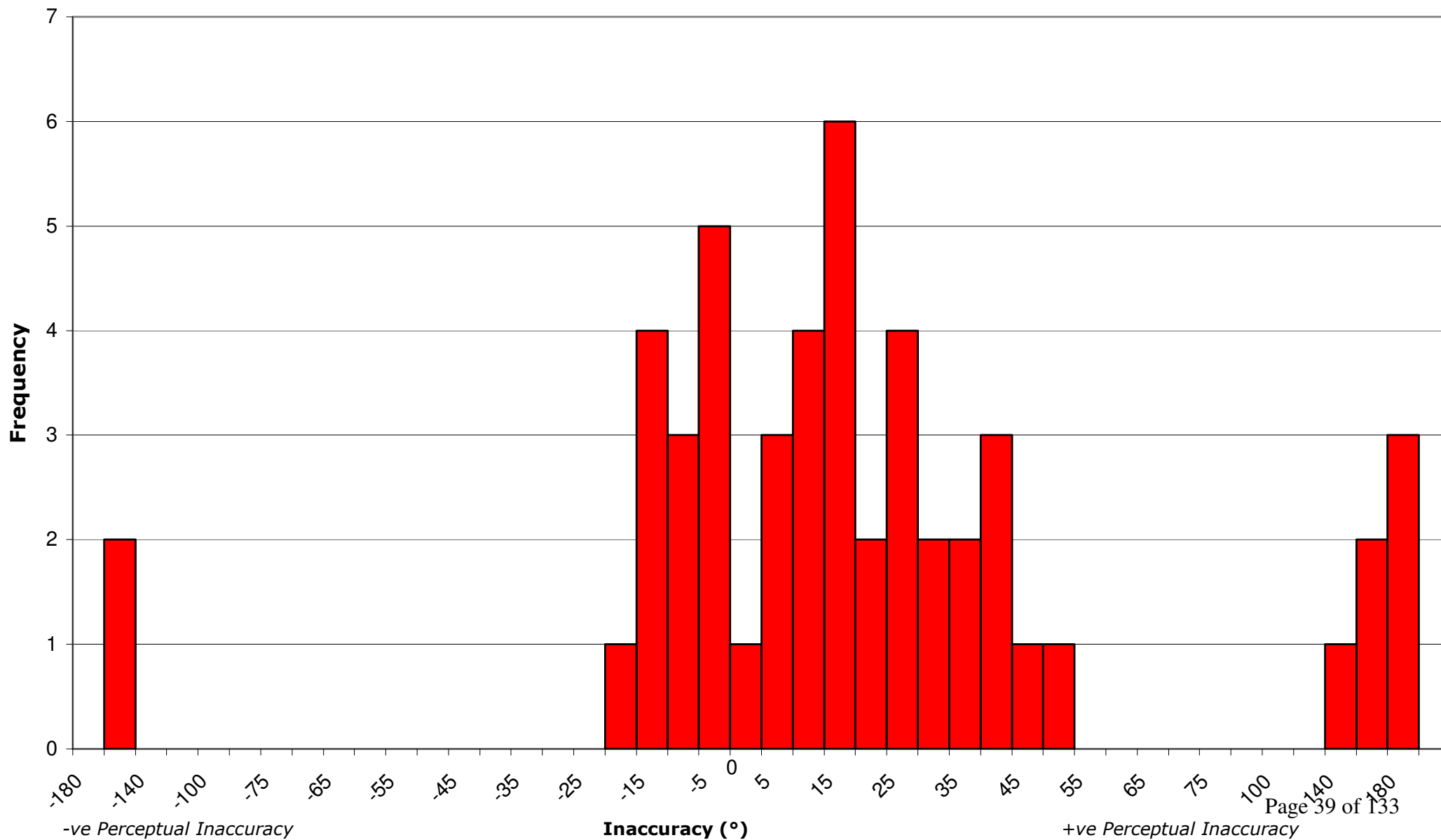
What makes interesting reading are any pertinent comments made by the participant. These are again found in Appendix B and are detailed below the participant's associated results.

Participant #	Picture	Experimental Instructions	Participant Name	Actual Azimuth (°)	Compass Reading (°)	Real World Bearing (°)	Approximate Compass Inaccuracy (°)	Mean Compass Inaccuracy (°)	Perceptive Inaccuracy (Δ)	Mean perceptive inaccuracy(ΣΔ)	Perceptive Inaccuracy MOD 180	Mean Perceptive Inaccuracy MOD 180	
26		1	Rachel Yeardeley	76	87	130	43		11		11		
27		1	Rachel Yeardeley	116	107	160	53		9		9		
28		1	Rachel Yeardeley	55	64	75	11		9		9		
29		1	Rachel Yeardeley	270	279	280	1		9		9		
30		1	Rachel Yeardeley	344	6	0	6	22.8	22	12	22	12	
31		1	Gary Porter	67	234	250	16		167		13	67	
32		1	Gary Porter	124	126	145	19		2		2		
33		1	Gary Porter	55	46	35	11		9		9		
34		1	Gary Porter	84	77	100	23		7		7		
35		1	Gary Porter	340	121	165	44	22.6	219	80.8	39	14	340
36		1	Sophie Bevan	308	341	340	1		33		33		
37		1	Sophie Bevan	264	292	315	23		28		28		
38		1	Sophie Bevan	332	12	40	28		40		40		
39		1	Sophie Bevan	70	215	260	45		145		35	70	
40		1	Sophie Bevan	238	56	100	44	28.2	162	85.6	2	27.6	238
41		1	Jo Reid	35	204	265	61		169		11	35	
42		1	Jo Reid	233	270	300	30		37		37		
43		1	Jo Reid	3	34	40	6		31		31		
44		1	Jo Reid	163	169	220	51		6		6		
45		1	Jo Reid	6	44	50	6	30.8	38	56.2	38	24.6	
46		1	Tom Miner	348	353	355	2		5		5		
47		1	Tom Miner	118	129	165	36		11		11		
48		1	Tom Miner	238	251	245	6		13		13		
49		1	Tom Miner	10	24	25	1		14		14		
50		1	Tom Miner	246	238	230	8	10.6	8	10.2	8	10.2	
51		1	Miranda Mowbray	187	357	350	7		170		10	187	
52		1	Miranda Mowbray	131	113	70	43		18		18		
53		1	Miranda Mowbray	97	144	150	6		47		47		
54		1	Miranda Mowbray	57	64	45	19		7		7		
55		1	Miranda Mowbray	223	2	0	2	15.4	221	92.6	41	24.6	223
56		1	Jennifer Newman	73	56	50	6		17		17		
57		1	Jennifer Newman	315	292	310	18		23		23		
58		1	Jennifer Newman	246	227	240	13		19		19		
59		1	Jennifer Newman	29	49	45	4		20		20		
60		1	Jennifer Newman	101	88	100	12	10.6	13	18.4	13	18.4	
61			Greg Bullock	285	291	310	19		6		6		
62			Greg Bullock	254	240	270	30		14		14		
63			Greg Bullock	216	208	240	32		8		8		
64			Greg Bullock	296	304	325	21		8		8		
65			Greg Bullock	48	78	90	12	22.8	30	13.2	30	13.2	
66			Howard Stredwick	71	37	45	8		34		34		
67			Howard Stredwick	0	22	35	13		22		22		
68			Howard Stredwick	202	20	5	15		182		2	202	
69			Howard Stredwick	69	49	50	1		20		20		

### 6.4. Chart to Show Perceptual Inaccuracies of Ten Participants Lateralizing Five Different Sounds



### 6.5. Chart to Show Distribution of Perceptual Inaccuracy Measurements



## 6.6. Summarised Results

The results obtained are spread very widely from some participants having a very low average error score (from as little as 10.2°) to others having as much of an average error as 85°.

Figure 123123 summarises the findings.

<b>Minimum perceptual inaccuracy</b>	2°
<b>Maximum perceptual inaccuracy</b>	179°
<b>Mean perceptual inaccuracy</b>	40.7°
<b>Standard deviation</b>	54.5°

Participant comments following the session would seem to indicate that the system was received well. The choice of the five different sounds was commented upon with some glee.

All participants seemed to enjoy the effect of having the sound respond to their movements. Some clearly found it harder than others and some reported the need to concentrate extremely hard.

One or two participants admitted having no idea at all about the direction from which some of the sounds were being projected.

Some participants went on to explain glitches in the sound which they had experienced and also to detail the manner in which the sound seemed to occasionally and inexplicably jump from one ear to the other. When this was reported to have happened, it was said to be extremely off-putting and very annoying.

<b>Minimum compass inaccuracy</b>	0°
<b>Maximum compass inaccuracy</b>	61°
<b>Mean compass inaccuracy</b>	18.7°
<b>Standard deviation</b>	19°

The results also serve as testament to the poor performance of the Vector head mounted compass. With an overall mean inaccuracy of  $18.6^\circ$  and a maximum inaccuracy of  $61^\circ$ , one just might be able to find one's way home with it although it would probably take a very long time.

## 6.7. Discussion

The poor performance of the compass has not directly affected the results since the sound is being projected relative to the bearing it returned. If, however, one used this compass in conjunction with some form of location sensor to home in upon a target in the real world, it would seem that it would not be the best choice. The compass inaccuracy may have affected the results by its non-linearity and jumpiness. Almost without doubt, it would have been the compass responsible for any sound glitches such as sound jumping jerkily from one ear to the other.

The distribution chart 6.5 mirrors the information illustrated by 6.4, the only difference being that 6.5 sacrifices participant performances in favour of positive versus negative error. This does demonstrate that the error was spread fairly evenly in both a clockwise and an anti-clockwise direction.

The standard deviation of 54.5 is extremely high compared to the mean of 40.7. This indicates a fairly erratic spread of data. From figure 6.5, it is clear to see that the distribution of data appears to be bi-modal with all but one of the fifty perceptual inaccuracies falling either below  $40^\circ$  or above  $140^\circ$ .

Let us consider for a moment the reason for this. If there were a perceptual inaccuracy reading in this  $40^\circ$ - $140^\circ$  region then it would mean that the reading would have been arrived at by the participant hitting signalling readiness when the sound was predominantly being heard by just one ear. So it is not therefore surprising that there are no cases such as this since it is extremely straight forward to determine that only one ear is receiving any sound. It is consequently fair to infer that, for the eight readings which are ostensibly outliers, the participants were unable to distinguish back from front. In

spite of the scarcity of such situations, it is still a surprising find since any head movement should immediately highlight to the participant which direction is which.

Although in a static situation the sound would seem to be the same whether being projected from either the back or the front, a head movement should serve to disambiguate this unnatural phenomenon. When the sound is to the front of the listener, a head movement to the left would result in more sound being heard by the right ear than the left and vice versa. Conversely, when the sound is being projected from the rear, a head movement to the left would this time result in more sound being picked up by the left ear.

