3D Audio Spatialization Effects for Nokia Mobile Devices:

ELEC 499 Project Report

Lucas O’Neil (0229436)
Kevin Pearson (9511167)
Ryan Fehr (0131707)
Brendan Cassidy (0226138)

Submitted

To
Dr. M McGuire

April 5, 2007

in partial fulfillment of the requirements of ELEC/CENG 499
Dr. McGuire,

Please accept the following ELEC 499 Project Report, “3D Audio Spatialization Effects for Nokia Mobile Devices”.

This report is a description and analysis of the design of a 3D audio spatialization application for the Nokia N80 smartphone. Specifically, it relates to such a project that was executed by Brendan Cassidy, Lucas O’Neil, Ryan Fehr, and Kevin Pearson during the Spring 2007 session at the University of Victoria.

You will also find the project web page online at http://web.uvic.ca/~loneil/elec499/index.htm. The webpage includes Project source code in a password protected archive for which the password will be emailed to you.

The project group would like to thank you for your time and for your guidance during the term.

Sincerely,

Brendan A. Cassidy
UVic Student No. 0226138
cassidyb@engr.uvic.ca

On behalf of Project Group 7
# Table of Contents

List of Figures ..................................................................................................................................................... iii  
Summary ................................................................................................................................................................... 1  
1.0 Introduction .......................................................................................................................................................... 2  
1.1 Purpose of Report .................................................................................................................................................. 2  
1.2 Scope of Report ..................................................................................................................................................... 2  
1.3 Background .......................................................................................................................................................... 2  
1.3.1 Audio Spatialization with Headphones ........................................................................................................... 2  
1.3.2 Nokia N80 ........................................................................................................................................................ 4  
1.3.3 Symbian S60 .................................................................................................................................................... 4  
1.3.4 Carbide C++ IDE .......................................................................................................................................... 7  
2.0 Problem Statement .............................................................................................................................................. 8  
3.0 Solutions ............................................................................................................................................................. 10  
3.1 Fundamental Approach to Solution ..................................................................................................................... 10  
3.1.1 Hardware Approach ....................................................................................................................................... 10  
3.1.2 Software Approach ....................................................................................................................................... 10  
3.1.3 Mixed Approach .......................................................................................................................................... 10  
3.1.4 Chosen Approach ........................................................................................................................................ 10  
3.2 Desirable Features ............................................................................................................................................. 11  
3.2.1 Real-time Rendering and Playback ................................................................................................................. 11  
3.2.2 Background Result Store ............................................................................................................................. 11  
3.2.3 Multiple Format Support ........................................................................................................................... 12  
3.2.4 Multiple Spatialization Effects ...................................................................................................................... 12  
4.0 Problems and Pitfalls ........................................................................................................................................... 13  
4.1 Audio Output Streaming ...................................................................................................................................... 13  
4.2 Audio Format Conversion ................................................................................................................................. 13  
4.3 Frequency Domain Filtering ........................................................................................................................... 13  
4.4 Blockwise Processing ...................................................................................................................................... 14  
5.0 Implemented Solution ....................................................................................................................................... 15  
5.1 Spatialization Effect ......................................................................................................................................... 15  
5.2 Filter Implementation ...................................................................................................................................... 16  
5.3 User Interface .................................................................................................................................................... 16  
5.3.1 GUI ................................................................................................................................................................. 17  
5.3.2 File I/O .......................................................................................................................................................... 18  
6.0 Results ................................................................................................................................................................. 19  
6.1 Audio Effect Results ........................................................................................................................................... 19  
6.2 Performance Results ......................................................................................................................................... 19  
6.3 Awards and Accolades ..................................................................................................................................... 19  
7.0 Future Extensions ............................................................................................................................................... 20  
7.1 Input Format Support ........................................................................................................................................ 20  
7.2 Frequency Domain Filtering .......................................................................................................................... 20  
7.3 Dynamic File Specification ............................................................................................................................ 20  
7.4 Output Buffering ............................................................................................................................................... 20
<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Audio externalization and spatialization</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Nokia N80 smartphone</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>S60 High level architecture from [3]</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Screen capture of Symbian S60 emulator in Windows</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>Carbide development tool screen shot</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>Stereo over loudspeaker effect</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>Stereo over loudspeaker block diagram implementation</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>Application GUI</td>
<td>18</td>
</tr>
<tr>
<td>9</td>
<td>Simulated 5.1 surround sound conceivable implementation</td>
<td>22</td>
</tr>
</tbody>
</table>
Summary

The use of mobile devices as personal media players has become increasingly prominent. Cell phones are now being used as portable audio and video players, eliminating the need for separate devices (i.e. A cell phone and an mp3 player). As cell phones' media capabilities are becoming more important in their marketability, the overall enjoyment of the mobile listening experience should be optimized.