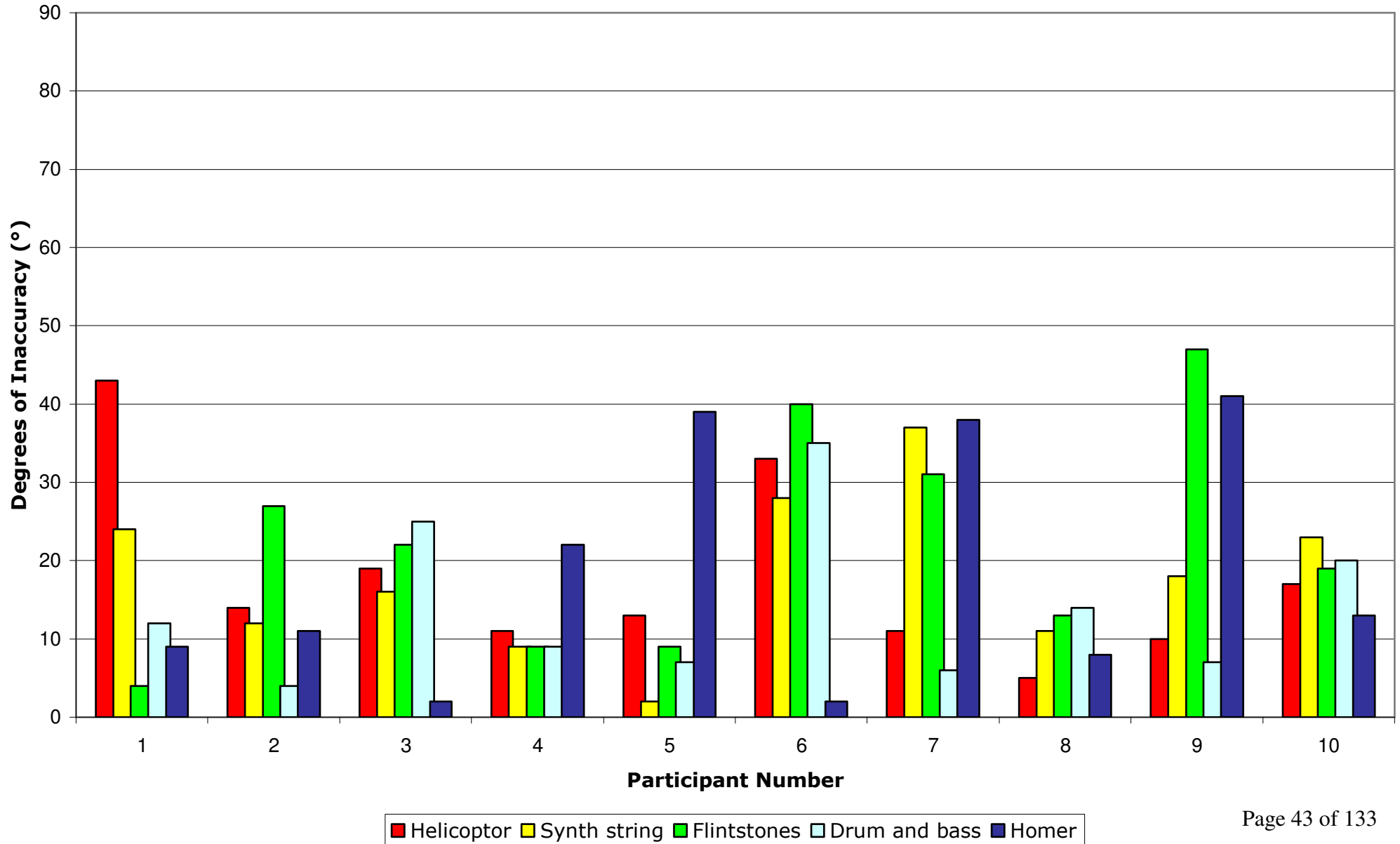
### 6.8. Revised Presentation of Data

Because of this interesting front / back ambiguity which still exists in these few cases, it would be interest to present the data 'mod 180'. So for those eight outliers, let us compute how close they are to 180° and treat them as if they were the same distance from 0°. So for a reading perceptual inaccuracy of 169°, we will convert this to 11.

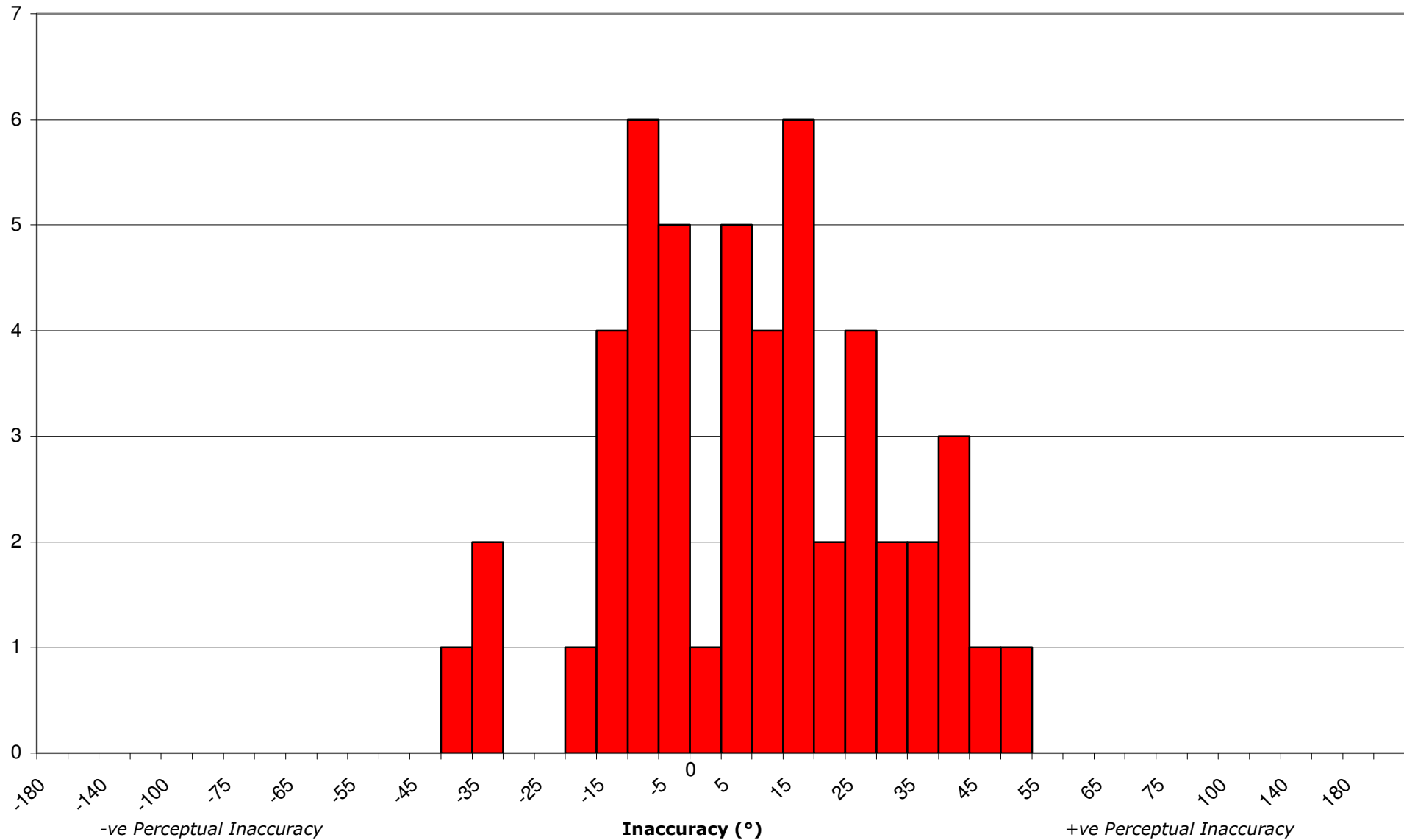
Doing exactly this yields some very different statistics. The mean is now found to be 18° (formerly 40.7°) and the standard deviation becomes 12.3° (formerly 54.5°). Comparing figures 6.4 and 6.9 (noting of course that the scale in use differs by a factor of two) demonstrates the marked difference that this has.

<b>Minimum revised perceptual inaccuracy</b>	2°
<b>Maximum revised perceptual inaccuracy</b>	47°
<b>Mean revised perceptual inaccuracy</b>	18°
<b>Standard deviation</b>	12.3°

### 6.9. Chart to Show Revised of Ten Participants Lateralizing Five Different Sounds



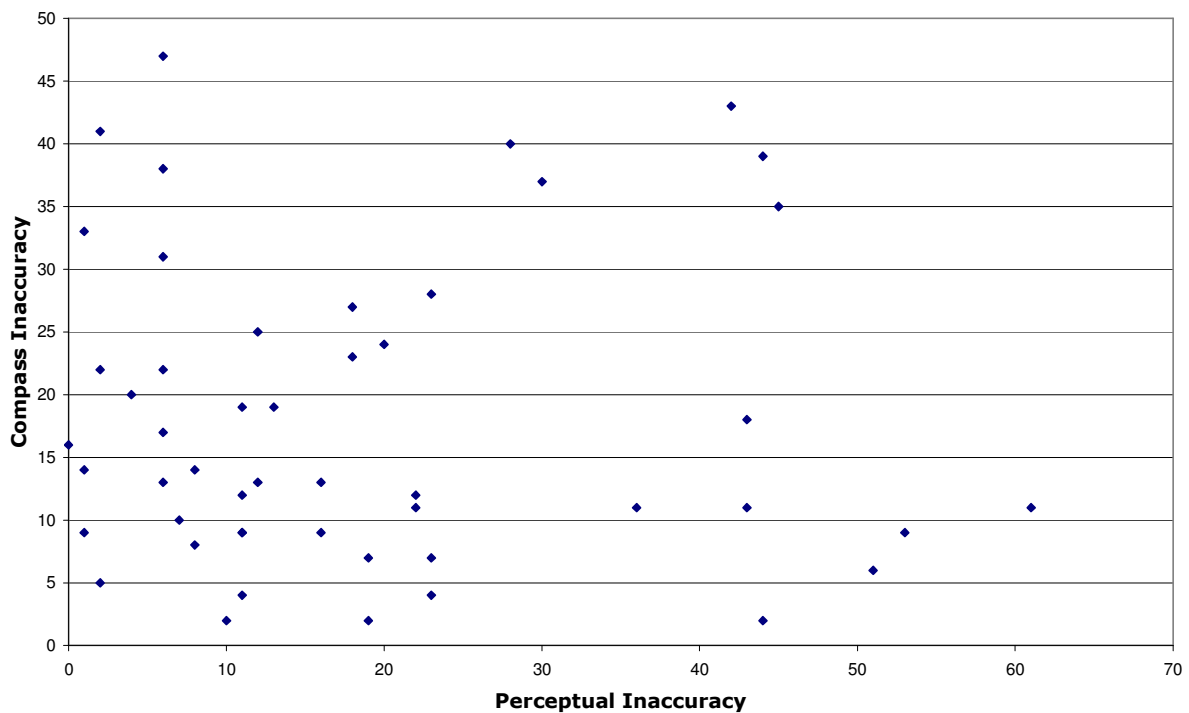
### 6.10. Chart to Show Revised Distribution of Perceptual Inaccuracies





### 6.12. Graph to Show Relationship Between Compass Inaccuracy and Perceptual Inaccuracy

At this stage it worth pointing out the slightly ominous similarity to be found between the means and standard deviations of this revised perceptual inaccuracy data and the compass inaccuracy data found in 6.6. Because of this it is of interest to plot to the two sets of data against each other to ensure that the compass inaccuracy has not skewed the results at each sound lateralization task.