The quality of audio can be subjectively improved with headphone spatialization techniques. Achieving headphone externalization can be accomplished through the use of filters that model the physical and psychoacoustic cues that we use to perceive directionality in sound. The main cues used to discern the angle of attack from around the listener as well as the elevation angle are the head shadowing effect. The filtering method implemented in this project was time domain convolution. The Head-Related Transfer Function (HRTF) used to model the physical cues was obtained in a Head-Related Impulse Response (HRIR) form.

A 3D audio spatialization application was developed for the Nokia N80 smartphone, a device which runs on the Symbian S60 operating system platform. Development of a Symbian S60 application requires the S60 3rd Edition SDK, and Carbide.C++. The S60 SDK provides an emulator for debugging S60 applications on a PC, which was helpful for the design and debugging process in conjunction with an on-device debugger.

The 3D spatialization effect that was implemented in Symbian C++ was a simulated stereo-over-loudspeakers effect. This effect simulates a listening environment in which two loudspeakers are behind the listener’s head, at angles of ±110°. The speaker to the left of (and behind) the listener’s head would play the left stereo channel; similarly the speaker to the right would play the right channel.

During the formal project presentations on March 30, 2007, the project was recognized with two awards: the Schneider Electric Relevant Innovation Award and third place in the IEEE award.

Although the implemented result did successfully apply the desired 3D spatialization effect, many improvements and expansions would be made with sufficient time and resources.

This project provided a valuable experience to the design team, especially with respect to the unique requirements of developing quality applications for embedded systems.
1.0 Introduction

1.1 Purpose of Report

This report is a description and analysis of the design of a 3D audio spatialization application for the Nokia N80 smartphone. Specifically, it relates to such a project that was executed by Brendan Cassidy, Lucas O’Neil, Ryan Fehr, and Kevin Pearson while completing their 4th year design project at the University Of Victoria.

The report will cover the following topics:

- Description of the Nokia N80 smartphone
- Project background
- Design approach to audio filtering
- Decisions made during the above processes and the justification and benefits of such decisions
- Problems and pitfalls encountered during development
- Results following the implementation of the software
- Possible opportunities for expansion of the project

1.2 Scope of Report

This report will focus on an engineering approach to developing and implementing an application for the Symbian S60 operating system running on the Nokia N80 smartphone. That is, design decisions will be presented and will be justified with respect to the satisfaction of requirements, design simplification, platform restrictions, time restrictions, and/or other factors influencing decisions.

The scope of this report will be primarily restricted to technical aspects of the project, focusing primarily on the filtering background and design.

1.3 Background

1.3.1 Audio Spatialization with Headphones

Achieving headphone externalization can be accomplished through the use of filters that model the physical and psychoacoustic cues that we use to perceive directionality in sound. The main cues used to discern the angle of attack from around the listener as well as the elevation angle are the head shadowing effect, the reflections off the shoulders and upper torso, and the reflections from the ridges of the ear and the outer shape of the ear (called the pinna). The head shadow refers to the attenuation caused by the sound waves coming from one side of the head bending around the head before reaching the other ear. The difference in signal between the on axis ear and the ear that receives the sound waves after the head shadow is used by the brain to discern the direction of the sound. Another of the auditory cues is the shoulder-torso system reflections. A sound source in different
positions around the head or at different angles of elevation will have differing reflections off of this system which, when reaching the ear are used to determine information about the spatial position of the sound. Another reflection based cue is the pinna system. The ridges in the ear and the shape of the outer ear itself will cause different reflections based on the position of the sound.

Taking these cues into account, one can create a filter system to recreate them with amplitude and phase responses (and delay) to simulate the results. A sound source can be perceived to be coming from a certain position by applying the relevant filter to the left and right headphone channels that would recreate the physical system response if the sound source were in that position in the real spatial environment. An application used in this project was to take a stereo music file and move the sound source from the lateralized position right at the ears to an externalized position simulating two loudspeakers behind the head. The reason speakers behind the head were used is that when the brain perceives a sound from within the visual field but can't see where the sound is coming from, there can be more auditory confusion when the signal is processed by the brain. When the sound is made to appear as if behind the head the externalization effect becomes clearer as there is no visual confusion.

A visualization of audio externalization and spatialization is presented in Figure 1 below.

The filtering method implemented in this project was time domain convolution. The Head-Related Transfer Function (HRTF) used to model the physical cues was obtained in a Head-Related Impulse Response (HRIR) form. The HRIRs were obtain from [KEMAR], a study done at the MIT Media Laboratory. The impulse response measurements were obtained by using the accurately constructed KEMAR dummy that appropriately models the head-shoulder-torso-pinna system. Microphones placed inside the ear canal pick up the results of frequency sweeps done at different positions around the dummy to give accurate HRIRs. [1]
1.3.2 Nokia N80

The Nokia N80 was introduced with the second wave of handsets in November 2005. The N80 is also a part of the Nokia XpressMusic brand this allows the phone to play store and upload media the on board integrated digital media player supports various file types including: MP3, AAC, M4A, eAAC+, and WMA. In addition to these file types the handset also supports (Global System for Mobile Communications) GSM (Enhanced Data GSM Environment) EDGE and 3G (Wideband Code Division Multiple Access) WCDMA. This support for these high-bandwidth networks allows the N80 to stream video and music. The N80 is built on the Symbian OS (9.1) with Series 60 version 3; this operating system along with open source development tool Carbide enables the users to develop software for this handset. More information about the Symbian OS and Carbide can be found in the following sections. [2]

1.3.3 Symbian S60

The Nokia N80 uses Symbian OS S60, which is the global industry standard for smartphones. This OS is a multi-runtime platform supporting C++. A rich set of APIs is available for the S60 platform, making development of applications easier and faster. The S60 development community works with different open source communities on different projects.

S60 Platform Architecture

Figure 3 illustrates the high level architecture of the S60 platform. It is based on the Symbian OS but also provides additional features.
• UI Layer – contains all software pertaining to the user interface and applications.
• Application Engine Layer – provides access to data that is required by the application.
• System Layer – provides core functionality of the system, usually in the form of system servers.
• Kernel Layer – software for providing kernel services (process and thread connection).
• Hardware Adaptation Layer – software that integrates higher level applications to the low level hardware, as these are independent of each other.
• Hardware – The actual physical hardware on which the software will run. [3]

**Symbian S60 Application Development Requirements**

Development of a Symbian S60 application requires the S60 3rd Edition SDK, and Carbide.C++. The S60 SDK contains an emulator which provides emulation of the S60 applications on a PC. The emulator runs in a single Windows process called. It provides a graphical interface of a real phone and phone functionality to test your application. Figure 4 illustrates how the emulator appears on the PC.
Although the emulator is an extensive tool for application testing, its functionality differs from the devices. This is because the devices vary and some features are device specific, like memory capacity. This can cause problems for functionality. Functionality differences include:

- Because the emulator uses threads and not processes like the device would use, bad pointers can exist
- The emulator uses default or set up heap and stack sizes, which are device-specific in the actual devices
- The emulator has a whole memory capacity of the test PC, but the device has the limited amount of embedded memory
- The processor capacity of a PC far exceeds the processor capacity of a target device. This can make big differences on actual performance of an application
- The emulator can be run without platform security, which is not possible in mobile devices based on the latest version of Symbian (Symbian OS v9.1) [3]
1.3.4 Carbide C++ IDE

The development of new smart phones created a large demand for more mobile applications. Recognizing this demand Nokia launched a new set of development tools called Carbide. This set of development tools greatly improved creation of applications for mobile devices.

Carbide was developed in conjunction with the Symbian operating system as described in previous sections. Symbian is the leading operating system for smart phones. Carbide is an attractive development tool with its framework being based off the open source Eclipse Integrated Development Tool.

The Eclipse (IDE) is an open source project with industry wide support originally developed by IBM. Eclipse is rapidly becoming the industry standard for application development.

The carbide development tools work with the SDK platform in this project will deal specifically with the S60 platform that is used for Nokia N80.