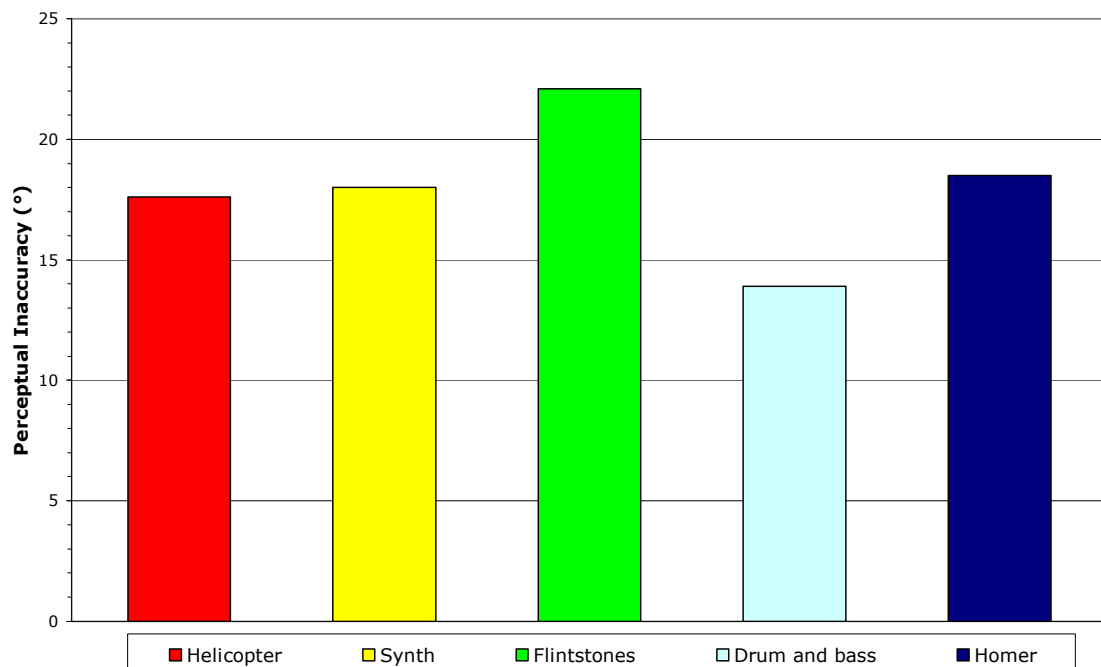
Logically, as has been mentioned, no correlation should exist between the two since unless the compass is particularly unsteady around a particular bearing. The lack of correlation exhibited by the graph would suggest that the compass imperfections are not having an immediate influence upon a participant's ability to lateralize. As reassurance, calculating a pair wise correlation matrix for the two sets of data bears a happily neutral coefficient of -0.01697 indicating that there is practically no correlation.

### 6.13. Chart to Show Average Perceptual Inaccuracies Grouped by Sample

Using as a baseline our revised perceptual inaccuracies which have ‘mod 180’d’, presented here are average perceptual inaccuracies on a per-sample basis.

#	Sample	Mean	Standard Deviation
1	Helicopter	17.6	11.7
2	Synth	18	10.3
3	Flintstones	22.1	14.2
4	Drum and bass	13.9	10
5	Homer	18.5	15.5

The chart below depicts these average perceptual inaccuracies, grouped according to which sample they were associated with.



This does not represent a particularly insightful set of results; no marked pattern or trend seems to be emerging from this data. The only observation we can make with

some degree of certainty is that the fourth sample, the drum and bass one, would seem to have been easier to localize than the other four.

There is no obvious relationship between the participant performance and sample length or frequency makeup and speculating about this is certainly not appropriate. No learning effects (such as participant performance improving sample by sample) are exhibited here either although it would be interesting to repeat the entire test on the same ten participants and investigate the resulting data with this in mind.

## **7. Conclusion**

### **7.1. Summary of Results**

The results obtained suggest that a directional sound system to support navigation is an entirely feasible proposal. Overall participant responses were positive and the sessions provided a novel and ostensibly compelling experience.

Prior to factoring in the front / back reversal anomalies, a rather distorted mean perceptual inaccuracy figure of  $40.7^\circ$  was presented, having an associated standard deviation of  $54.5^\circ$ . These disturbingly high figures were skewed by the eight occasions when participants found ambiguity perceiving whether the sound was being projected from their front or their rear. This is understandable since, although the sound performed seemingly well to head movements, the shape of the sound is identical for a  $0^\circ$  and a  $180^\circ$  projection.

Re-computing the figures (taking into account these reversals and treating them as if the sound was being projected from the opposing hemisphere) yields a far lower mean perceptual inaccuracy of  $18^\circ$  and standard deviation of  $12.3^\circ$ . However, these figures are not quite as persuasive as those arrived at in other studies (see section 2.5). Possible reasons for this short-coming are considered in the next section. Despite this, there is evidence enough to suggest that, even with an average inaccuracy of  $18^\circ$ , participants would be able to navigate towards the source of a sound with relative ease.

### **7.2. Evaluation and Future Work**

Having demonstrated and discussed an interesting array of results stemming from this admittedly unrefined prototype system, consideration is now given to the virtues and short-comings of this investigation; what went wrong, how it feasibly could have been improved and how external constraints have limited the scope of the study.

Throughout its duration, the study has changed course and responded to varying constraints and unexpected happenings. In retrospect, it would be only fair to admit that the initial objectives laid down in the original project proposal document were

unattainable in the time made available. Critically speaking, this should not have been the case although a good argument can be made for why these initial oversights were unforeseen.

### **7.2.1. Software**

A jovial law recited by many software engineers is this: “The first law of software is that it’s late. Always.” Although this is undoubtedly a tongue-in-cheek remark, it is true in many, many cases and for good reasons. Without going into a lengthy discussion, I would assert that this is partly to do with almost immeasurable complexity and partly to do with its abstract nature. In future undertakings, any timing estimates regarding how long software will take to produce, will be doubled from the outset.

It was the belief, at the project’s inception, that the system to be implemented would simulate not only the direction of a sound source but also its range. As is used in the current Mobile Bristol framework, an ultrasound emitter and sensors would have been used to provide data on the user’s location and this in turned used to calculate the range of the sound source. As a natural follow-on from this, the tests were going to be conducted in an entirely different manner and would have included a very interesting and novel audio maze navigation test.

This was not to be the case for a number of reasons and furthermore, as has been mentioned, this would have taken yet more precious software development time. One of the main factors for this was the absence of a sensor bus which allows for two or more serial devices may be simultaneously plugged into the iPAQ. It was believed at the outset that such a device would be available during this project’s lifetime but sadly, but not surprisingly, third party implementing this addition found that the software development process took longer than anticipated.

Not to be deterred, however, an attempt to circumvent this problem (or at least a short term hack) was explored. The idea of using two iPAQ was proposed negating the need for the serial bus; the serial port on each iPAQ could have

been used to accommodate the two serial devices and the iPAQ could theoretically have overcome the problem by continually talking over a wireless network. This was feasible but not practicable in the given time.

Also featuring under this heading and sectioned out for criticism is the iPAQ's operating system, *Microsoft Pocket PC 2002*. To say it is unstable is an understatement. Jokes have been made about the need for Hewlett Packard to relocate the reset button so that is enlarged and positioned centrally on the front of the iPAQ. Its constant crashing and determination to empty its main memory when running out of batteries proved to be a widely acknowledged and constant source of frustration and eventually hatred. On a personal note, the so-called complementary development environment *Embedded Visual C* found a neighbouring place in my heart.

#### 7.2.2. *Hardware*

The next major hindrance which caused the project to change tact was to do with the means of spatializing –or as turned out– lateralizing the sound. The HRTF software kindly provided by Karen Bryden could not be coaxed into working at all in the allotted time. The problem was attributable to the iPAQ's inability to perform the required floating point arithmetic in order to implement the software as is. As mentioned, this could have been addressed by re-coding the software so that it performed its calculations solely in integer arithmetic but once again, this would have required yet more time.

As an alternative to the HRTF software, FMOD, the computationally fast and programmer-friendly API package was used. This was still a good choice. It does in fact provided support for 3D hardware acceleration – yet the iPAQ does not currently complement this so it is not taken advantage of at runtime. If the target machine is equipped with suitable hardware (and most new desktop computers are) the FMOD engine will take full advantage of the onboard sound acceleration and advanced 3D algorithms which are part of EAX (see 4.1.2).

Another major short-coming of the hardware side of the implementation was the compass itself. Had the reputedly more stable and more accurate Honeywell compass actually worked, then this may have favourably affected the results. As it stands, great gratitude is owed to Cliff Randall who was kind enough to provide the ill-mannered Vector compass. The effects of the highly tilt-sensitive Vector were apparent to all participants as they were reminded to keep their heads perfectly level, using as an aid (and rather unscientifically) the chair's headrest.

Nevertheless, to be more positive these deficiencies can all be resolved. The software to interface with the cursed Honeywell compass has now been written so replacing the Vector with the more reliable three-axis compass should be facile. The HRTF algorithm *can* be rewritten to use only integer arithmetic and this would drastically improve the sound spatialization. This would probably explain the perceptual inaccuracy measurements, which were almost twice that of the study making use of the far superior HRTF's as outlined in 2.5.1. It was originally hoped to improve upon these scores as the hardware and software have both matured in the interim period.

### **7.2.3. Design**

In terms of evaluating the experimental design, there is still much that can be improved upon. The aim of the pilot studies was to highlight any changes which needed making but they still failed to draw attention to a number of prominent weaknesses.

Why the experimental design failed to include asking participants simple questions such as their age and their experience with sound is a true mockery. This information would have been extremely easy to acquire and its glaring absence was a blatant oversight. It is highly vexing to point out that the same can be said for not having programmed the iPAQ record the times of the lateralization tasks.

The five sound lateralization tasks themselves were originally each to be repeated by participants, with varying levels of background noise in each repetition. Again, this would have been extremely straightforward to implement and would have added a completely different and extremely useful dimension to the data obtained.

It is sincerely hoped that carrying out this project in the way it was originally intended will provide the basis for future work in this very interesting area.

### **7.3. Implications**

The findings presented here have noteworthy implications for the implementation of the release candidate of any directional sound system intended to support navigation.

Of primary importance is the need for a three axis compass which will hence detect changes of orientation in both the lateral plane and in terms of elevation. Although the cost is a key issue in making this choice, it is my opinion that anything less than this simply cannot provide the basis for a reliable or accurate *head mounted* compass.

Moreover, the need for sufficient and independently assessed auditory cues is clearly going to be of great importance. Whilst it is accepted that the average accuracy demonstrated by this coarse system will doubtless improve when upgrading the sound projection algorithm to one which encompasses HRTF's, other studies have shown that troublesome front / back and up / down ambiguities still endure. It is suggested that an additional auditory cue is put in place to aid resolution of this confusion. A simple low-pass filter may be all that is required.

Many of the personal digital assistants (PDA's – such as the iPAQ, Palm Pilot and innumerable others) available today do not come equipped with 3D accelerated sound or FPU processors. It is my expectation that PDA's of the future will slide into the role of Walkman's and Minidisc players. If this transformation happens, it is my opinion that the inbuilt sound card will play an increasingly important role and this in turn will facilitate far easier programming of spatialized sound. For the time being though, software must be optimised in a way to minimise floating point arithmetic.



## 8. Appendices

<b>A. References</b> .....	<b>56</b>
A.1. <i>Books</i> .....	56
A.2. <i>Websites &amp; Online Documents</i> .....	56
<b>B. Test Results – Raw Data</b> .....	<b>58</b>
B.1.    Participant Data Sheets.....	62
<b>C. Source Code Listing</b> .....	<b>75</b>
C.1. <i>Context.cpp</i> .....	76
C.2. <i>Sensor.cpp</i> .....	84
C.3. <i>Serial.cpp</i> .....	104
C.4. <i>SettingsDlg.cpp</i> .....	110
C.5. <i>StdAfx.cpp</i> .....	113
C.6. <i>UStest.cpp</i> .....	114
C.7. <i>UStestDlg.cpp</i> .....	117
<b>D. Project Timetable</b> .....	<b>132</b>
<b>E. Thanks and Acknowledgements</b> .....	<b>133</b>

## A. References

References used through the undertaking of this study have been categorised into two groups.

### A.1. Books & Printed Journals

“An Introduction to the Psychology of Hearing” - Moore, Brian C. J.

"Challenges to the Successful Implementation of 3-D Sound" - Begault, Durand

"Techniques for Low Cost Spatial Audio" - Burgess, David A.

"Sound Rendering" - Computer Graphics - Takala, Tapio & Hahn.

“Virtual Reality Systems” – Vince, John

“Spatial Hearing - The Psychophysics of Human Sound Localization” – Blauert, Jens

**AES: Journal of the Audio Engineering Society**

*Numerous editions with 3D auditory specialization references.*

### A.2. Websites & Online Documents

“Localization with non-individualized Virtual Acoustic Display Cues” - Elizabeth M. Wenzel, Frederic L. Wightrnan & Doris J. Kistler

“Head Motion and Latency Compensation on Localization of 3D Sound in Virtual Reality” - Jiann-Rong Wu, Cha-Dong Duh, Ming Ouhyoung

“Self-Organization Of A Sound Source Localization Robot By Perceptual Cycle” - Hiromichi Nakashima, Toshiharu Mukai, Noboru Ohnishi

[http://www.bmc.riken.go.jp/~sensor/papers/ICONIP2002\(Nakashima\).pdf](http://www.bmc.riken.go.jp/~sensor/papers/ICONIP2002(Nakashima).pdf)

“Multi-Sensory Rendering for Viewpoint-Independent Scenes” – Pope, Jackson

[http://citeseer.nj.nec.com/cache/papers/cs/24007/http://zSzzSzwww.cs.bris.ac.ukzSz~popezSzpaperszSzjackson\\_pope\\_phd\\_thesis.pdf/multi-sensory-rendering-for.pdf](http://citeseer.nj.nec.com/cache/papers/cs/24007/http://zSzzSzwww.cs.bris.ac.ukzSz~popezSzpaperszSzjackson_pope_phd_thesis.pdf/multi-sensory-rendering-for.pdf)

“Realtime Room Acoustics Using Ambisonics” – Pope, Jackson

<http://www.cs.bris.ac.uk/Tools/Reports/Abstracts/1999-pope-0.html>

“HRTF 3D Positional Audio Technology Overview” - Unknown author

[http://www.cmedia.com.tw/pdf/e\\_c3d\\_tech.pdf](http://www.cmedia.com.tw/pdf/e_c3d_tech.pdf)

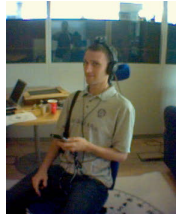

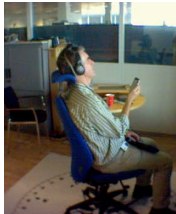

**Online Bibliography of Sound Localization**

<http://www.tonmeister.ca/main/bibliography/localization.html>


**FMOD - Tutorial and Forum**

<http://www.fmod.org/>

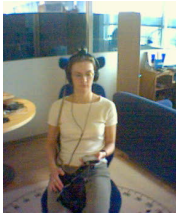
## B. Test Results – Raw Data

Picture	#	Participant Name	Actual Azimuth (°)	Compass Reading (°)	Real World Bearing (°)	Approximate Compass Inaccuracy (°)	Mean Compass Inaccuracy (°)	Perceptual Inaccuracy(Δ)	Mean perceptual inaccuracy(ΣΔ)
	1	Ben Clayton	139	182	140	42		43	
			41	65	45	20		24	
			107	103	80	23		4	
			214	202	180	22		12	
			220	211	195	16	<b>24.6</b>	9	<b>18.4</b>
<b>Participant Comments:</b> "Better than I thought it would be. You can definitely hear the sound moving around."									
	2	Erik Geelhoed	3	17	355	8		14	
			29	41	30	11		12	
			185	212	230	18		27	
			217	36	25	11		179	
			249	238	260	22	<b>14</b>	11	<b>48.6</b>
<b>Participant Comments:</b> "Very impressive but I could not tell the difference between front and back."									
	3	Richard Hull	100	119	130	11		19	
			236	220	220	0		16	
			240	262	260	2		22	
			17	42	30	12		25	
			268	270	280	10	<b>7</b>	2	<b>16.8</b>
	4	Rachel Yeardeley	76	87	130	43		11	
			116	107	160	53		9	
			55	64	75	11		9	

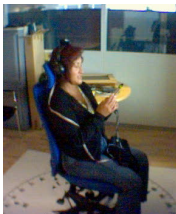
270	279	280	1	9		
344	6	0	6	<b>22.8</b>	22	<b>12</b>


Picture	#	Participant Name	Actual Azimuth (°)	Compass Reading (°)	Real World Bearing (°)	Approximate Compass Inaccuracy (°)	Mean Compass Inaccuracy (°)	Perceptual Inaccuracy(Δ)	Mean perceptual inaccuracy(ΣΔ)
	5	Gary Porter	67	234	250	16		167	
			124	126	145	19		2	
			55	46	35	11		9	
			84	77	100	23		7	
			340	121	165	44	<b>22.6</b>	141	<b>65.2</b>

**Participant Comments:** "Wow! It's really hard... I cheated by lining up the sound in one ear then rotating through 90°!"

	6	Sophie Bevan	308	341	340	1		33	
			264	292	315	23		28	
			332	12	40	28		40	
			70	215	260	45		145	
			238	56	100	44	<b>28.2</b>	178	<b>84.8</b>

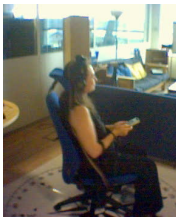
**Participant Comments:** "Harder than I thought it would be...you really have to concentrate."

	7	Jo Reid	35	204	265	61		169	
			233	270	300	30		37	
			3	34	40	6		31	
			163	169	220	51		6	
			6	44	50	6	<b>30.8</b>	38	<b>56.2</b>


	8	Tom Milner	348	353	355	2		5	
			118	129	165	36		11	
			238	251	245	6		13	
			10	24	25	1		14	

246	238	230	8	10.6	8	10.2
-----	-----	-----	---	------	---	------

**Participant Comments:** "Pretty hard but I have musical ears!"

Picture	#	Participant Name	Actual Azimuth (°)	Compass Reading (°)	Real World Bearing (°)	Approximate Compass Inaccuracy (°)	Mean Compass Inaccuracy (°)	Perceptual Inaccuracy(Δ)	Mean perceptual inaccuracy(ΣΔ)
	9	Miranda Mowbray	187	357	350	7		170	
			131	113	70	43		18	
			97	144	150	6		47	
			57	64	45	19		7	
			223	2	0	2	15.4	139	76.2

**Participant Comments:** "It's good but the sound kept jumping from one ear to the other."

	10	Jennifer Newman	73	56	50	6		17	
			315	292	310	18		23	
			246	227	240	13		19	
			29	49	45	4		20	
			101	88	100	12	10.6	13	18.4

**Participant Comments:** "Harder than I thought...how did I do?"



## **B.1. Participant Data Sheets**

The following pages (included for the sake of completeness) are the original data sheets used to record the 'real world bearings' towards which a participant was facing. Presented here are data sheets from the two pilot studies, together with the ten from the main sessions.

























## C. Source Code Listing

Presented in this appendix is the source code for the test application. Note that any header files have not been included in order that this document remains relatively compact.

The functionality is relatively straight forward but understanding it may not be. It should be noted that the code has been written with neither elegance, performance nor recoverability from errors in mind. Comments are present in the code but are not extensive.

To outline *very* basically what it does is straight forward;

- The program allocates five software channels for the five sounds used in the test.
- The main loop of the program reads from the serial port to obtain the compass heading.
- Upon the first reading, the compass heading is stored and subtracted from all subsequent compass readings to achieve the effect that the compass automatically ‘calibrates’ to 0° every time the program starts.
- A random number is generated and from this angle the sound is positioned.
- Using this compass heading of  $\theta$ , the sound is repositioned since this creates the same effect as head movement. In reality the compass reading makes the sound move around the listeners head, rather than the head rotation directly affecting the sounds’ projection – but the effect is the same.

The source has been syntax highlighted to ease (!) comprehension...