![Carbide development tool screen shot](image)

Figure 5: Carbide development tool screen shot
2.0 Problem Statement

Mobile devices today are increasingly becoming more and more reliant on multimedia applications as a selling point. Particularly, the ability to playback audio and video files is becoming a key aspect in cellular phones and other portable devices. As such, optimizing the listening or viewing experience for the user, delivering a higher quality or more immersive playback environment will have a large market potential. Filtering techniques can be used to enhance the audio playback for different formats of audio files as well as the sound streams from the video files being used. The goal of this project was to implement digital filtering techniques for use on cellular phones, specifically the Nokia N80 which is based on the Symbian operating system. Symbian is an open industry standard operating system for data enabled mobile phones with obtainable development support and tools.

The technique implemented in this project is headphone sound externalization. Since a large amount of the use of mobile devices is listening to their output over headphones, providing an option for improving the perceptive quality of listening over headphones is desirable. When listening to audio over headphones, the sound can become localized inside the head causing the sound field to appear flat and lacking dimensional qualities. This lack of spatialization can be awkward to listen to and for long term use cause listening fatigue. This 'in-head' localization, referred to as lateralization, is caused by the lack of normal spatial cues that the human brain uses to place the sound source in the listening environment. For recordings that have been intended to listen to on a speaker system this is especially noticeable.

To fix this issue of lateralization a method of achieving externalization, where the sound source is localized outside of the head, is implemented. In headphone sound externalization, processing is applied to perceptually move the sound to a location away from the axis between the two ears to away from the head. Placing the audio source in an externalized environment gives the user a sensation of spatial area that is more pleasant to listen to than the point source sensation at the ears occurs when the sound is lateralized. The format of the externalization implemented on the phone hardware was to place the sound source as if it were occurring from a pair of speakers spaced apart behind the head. There are filtering techniques that are able to take a sound source and, using principles of psychoacoustics and physical modeling, place that source at a specified angle (azimuth and elevation) and radius. The goal will be to experiment with differing parameters for these filtering techniques as well as methods of implementation to find the best way to achieve the results on the hardware of the phone within the limitations of the operating system.

Another potential application of the headphone externalization techniques is for viewing movies in surround sound. Since most mobile devices primarily use headphones for their audio output, movies viewed on the device could definitely benefit from headphone externalization and spatialization. Movies for home release on DVD almost always
include a 5.1 surround sound audio track for home theater applications. The filter
techniques used can be applied to place the perceived sound sources in an array of points
around the listener, effectively creating a surround sound effect through headphones. If
storage space permits, the 5 individual surround tracks could be included with the movie
and then externalized to their respective positions during playback. Otherwise, the
surround track could be mixed down to stereo and then using sophisticated upmixing
techniques re-separated and then externalized.

The Nokia N80 runs on the open standard Symbian operating system which has
development tools and support available on the web. The Application Programming
Interface (API) is publicly documented, allowing for third party application
development. The goal of this project was to implement some form of the audio
externalization techniques on a sample N80 handset acquired. The idea for the
functionality of the phone as a media player device is for the user to be able to store an
audio file of some type and, after converting the file to pulse code modulated audio data
if necessary, play it back with the externalization applied. Whether the phone could
perform this filtering in real time and play back without any delay would depend on the
capabilities of the development environment as well as the optimization of the software
routines. If the time taken to filter would work out to be less than the length of the song
then a buffering technique could be implemented where the procedure divides the file up
into frames and processes them in a block wise manner. After completing a suitable
number of frames, the audio would start playing back the processed frames while the
program works on the remaining ones.

This project entailed obtaining the development tools for Symbian based mobile devices
and learning the necessary libraries and methods for handling audio data on the device
and how to accomplish the filtering techniques. The filters themselves were to be
implemented in MATLAB in order to find the best parameters and audio quality and to
facilitate listening tests. The MATLAB functions written could be ported to the C++
development environment for Symbian.
3.0 Solutions

In order to implement a solution to the problem at hand, a hardware-based, software-based or mixed approach could be taken. Each approach could have its relative advantages and disadvantages.

3.1 Fundamental Approach to Solution

3.1.1 Hardware Approach

A hardware approach to the problem would be beneficial for a few reasons, primarily revolving around performance concerns. As the use of high-order FIR filters or smaller, but still significant IIR filters would be required, a hardware solution could be beneficial in that generally speaking, hardware filters will give better performance results than software implementations.