## C.1. Context.cpp

```
1: // Context.cpp: implementation of the CContext class.
2: //
3: //////////////////////////////////////
4:
5: #include "stdafx.h"
6: #include "USTest.h"
7: #include "Context.h"
8: // #include "MediaEvent.h"
9:
10: #ifdef _DEBUG
11: #undef THIS_FILE
12: static char THIS_FILE[]=__FILE__;
13: #define new DEBUG_NEW
14: #endif
15:
16: //////////////////////////////////////
17: // Construction/Destruction
18: //////////////////////////////////////
19:
20:
21: CContext::CContext ()
22: {
23:     int i;
24:
```

```
25:  m_pingno = 0;
26:  m_location.x = 0;
27:  m_location.y = 0;
28:  for (i = 1; i <= 8; i++)
29:      m_time[i] = 0;
30:
31:  m_lastLocation = m_location;
32:  m_nearest = 0;
33:  m_numtx = 0;
34:  m_xpos = -1;
35:  m_ypos = -1;
36:  m_txid = "";
37:  m_listener = NULL;
38:  m_listenerReady = false;
39:  m_changePending = false;
40:  InitializeCriticalSection(&m_lock);
41: }
42:
43:
44:
45: CContext::~CContext ()
46: {
47:     DeleteCriticalSection(&m_lock);
48: }
49:
50:
51:
52: //////////////////////////////////////
```

```
53: // Access functions
54: //////////////////////////////////////
55:
56:
57: CPoint CContext::location()
58: {
59:     return m_location;
60: }
61:
62:
63: int CContext::pingno()
64: {
65:     return m_pingno;
66: }
67:
68: int CContext::time(int tx)
69: {
70:     return m_time[tx];
71: }
72:
73:
74: int CContext::nearest()
75: {
76:     return m_nearest;
77: }
78:
79:
80: int CContext::numtx()
```

```
81: {
82:     return m_numtx;
83: }
84:
85:
86: int CContext::xpos ()
87: {
88:     return m_xpos;
89: }
90:
91:
92: int CContext::ypos ()
93: {
94:     return m_ypos;
95: }
96:
97: CString CContext::txid()
98: {
99:     return m_txid;
100: }
101:
102:
103: int CContext::heading()
104: {
105:     return m_hdg;
106: }
107:
108: void CContext::raw(int *buf)
```

```
109: {
110:     buf[0] = m_raw[0];
111:     buf[1] = m_raw[1];
112:     buf[2] = m_raw[2];
113: }
114:
115: void CContext::setLocation(int pingno, CString txid, int time[], int xpos, int ypos, long x, long y)
116: {
117:     POINT p;
118:     p.x = x;
119:     p.y = y;
120:     setLocation(pingno, txid, time, xpos, ypos, p);
121: }
122:
123:
124: void CContext::setLocation(int pingno, CString txid, int time[], int xpos, int ypos, CPoint p)
125: {
126:     int i;
127:     int mintime;
128:     int numtx;
129:
130:     TRACE (_T("x = %d, y = %d\n"), p.x, p.y);
131:     EnterCriticalSection(&m_lock);
132:
133:     m_pingno = pingno;
134:     m_lastLocation = m_location;
135:     m_location = p;
136:
```

```
137: numtx = 0;
138: mintime = 256;
139:
140: for (i = 1; i <= 8; i++) {
141:     m_time[i] = time[i];
142:
143:     if (time[i] < mintime) {
144:         m_nearest = i;
145:         mintime = time[i];
146:     }
147:
148:     if (time[i] < 250)
149:         numtx++;
150: }
151:
152: if (mintime >= 250)
153:     m_nearest = 0;
154:
155: m_numtx = numtx;
156: m_xpos = xpos;
157: m_ypos = ypos;
158: m_txid = txid;
159:
160: if (m_listener && m_listenerReady) {
161:     PostMessage(m_listener, UWM_MB_LOCATION_CHANGED, 0, 0); // Two unused parameters
162:     m_listenerReady = false;
163:     m_changePending = false;
164: }
```

```
165:     else {
166:         m_changePending = true;
167:     }
168:
169:     LeaveCriticalSection(&m_lock);
170: }
171:
172: void CContext::setHeading(BYTE *buf, int hdg)
173: {
174:     // TRACE (_T("x = %d, y = %d\n"), p.x, p.y);
175:     EnterCriticalSection(&m_lock);
176:
177:     //m_pingno = pingno;
178:     m_lastLocation = m_location;
179:     //m_location = p;
180:
181:     m_hdg = hdg;
182:     m_raw[0] = buf[0];
183:     m_raw[1] = buf[1];
184:     m_raw[2] = buf[2];
185:
186:     if (m_listener && m_listenerReady) {
187:         PostMessage(m_listener, UWM_MB_LOCATION_CHANGED, 0, 0); // Two unused parameters
188:         m_listenerReady = false;
189:         m_changePending = false;
190:     }
191:     else {
192:         m_changePending = true;
```

```
193:     }
194:
195:     LeaveCriticalSection(&m_lock);
196: }
197:
198:
199: void CContext::setListener(HWND h)
200: {
201:     m_listener = h;
202: }
203:
204:
205: void CContext::setListenerReady()
206: {
207:     EnterCriticalSection(&m_lock);
208:     m_listenerReady = true;
209:     if (m_changePending) {
210:         PostMessage(m_listener, UWM_MB_LOCATION_CHANGED, 0, 0);
211:         m_listenerReady = false;
212:         m_changePending = false;
213:     }
214:     LeaveCriticalSection(&m_lock);
215: }
```

## C.2. *Sensor.cpp*

```
// Sensor.cpp: implementation of the CSensor class.
//
////////////////////////////////////

#include "stdafx.h"
#include "USTest.h"
#include "Sensor.h"
#include "time.h"

#ifdef _DEBUG
#undef THIS_FILE
static char THIS_FILE[]=__FILE__;
#define new DEBUG_NEW
#endif

#define PI 3.14159265

#include "fmod.h"
#include "common.h"

#define DISTANCESCALE      100.0f
```

```

#define INTERFACE_UPDATETIME 100

FSOUND_SAMPLE *samp1      = NULL;
FSOUND_SAMPLE *samp2      = NULL;
FSOUND_SAMPLE *samp3      = NULL;
int          channel1     = -1;
int          channel2     = -1;
float        soundpos[3] = { 0.0f, 0.0f, 0.0f };
char        listenerflag = 1;

////////////////////////////////////
// Construction/Destruction
////////////////////////////////////

CSensor::CSensor()
{
    bool stat = m_serialPort.open();
    if (!stat) {
        TRACE(_T("Error opening serial port\n"));
        AfxMessageBox(_T("Fail"));
    }
    m_context = NULL;
    m_position.x = 0.0;
    m_position.y = 0.0;
    m_position.z = 0.0;
    m_position.hdg = 0;
    m_xgrid = GRID;
    m_ygrid = GRID;
    m_pre = HP_PRE;

```

```
        m_lastPosition = m_position;
m_xpos = -1;
m_ypos = -1;
m_hdg = -1;
m_txid = "";
        m_running = false;
        calibrate_mode = false;
}
```

```
CSensor::~~CSensor()
```

```
{
    m_serialPort.close();
    stop();
}
```

```
void CSensor::Trigger ()
```

```
{
#ifdef HONEYWELL
    m_serialPort.PulseRTS();
#else
    m_serialPort.write('$');
    m_serialPort.write('S');
    m_serialPort.write('C');
    m_serialPort.write('0');
#endif
}
```

```
void CSensor::CalibrateOn()
{
    calibrate_mode=true;
}

void CSensor::CalibrateOff()
{
    calibrate_mode=false;
}

void CSensor::ResetHDG()
{
    first_heading = -1;
}

void CSensor::SerialBreak(bool state)
{
    m_serialPort.SerialBreak(state);
}

void CSensor::setContext(CContext* c)
{
    m_context = c;
    m_running = true;
    AfxBeginThread(run, this, THREAD_PRIORITY_TIME_CRITICAL);
}
```

```
void CSensor::setGrid (double xgrid, double ygrid, double pre)
{
    m_xgrid = xgrid;
    m_ygrid = ygrid;
    m_pre = pre;
}

void CSensor::getGrid (double *xgrid, double *ygrid, double *pre)
{
    if (xgrid != NULL)
        *xgrid = m_xgrid;

    if (ygrid != NULL)
        *ygrid = m_ygrid;

    if (pre != NULL)
        *pre = m_pre;
}

void CSensor::WriteData ()
{
    HANDLE OutFile = CreateFile(L"\\Results.txt",
                                GENERIC_WRITE,
                                FILE_SHARE_WRITE,
                                NULL,
```

```

OPEN_ALWAYS,
FILE_ATTRIBUTE_NORMAL,
NULL);

SetFilePointer(OutFile, 0, 0, FILE_END);

char buffer[100];

sprintf(buffer, "Actual azimuth: %d\t\t Compass reading: %d\n\n", sound_azimuth, m_hdg); //scope problem?

DWORD numBytesWritten = 0;
WriteFile(OutFile, buffer, sizeof(buffer), &numBytesWritten, NULL);
int i = GetLastError();

CloseHandle(OutFile);

////////////////////////////////////

SYSTEMTIME time;
GetSystemTime(&time);
//srand( (unsigned)&time); // set initial seed

if (!FSOUND_GetPaused(channel1))
{
    FSOUND_SetPaused(channel1, TRUE);
}

```

```
    FSOUND_SetPaused(channel2, FALSE);
    srand( (unsigned int)GetTickCount()); // set initial seed
}
else if (!FSOUND_GetPaused(channel2))
{
    FSOUND_SetPaused(channel2, TRUE);
    FSOUND_SetPaused(channel3, FALSE);

}
else if (!FSOUND_GetPaused(channel3))
{
    FSOUND_SetPaused(channel3, TRUE);
    FSOUND_SetPaused(channel4, FALSE);

}
else if (!FSOUND_GetPaused(channel4))
{
    FSOUND_SetPaused(channel4, TRUE);
    FSOUND_SetPaused(channel5, FALSE);

}
else if (!FSOUND_GetPaused(channel5))
{
    FSOUND_SetPaused(channel5, TRUE);

}
sound_azimuth = rand() % 360; //(int) 360 * rand() / (RAND_MAX + 1.0); //pick another random angle
}
```

```
////////////////////////////////////  
// Utility functions  
////////////////////////////////////  
  
void CSensor::update()  
{  
    /*  
    // Debug  
    if (m_context) {  
        CPoint location = m_context->location();  
        long y = location.y + 25 > 900 ? -1200 : location.y + 25;  
        m_context->setLocation(location.x, y);  
        TRACE(_T("%d %d\n"), location.x, location.y);  
    }  
    Sleep(1000);  
    */  
    static int pingno = 0;  
    BYTE packet[MAX_PACKET_LENGTH];  
    int time[10];  
    int stat;  
    int i, j;  
  
    long int x, y;  
  
    float vel[3] = { 0,0,0 };  
}
```

```

static float t = 0;

if (!calibrate_mode)
    Trigger();

stat = read(packet);

switch (stat) {
case SENSOR_ERROR:
    //          TRACE(_T("Error reading sensor \n"));
    // Do nothing
    break;
case ULTRASOUND_DATA:
    //          TRACE(_T("Ultrasound reading \n"));
    pingno++;

    for (i = 1, j = PINGER_DATA_OFFSET; i <= 8; i++, j++)
        time[i] = packet[j];

    getLocationUS(packet);

    x = long(ceil(m_position.x * 100.0));
    y = long(ceil(m_position.y * 100.0));

    //    TRACE (_T("x = %d, y = %d\n"), x, y);
    m_context->setLocation(pingno, m_txid, time, m_xpos, m_ypos, x, y);
    break;
}

```

```
case HEADING_DATA:

    //          TRACE(_T("Heading reading \n"));

    m_hdg = DecodePkt(packet);

    // if first_heading is set to -1 then hdg is reset to 0 irrespective of true north

    if (first_heading == -1)
        first_heading = m_hdg; // store initial heading value

    if (m_hdg >= first_heading)
    {
        m_hdg = m_hdg - first_heading;
    }
    else
    {
        m_hdg = 360 + m_hdg - first_heading;
    }

    m_hdg = m_hdg;
```

```

soundpos[0] = (float) 500*sin((-m_hdg + sound_azimuth) * (PI / 180));
//pos[1] = ;
soundpos[2] = (float) 500*cos((-m_hdg + sound_azimuth) * (PI / 180));

FSOUND_3D_SetAttributes(channel1, &soundpos[0], NULL); //Move the sound source acc. head dir
FSOUND_3D_SetAttributes(channel2, &soundpos[0], NULL); //Move the sound source acc. head dir
FSOUND_3D_SetAttributes(channel3, &soundpos[0], NULL); //Move the sound source acc. head dir
FSOUND_3D_SetAttributes(channel4, &soundpos[0], NULL); //Move the sound source acc. head dir
FSOUND_3D_SetAttributes(channel5, &soundpos[0], NULL); //Move the sound source acc. head dir

//FSOUND_3D_Listener_SetAttributes(NULL, NULL, -100.0f, 0, 0, 0, 0, 0);

FSOUND_Update();

m_context->setHeading(packet, m_hdg);

break;

case GPS_DATA:
    //          TRACE(_T("GPS reading \n"));
    // Do nothing
    break;
}

// Sleep (90); // Slow down

```

```

}

int CSensor::DecodePkt(BYTE *buf)
{
#ifdef HONEYWELL
    return ((buf[2] + (buf[1] * 256)) / 2);
#else
    return ((buf[2] + (buf[1] * 256)));
#endif
}

void CSensor::getLocationUS(BYTE* packet)
{
    double dist[NUM_OF_PINGERS];
    int flag[NUM_OF_PINGERS];
    int i;
    double d_units;
    CString label;

    // Ultrasonic packet format from PIC16F84
    // Bytes 0-7 = Label from RF pinger
    // Bytes 8-15 = Raw distances from ultrasonic receivers
    // Byte 16 = terminator = ':'

    // Extract RF pinger label for interest
    for (i = 0; i < RF_LABEL_LENGTH; i++)

```

```

        label += (char) packet[i];

m_txid = label;
    //TRACE(label + "\n");

    // Extract raw distance information
    for (i = 0; i < NUM_OF_PINGERS; i++) {
        flag[i] = 0;

        d_units = (double)packet[i + PINGER_DATA_OFFSET];
        dist[i] = (UNIT_DIST * (d_units / 1000.0)) + m_pre;

        if ((dist[i] > 1.0) && (dist[i] < (RANGE + m_pre))) {
            flag[i]++;                // in range o.k.
        }
    }

    m_lastPosition.x = m_position.x;
    m_lastPosition.y = m_position.y;
m_lastPosition.z = m_position.z;
m_lastPosition.hdg = m_position.hdg;

}

```

```
int CSensor::read(BYTE *packet)
{
    int retStat = SENSOR_ERROR;
    BYTE inByte1 = 0;
    BYTE inByte2 = 0;
    BYTE inByte3 = 0;

#ifdef HONEYWELL

    inByte1 = m_serialPort.read();
    inByte2 = m_serialPort.read();
    inByte3 = m_serialPort.read();

#else

    do {
        inByte1 = m_serialPort.read();
    }
    while (inByte1 != '$');

    inByte1 = m_serialPort.read(); // read R
    inByte1 = m_serialPort.read(); // then C
    inByte1 = m_serialPort.read(); // then 0
    inByte2 = m_serialPort.read(); // we want this byte
    inByte3 = m_serialPort.read(); // and this one
//////////delme
```

```
inByte1 = m_serialPort.read(); // reading CR
inByte1 = m_serialPort.read(); // reading LF

////delme

#endif

//Sleep (50);

//inByte1 = 0x80;
//inByte2 = 0x02;
//inByte3 = 0x85;

//if (inByte1 == 0x80 || inByte1 == 0x81) {
    packet[0] = inByte1;
    packet[1] = inByte2;
    packet[2] = inByte3;
    retStat = HEADING_DATA;
//}

return retStat;
}
```

```
////////////////////////////////////
```

```

// Thread functions
////////////////////////////////////

UINT CSensor::run(LPVOID p)
{
    CSensor *me = (CSensor *)p;
    me->run();
    return 0;
}

void CSensor::run()
{
    first_heading = -1;

    SYSTEMTIME time;
    GetSystemTime(&time);
    srand( (unsigned int)GetTickCount()); // set initial seed
    //srand( (unsigned)time( NULL ) );

    sound_azimuth = rand() % 360; //(int) 360 * rand() / (RAND_MAX + 1.0);

    int retrycount = 0;
    while (!FSOUND_Init(22050, 48, FSOUND_INIT_GLOBALFOCUS) && retrycount < 10)
    {
        Sleep(100);
    }
}

```

```

        retrycount++;
    }

    if (retrycount == 10)
    {
        //Common_DisplayError();
    }

    FSOUND_SAMPLE *samp1 = NULL;
    FSOUND_SAMPLE *samp2 = NULL;
    FSOUND_SAMPLE *samp3 = NULL;
    FSOUND_SAMPLE *samp4 = NULL;
    FSOUND_SAMPLE *samp5 = NULL;

    FSOUND_3D_SetDistanceFactor(DISTANCESCALE);

    // =====
    // 3D HARDWARE MONO
    // =====
    samp1 = FSOUND_Sample_Load(FSOUND_UNMANAGED, "\\My Documents\\Personal\\hel.wav", FSOUND_2D |
FSOUND_NORMAL, 0);

    FSOUND_Sample_SetMinMaxDistance(samp1, 4.0f * DISTANCESCALE, 1000.0f * DISTANCESCALE); //
increasing mindistance makes it louder in 3d space
    FSOUND_Sample_SetMode(samp1, FSOUND_LOOP_NORMAL);

```

```

samp2 = FSOUND_Sample_Load(FSOUND_UNMANAGED, "\\My Documents\\Personal\\string.wav", FSOUND_2D
| FSOUND_NORMAL, 0);
FSOUND_Sample_SetMinMaxDistance(samp2, 4.0f * DISTANCESCALE, 1000.0f * DISTANCESCALE); //
increasing mindistance makes it louder in 3d space
FSOUND_Sample_SetMode(samp2, FSOUND_LOOP_NORMAL);

samp3 = FSOUND_Sample_Load(FSOUND_UNMANAGED, "\\My Documents\\Personal\\flint.wav", FSOUND_2D |
FSOUND_NORMAL, 0);
FSOUND_Sample_SetMinMaxDistance(samp3, 4.0f * DISTANCESCALE, 1000.0f * DISTANCESCALE); //
increasing mindistance makes it louder in 3d space
FSOUND_Sample_SetMode(samp3, FSOUND_LOOP_NORMAL);

samp4 = FSOUND_Sample_Load(FSOUND_UNMANAGED, "\\My Documents\\Personal\\dnb.wav", FSOUND_2D |
FSOUND_NORMAL, 0);
FSOUND_Sample_SetMinMaxDistance(samp4, 4.0f * DISTANCESCALE, 1000.0f * DISTANCESCALE); //
increasing mindistance makes it louder in 3d space
FSOUND_Sample_SetMode(samp4, FSOUND_LOOP_NORMAL);

samp5 = FSOUND_Sample_Load(FSOUND_UNMANAGED, "\\My Documents\\Personal\\doh.wav", FSOUND_2D |
FSOUND_NORMAL, 0);
FSOUND_Sample_SetMinMaxDistance(samp5, 4.0f * DISTANCESCALE, 1000.0f * DISTANCESCALE); //
increasing mindistance makes it louder in 3d space
FSOUND_Sample_SetMode(samp5, FSOUND_LOOP_NORMAL);

channel1 = FSOUND_PlaySound(FSOUND_FREE, samp1);
channel2 = FSOUND_PlaySound(FSOUND_FREE, samp2);

```

```

channel3 = FSOUND_PlaySound(FSOUND_FREE, samp3);
channel4 = FSOUND_PlaySound(FSOUND_FREE, samp4);
channel5 = FSOUND_PlaySound(FSOUND_FREE, samp5);

FSOUND_SetPaused(channel2, TRUE);
FSOUND_SetPaused(channel3, TRUE);
FSOUND_SetPaused(channel4, TRUE);
FSOUND_SetPaused(channel5, TRUE);

//channel1 = FSOUND_PlaySoundEx(FSOUND_FREE, samp1, NULL, TRUE);

{
    // Set Listener pos to (0,0,0)
    float soundpos[3] = { 0.0f, 0.0f, 0.0f };
    FSOUND_3D_Listener_SetAttributes(soundpos, NULL, 0, 0, 1.0f, 0, 1.0f, 0);

    // Set sound source
    //float pos[3] = { -10.0f * DISTANCESCALE, -0.0f, 1.0f * DISTANCESCALE };
    float pos[3] = { 0 * DISTANCESCALE, -0.0f, 10.0f * DISTANCESCALE };

    FSOUND_3D_SetAttributes(channel1, pos, NULL);
if (!FSOUND_SetPaused(channel1, FALSE))
{
    //Common_DisplayError();

```

```
    }  
}
```

```
    while (m_running) {  
        update();  
    }
```

```
    }  
}
```

```
void CSensor::stop()  
{  
    m_running = false;  
}
```

### C.3. Serial.cpp

```
1: // Serial.cpp: implementation of the CSerial class.
2: //
3: //////////////////////////////////////
4:
5: #include "stdafx.h"
6: #include "USTest.h"
7: #include "Serial.h"
8:
9: #ifdef _DEBUG
10: #undef THIS_FILE
11: static char THIS_FILE[]=__FILE__;
12: #define new DEBUG_NEW
13: #endif
14:
15: //////////////////////////////////////
16: // Construction/Destruction
17: //////////////////////////////////////
18:
19: CSerial::CSerial()
20: {
21:     m_comPort = INVALID_HANDLE_VALUE;
22: }
23:
24:
```

```

25:
26: CSerial::~CSerial()
27: {
28:     close();
29: }
30:
31:
32: ////////////////////////////////////////////////////////////////////
33: // Access functions
34: ////////////////////////////////////////////////////////////////////
35:
36:
37: bool CSerial::open()
38: {
39:     DCB dcb;
40:     COMMTIMEOUTS cto;
41:
42:     // Close port if already open
43:     close();
44:     // Open port
45:     m_comPort = CreateFile (SERIAL_PORT,
46:                             GENERIC_READ | GENERIC_WRITE,
47:                             0,
48:                             NULL,
49:                             OPEN_EXISTING,
50:                             0,
51:                             NULL);
52:     if (m_comPort == INVALID_HANDLE_VALUE) {

```

```
53:         TRACE(_T("Error opening serial port\n"));
54:         AfxMessageBox(_T("Error opening serial port."));
55:         return false;
56:     }
57:
58:     // Configure port.
59:     GetCommState (m_comPort, &dcb);
60:     dcb.BaudRate = BIT_RATE;
61:     dcb.fParity = FALSE;
62:     dcb.fNull = FALSE;
63:     dcb.StopBits = ONESTOPBIT;
64:     dcb.Parity = NOPARITY;
65:     dcb.ByteSize = 8;
66:     dcb.fRtsControl = RTS_CONTROL_ENABLE;
67:     SetCommState (m_comPort, &dcb);
68:
69:
70:
71:     // Set the timeouts. Set infinite read timeout.
72:     cto.ReadIntervalTimeout = 100;
73:     cto.ReadTotalTimeoutMultiplier = 0;
74:     cto.ReadTotalTimeoutConstant = 0;
75:     cto.WriteTotalTimeoutMultiplier = 100;
76:     cto.WriteTotalTimeoutConstant = 0;
77:     SetCommTimeouts (m_comPort, &cto);
78:
79:     // And return
80:     return true;
```

```
81: }
82:
83:
84:
85: void CSerial::PulseRTS ()
86: {
87:     DCB dcb;
88:
89:     GetCommState (m_comPort, &dcb);
90:     dcb.fRtsControl = RTS_CONTROL_DISABLE;
91:     SetCommState (m_comPort, &dcb);
92:
93:     Sleep(2);
94:
95:     GetCommState (m_comPort, &dcb);
96:     dcb.fRtsControl = RTS_CONTROL_ENABLE;
97:     SetCommState (m_comPort, &dcb);
98: }
99:
100: void CSerial::SerialBreak(bool state)
101: {
102:
103:     if (state == true)
104:         SetCommBreak(m_comPort);
105:     else
106:         ClearCommBreak(m_comPort);
107: }
108:
```

```
109:
110:
111: void CSerial::close()
112: {
113:     if (m_comPort != INVALID_HANDLE_VALUE) {
114:         CloseHandle(m_comPort);
115:         m_comPort = INVALID_HANDLE_VALUE;
116:     }
117: }
118:
119:
120:
121: BYTE CSerial::read()
122: {
123:     BYTE retByte;
124:     unsigned long cBytesR;
125:     if (m_comPort == INVALID_HANDLE_VALUE) {
126:         TRACE(_T("Serial port is not open\n"));
127:         return 0;
128:     }
129:     ReadFile (m_comPort, &retByte, 1, &cBytesR, 0);
130:     if (cBytesR == 0 ) {
131:         TRACE(_T("Error reading serial port\n"));
132:         return 0;
133:     }
134:     return retByte;
135: }
136:
```

```
137:
138: int CSerial::write(BYTE ch)
139: {
140:     unsigned long cBytesW;
141:     BYTE buf;
142:
143:     if (m_comPort == INVALID_HANDLE_VALUE) {
144:         TRACE(_T("Serial port is not open\n"));
145:         return (0);
146:     }
147:     buf = ch;
148:     WriteFile (m_comPort, &buf, 1, &cBytesW, 0);
149:     if (cBytesW == 0 ) {
150:         TRACE(_T("Error writing serial port\n"));
151:         return (0);
152:     }
153:     return (1);
154: }
```

#### C.4. SettingsDlg.cpp

```
1: // SettingsDlg.cpp : implementation file
2: //
3: #include "stdafx.h"
4: #include "resource.h"
5: #include "SettingsDlg.h"
6:
7: #ifdef _DEBUG
8: #define new DEBUG_NEW
9: #undef THIS_FILE
10: static char THIS_FILE[] = __FILE__;
11: #endif
12:
13: ////////////////////////////////////////////////////
14: // SettingsDlg dialog
15:
16:
17: SettingsDlg::SettingsDlg(CWnd* pParent /*=NULL*/)
18:     : CDialog(SettingsDlg::IDD, pParent)
19: {
20:    //{{AFX_DATA_INIT(SettingsDlg)
21:     m_pre = 0.0;
22:     m_xgrid = 0.0;
23:     m_ygrid = 0.0;
24:     //}}AFX_DATA_INIT
```

```
25: }
26:
27:
28: void SettingsDlg::DoDataExchange (CDataExchange* pDX)
29: {
30:     CDialog::DoDataExchange (pDX);
31:     //{{AFX_DATA_MAP (SettingsDlg)
32:     DDX_Text (pDX, IDC_PRE, m_pre);
33:     DDV_MinMaxDouble (pDX, m_pre, 0., 100.);
34:     DDX_Text (pDX, IDC_XGRID, m_xgrid);
35:     DDV_MinMaxDouble (pDX, m_xgrid, 0., 100.);
36:     DDX_Text (pDX, IDC_YGRID, m_ygrid);
37:     DDV_MinMaxDouble (pDX, m_ygrid, 0., 100.);
38:     //}}AFX_DATA_MAP
39: }
40:
41:
42: BEGIN_MESSAGE_MAP (SettingsDlg, CDialog)
43:     //{{AFX_MSG_MAP (SettingsDlg)
44:     ON_BN_CLICKED (IDC_HPLAB, OnHplab)
45:     ON_BN_CLICKED (IDC_WSHED, OnWshed)
46:     //}}AFX_MSG_MAP
47: END_MESSAGE_MAP ()
48:
49: ////////////////////////////////////
50: // SettingsDlg message handlers
51:
52: void SettingsDlg::OnOK ()
```

```
53: {  
54:     UpdateData ();  
55:  
56:     CDialog::OnOK ();  
57: }  
58:  
59: void SettingsDlg::OnHplab()  
60: {  
61:  
62: }  
63:  
64: void SettingsDlg::OnWshed()  
65: {  
66:  
67: }
```

## **C.5.** *StdAfx.cpp*

```
1: // stdafx.cpp : source file that includes just the standard includes
2: //    USTest.pch will be the pre-compiled header
3: //    stdafx.obj will contain the pre-compiled type information
4:
5: #include "stdafx.h"
```

**C.6.** *UStest.cpp*

```
1: // UStest.cpp : Defines the class behaviors for the application.
2: //
3:
4: #include "stdafx.h"
5: #include "UStest.h"
6: #include "UStestDlg.h"
7:
8: #ifdef _DEBUG
9: #define new DEBUG_NEW
10: #undef THIS_FILE
11: static char THIS_FILE[] = __FILE__;
12: #endif
13:
14: //////////////////////////////////////
15: // CUSTestApp
16:
17: BEGIN_MESSAGE_MAP(CUSTestApp, CWinApp)
18:    //{{AFX_MSG_MAP(CUSTestApp)
19:         // NOTE - the ClassWizard will add and remove mapping macros here.
20:         // DO NOT EDIT what you see in these blocks of generated code!
21:    //}}AFX_MSG
22: END_MESSAGE_MAP()
23:
24: //////////////////////////////////////
```

```
25: // CUSTestApp construction
26:
27: CUSTestApp::CUSTestApp()
28:     : CWinApp()
29: {
30:     // TODO: add construction code here,
31:     // Place all significant initialization in InitInstance
32: }
33:
34: ///////////////////////////////////////////////////////////////////
35: // The one and only CUSTestApp object
36:
37: CUSTestApp theApp;
38:
39: ///////////////////////////////////////////////////////////////////
40: // CUSTestApp initialization
41:
42: BOOL CUSTestApp::InitInstance()
43: {
44:     // Standard initialization
45:     // If you are not using these features and wish to reduce the size
46:     // of your final executable, you should remove from the following
47:     // the specific initialization routines you do not need.
48:
49:     CUSTestDlg dlg;
50:     m_pMainWnd = &dlg;
51:     int nResponse = dlg.DoModal();
52:     if (nResponse == IDOK)
```

```
53:     {
54:         // TODO: Place code here to handle when the dialog is
55:         // dismissed with OK
56:         //AfxMessageBox(_T("Closing....."));
57:         AfxEndThread(0, true);
58:     }
59:     else if (nResponse == IDCANCEL)
60:     {
61:         // TODO: Place code here to handle when the dialog is
62:         // dismissed with Cancel
63:     }
64:
65:     // Since the dialog has been closed, return FALSE so that we exit the
66:     // application, rather than start the application's message pump.
67:     return FALSE;
68: }
```

**C.7.** *UStestDlg.cpp*

```
// UStestDlg.cpp : implementation file
//

#include "stdafx.h"
#include "UStest.h"
#include "UStestDlg.h"
#include "Serial.h"

#define REVERB_NUMTAPS    7
#define OUTPUT_RATE 22050
#define PI 3.14159

#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

////////////////////////////////////
// CUSTestDlg dialog
```

```
CUSTestDlg::CUSTestDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CUSTestDlg::IDD, pParent)
{
   //{{AFX_DATA_INIT(CUSTestDlg)
        // NOTE: the ClassWizard will add member initialization here
   //}}AFX_DATA_INIT
    // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CUSTestDlg::DoDataExchange(CDataExchange* pDX)
{
    CDialog::DoDataExchange(pDX);
   //{{AFX_DATA_MAP(CUSTestDlg)
        // NOTE: the ClassWizard will add DDX and DDV calls here
   //}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CUSTestDlg, CDialog)
   //{{AFX_MSG_MAP(CUSTestDlg)
        ON_WM_PAINT()
        ON_BN_CLICKED(IDC_SETTINGS, OnSettings)
        ON_EN_CHANGE(IDC_TIME4, OnChangeTime3)
        ON_EN_CHANGE(IDC_TIME6, OnChangeTime4)
        ON_BN_CLICKED(IDC_BUTTON1, OnButton1)
        ON_BN_CLICKED(IDC_BUTTON3, OnSetBreak)
        ON_BN_CLICKED(IDC_BUTTON2, OnClearBreak)
   //}}AFX_MSG_MAP

```

```

    ON_BN_CLICKED(IDC_BUTTON4, OnStart)
    ON_BN_CLICKED(IDC_BUTTON5, OnStop)
    ON_BN_CLICKED(IDC_BUTTON6, OnButton6)
    //}}AFX_MSG_MAP
    ON_REGISTERED_MESSAGE(UWM_MB_LOCATION_CHANGED, OnLocationChanged)
END_MESSAGE_MAP ()

////////////////////////////////////
// CUSTestDlg message handlers

BOOL CUSTestDlg::OnInitDialog()
{
    CString str;
    long height, width;

    CDialog::OnInitDialog();

    // Set the icon for this dialog. The framework does this automatically
    // when the application's main window is not a dialog
    SetIcon(m_hIcon, TRUE);           // Set big icon
    SetIcon(m_hIcon, FALSE);        // Set small icon

    CenterWindow(GetDesktopWindow()); // center to the hpc screen

    // Set up ultrasound location parameters
    //m_settingsDlg.m_xgrid = GRID;
    //m_settingsDlg.m_ygrid = GRID;
    //m_settingsDlg.m_pre  = HP_PRE;

```

```
//str.Format (_T("%.2f"), m_settingsDlg.m_xgrid);
//SetDlgItemText (IDC_XGRID, str);

//str.Format (_T("%.2f"), m_settingsDlg.m_ygrid);
//SetDlgItemText (IDC_YGRID, str);

//str.Format (_T("%.2f"), m_settingsDlg.m_pre);
//SetDlgItemText (IDC_PRE, str);

// Set up size of ultrasound space
m_boundingBox.top = 900;
m_boundingBox.left = -450;
m_boundingBox.bottom = -900;
m_boundingBox.right = 450;

width = m_boundingBox.right - m_boundingBox.left;
height = m_boundingBox.top - m_boundingBox.bottom;

// Scale to fit it all onto the screen
if (width >= height) {
    m_scalar = width / CANVAS_WIDTH;
    if (width % m_scalar > 0 ) {
        m_scalar++;
    }
}
else {
    m_scalar = height / CANVAS_HEIGHT;
```

```
        if (height % m_scalar > 0 ) {
            m_scalar++;
        }
    }

// Compute size of rectangular canvas
m_canvas.left = BORDER;
    m_canvas.top = BORDER;
    m_canvas.right = BORDER + (width / m_scalar);
    m_canvas.bottom = BORDER + (height / m_scalar);

// Actual lines in use (initially none)
//m_xpos = -1;
//m_ypos = -1;

//m_prev_xpos = -1;
//m_prev_ypos = -1;

    // Context
    m_context.setListener(m_hWnd);
    m_context.setListenerReady();
    m_sensor.setContext (&m_context);

    // Show initial parameters
//SetDlgItemInt (IDC_PINGNO, m_context.pingno());
//SetDlgItemInt (IDC_XCOORD, m_context.location().x);
```

```
//SetDlgItemInt (IDC_YCOORD, m_context.location().y);
//SetDlgItemInt (IDC_TIME1, m_context.time(1));
//SetDlgItemInt (IDC_TIME2, m_context.time(2));
//SetDlgItemInt (IDC_TIME3, m_context.time(3));
//SetDlgItemInt (IDC_TIME4, m_context.time(4));
//SetDlgItemInt (IDC_TIME5, m_context.time(5));
//SetDlgItemInt (IDC_TIME6, m_context.time(6));
//SetDlgItemInt (IDC_TIME7, m_context.time(7));
//SetDlgItemInt (IDC_TIME8, m_context.time(8));
//SetDlgItemInt (IDC_NTX, m_context.numtx());
//SetDlgItemInt (IDC_NEAREST, m_context.nearest());
//SetDlgItemText (IDC_TXID, m_context.txid());

return TRUE; // return TRUE unless you set the focus to a control
}

////////////////////////////////////
// OnLocationChanged
////////////////////////////////////

void CUSTestDlg::OnLocationChanged(WPARAM wParam, LPARAM lParam)
{
```

```
// wParam and lParam are unused
CPoint screenLoc;
CPoint location;
CPoint p1, p2;
int hdg;
int raw[3];
CString hex;

// Display and log coordinates
//location = m_context.location();
//nearest = m_context.nearest();
hdg = m_context.heading();
m_context.raw(raw);

//SetDlgItemInt (IDC_PINGNO, m_context.pingno());
//SetDlgItemInt (IDC_XCOORD, location.x);
//SetDlgItemInt (IDC_YCOORD, location.y);
//SetDlgItemInt (IDC_TIME1, m_context.time(1));
//SetDlgItemInt (IDC_TIME2, m_context.time(2));
//SetDlgItemInt (IDC_TIME3, m_context.time(3));
//SetDlgItemInt (IDC_TIME4, m_context.time(4));
//SetDlgItemInt (IDC_TIME5, m_context.time(5));
//SetDlgItemInt (IDC_TIME6, m_context.time(6));
//SetDlgItemInt (IDC_TIME7, m_context.time(7));
//SetDlgItemInt (IDC_TIME8, m_context.time(8));

SetDlgItemInt (IDC_HEADING, hdg);
```

```

//SetDlgItemInt (IDC_BYTE1, raw[0]);
//SetDlgItemInt (IDC_BYTE2, raw[1]);
//SetDlgItemInt (IDC_BYTE3, raw[2]);

hex.Format (_T("%02x"), toupper(raw[0]));
SetDlgItemText (IDC_BYTE1, hex);

hex.Format (_T("%02x"), toupper(raw[1]));
SetDlgItemText (IDC_BYTE2, hex);

hex.Format (_T("%02x"), toupper(raw[2]));
SetDlgItemText (IDC_BYTE3, hex);

// Tell context that we are ready to receive another message
m_context.setListenerReady();
}

////////////////////////////////////
// OnPaint
////////////////////////////////////

void CUSTestDlg::OnPaint ()
{
#ifdef BAGOFSHITE
    int i;
    CPoint p, p1, p2;
    CPoint location;
    CPen *pOldPen;

```

```

CPen penWide(PS_SOLID, 3, RGB(0, 0, 0));

CBrush brushGray(RGB(196, 196, 196));
CBrush brushRed(RGB(255, 0, 0));
CBrush brushYellow(RGB(255, 255, 0));
CBrush brushGreen(RGB(0, 255, 0));
CBrush brushCyan(RGB(0, 255, 255));

// EnterCriticalSection(&m_lock);
#endif
    // TRACE(_T("Paint\n"));
    CPaintDC dc(this); // device context for painting
    CBrush *pOldBrush;
    CBrush brushWhite(RGB(255, 255, 255));

    // Draw space
    pOldBrush = dc.SelectObject(&brushWhite);
    //dc.Rectangle(m_canvas);
    dc.SelectObject(pOldBrush);

#ifdef BAGOFSHITE
    // Draw transducers
    for (i = 0; i < 8; i++) {
    switch (m_txs[i]) {
    case TX_TOO FAR:

```

```

        pOldBrush = dc.SelectObject (&brushRed);
        break;
    case TX_INRANGE:
        pOldBrush = dc.SelectObject (&brushGreen);
        break;
    case TX_INUSE:
        pOldBrush = dc.SelectObject (&brushGray);
        break;
    case TX_NEAREST:
        pOldBrush = dc.SelectObject (&brushYellow);
        break;
}

p = convertPointToScreen(m_tx[i]);
    dc.Ellipse(p.x - TX_SIZE/2, p.y - TX_SIZE/2, p.x + TX_SIZE/2, p.y + TX_SIZE/2);

        dc.SelectObject (pOldBrush);
    }

pOldPen = dc.SelectObject (&penWide);

// Draw positioning baselines
if (m_xpos >= 0) {
    p1 = convertPointToScreen(m_xbase[m_xpos].p1);
    p2 = convertPointToScreen(m_xbase[m_xpos].p2);
    dc.MoveTo (p1);
    dc.LineTo (p2);
}

```

```

if (m_ypos >= 0) {
    p1 = convertPointToScreen(m_ybase[m_ypos].p1);
    p2 = convertPointToScreen(m_ybase[m_ypos].p2);
    dc.MoveTo(p1);
    dc.LineTo(p2);
}

dc.SelectObject(pOldPen);

    // Draw our location
    pOldBrush = dc.SelectObject(&brushCyan);
    location = m_context.location();
    p = convertPointToScreen(location);
    dc.Ellipse(p.x - DOT_SIZE/2, p.y - DOT_SIZE/2, p.x + DOT_SIZE/2, p.y + DOT_SIZE/2);
    dc.SelectObject(pOldBrush);
    m_lastLoc = location;

    // LeaveCriticalSection(&m_lock);

    // Do not call CDialog::OnPaint() for painting messages
#endif
}

CPoint CUSTestDlg::convertPointToScreen (int x, int y)
{
    // Invert Y to conform to screen coordinates

```

```
    CPoint p;
    int height;

    height = m_boundingBox.top - m_boundingBox.bottom;
    y = -y;

    p.x = ((x - m_boundingBox.left) / m_scalar) + BORDER;
    p.y = ((y - m_boundingBox.bottom) / m_scalar) + BORDER;

    return (p);
}

CPoint CUSTestDlg::convertPointToScreen (CPoint p)
{
    return (convertPointToScreen (p.x, p.y));
}

void CUSTestDlg::OnSettings ()
{
    CString msg;

    m_sensor.getGrid (&m_settingsDlg.m_xgrid, &m_settingsDlg.m_ygrid, &m_settingsDlg.m_pre);

    if (m_settingsDlg.DoModal () == IDOK) {
        // msg.Format(_T("This is what I got = %f"), m_settingsDlg.m_xgrid);
    }
}
```

```
// MessageBox(msg);

m_sensor.setGrid (m_settingsDlg.m_xgrid, m_settingsDlg.m_ygrid, m_settingsDlg.m_pre);
msg.Format (_T("%.2f"), m_settingsDlg.m_xgrid);
SetDlgItemText(IDC_XGRID, msg);
msg.Format (_T("%.2f"), m_settingsDlg.m_ygrid);
SetDlgItemText(IDC_YGRID, msg);
msg.Format (_T("%.2f"), m_settingsDlg.m_pre);
SetDlgItemText(IDC_PRE, msg);
    }
}

void CUSTestDlg::OnChangeTime3()
{
    // TODO: If this is a RICHEDIT control, the control will not
    // send this notification unless you override the CDialog::OnInitDialog()
    // function to send the EM_SETEVENTMASK message to the control
    // with the ENM_CHANGE flag ORed into the lParam mask.

    // TODO: Add your control notification handler code here

}

void CUSTestDlg::OnChangeTime4()
{
    // TODO: If this is a RICHEDIT control, the control will not
    // send this notification unless you override the CDialog::OnInitDialog()
    // function to send the EM_SETEVENTMASK message to the control
```

```
        // with the ENM_CHANGE flag ORed into the lParam mask.

        // TODO: Add your control notification handler code here

    }

    void CUSTestDlg::OnButton1 ()
    {
        // TODO: Add your control notification handler code here

        m_sensor.Trigger();

    }

    void CUSTestDlg::OnSetBreak ()
    {
        m_sensor.SerialBreak(true);

    }

    void CUSTestDlg::OnClearBreak ()
    {
        m_sensor.WriteData();

    }

    void CUSTestDlg::OnStart ()
```

```
{  
    m_sensor.CalibrateOn();  
}
```

```
void CUSTestDlg::OnStop()  
{  
    m_sensor.CalibrateOff();  
}
```

```
void CUSTestDlg::OnButton6()  
{  
    m_sensor.ResetHDG();  
}
```

## D. Project Timetable

This timetable gives a very rough indication of the time frame of the tasks undertaken throughout the course of this study.

<b>June</b>	Implement 1 <sup>st</sup> attempt at test harness
	Attempt to incorporate HRTF algorithm, and reconfigure sampling rates etc
<b>July</b>	Investigate other HRTF mechanisms
	Source appropriate compass
	Write compass software driver
	Investigate compass refusal to work
<b>August</b>	Source alternative compass
	Write another compass driver
	Data collection
	Data analysis
<b>September</b>	Start documentation
	Finish documentation

## **E. Thanks and Acknowledgements**

At this stage, I would like to say a very big thanks to the following people who helped me throughout the course of this project.

***Karen Bryden** – for her help with HRTF's – even though they have not (yet) been implemented as intended.*

***Ralfe Windte** – for his email answers and HRTF source code (again, not yet used)*

***John Honnibal** – for his hours of help in the hardware lab at HP. John is The Man.*

***Cliff Randall** – From Bristol University, the extremely generous compass donor.*

***Richard Hull & Jo Reid** – Without whom none of this would have been possible – for their advice and help.*

***Richard Joiner** – My project supervisor – for his support throughout.*

***Ben Clayton** – For suffering and answering my C++ questions.*

***Mike McCarthy** – For timely and helpful responses to email questions.*

***All my participants** – for their time and unrewarded efforts.*