Additionally, a hardware-based solution would be much more deterministic in its performance. That is, the ability to render audio effects in real time would be consistent, whereas non-real-time operating systems would be dependant on other load on the system and may behave non-deterministically.

Unfortunately, access to a mobile phone’s hardware is greatly restricted and therefore a hardware solution was unlikely.

3.1.2 Software Approach

A software approach to the problem would have the advantage in that implementation would likely be much simpler, and that a software solution would be portable across many devices, unlike a hardware solution. Unfortunately, performance and deterministic reliability would be sacrificed in a software solution.

3.1.3 Mixed Approach

A hybrid software/hardware approach could possibly offer advantages of both techniques. Indeed a hardware implementation would have to be some form of a hybrid approach anyways.

3.1.4 Chosen Approach

It was decided very early on that a software approach was unquestionably the most suitable method to implement a solution to the problem. The hardware of the Nokia N80 is completely closed to third party developers for many reasons, and the hardware approach was infeasible.

Additionally, the N80 uses the Symbian S60 operating system, which is an open and fully-documented platform for third party application development. Audio and multimedia libraries may provide some assistance as well. One of the most important
features is that applications for Symbian S60 should run on all devices using this and newer Symbian platforms, broadening the scope of the solution to a range of mobile devices.

Symbian S60 supports C++ and Java development. As an embedded system operating system, much effort is spent in the managing of garbage collection and other high-level concepts. Symbian C++ is in essence a unique flavor of the standard programming language, with barriers in place to use standard libraries and with strict coding conventions. Although these features fundamentally make Symbian a robust and reliable platform, they could also cause many pitfalls for the design team of electrical and computer engineering students, with a much more low-level focus to design experience.

Indeed audio effect applications are much more suited to low-level design environments, and the higher level Symbian platform provided barriers to the project implementation, narrowing the scope of the solution from the original intention in order to meet scheduled project deadlines.

### 3.2 Desirable Features

#### 3.2.1 Real-time Rendering and Playback

One feature that an ideal solution would include is the ability to render and playback audio in a somewhat real-time manner. This feature would mean that a user would be able to enable the effect (while playing a song or perhaps watching video) and have the playback occur almost immediately (i.e. the song continues playing from the same point but with the desired spatialization effect).

To implement this technically, some form of output buffering must be available, and the rendering of effects must occur in somewhat small blocks, rather than on an entire input at once. Output buffering would involve buffering output data when it is available and playing it when needed. As long as the data can be produced before it is needed, playback would be seamless and continuous. If the data is unavailable (a buffer underflow), the playback would stop until sufficient output data is again ready.

Symbian S60 C++ supports an output buffering class as part of its multimedia framework (MMF): the CmdaAudioOutputStream class provides an “interface to an audio stream player passing audio data from specified buffers to the audio hardware” [4]. Potentially leveraging this class’ functionality could provide real time rendering, buffering and playback; this would require that the condition of rendering the data in sufficient time is met.

#### 3.2.2 Background Result Store

In addition to or in lieu of real-time playback, the spatialized audio should be able to be stored to the device’s non-volatile memory for future playback. This could be desirable if the effect greatly improved the quality of the particular song and the user did not feel that
the original song was worth keeping; the pre-rendered spatialization effects would thus be permanent and not require any processing for future playbacks.

### 3.2.3 Multiple Format Support

An ideal solution would also support multiple input audio formats. Most mobile audio player users are primarily concerned with support of only MP3s, but new lossless formats such as FLAC may also be desirable. Ideally, audio effects would be applied to decoded pulse code modulated (PCM) data, and the decoding/encoding would be taken care of outside the scope of this project.

The CmdaAudioConvertUtility class in Symbian, which is intended to “[convert] audio sample data from one format to another” [4], could be leveraged to decode any input format to a PCM representation, applying the effect to the OCM data and then playing back or reencoding for file store the result.

### 3.2.4 Multiple Spatialization Effects

An ideal solution would also provide multiple spatialization effects. Such effects could be simulated stereo loudspeakers and simulated 5.1 surround sound. Also, with a simple GUI, the user could possibly customize the simulated “speaker” placements. This would allow the user to choose an effect which they find best enhances the audio, as audio effects are very subjective in terms of observed quality.
4.0 Problems and Pitfalls

4.1 Audio Output Streaming

The CmdaAudioOutputStream class enables the multimedia file clients to stream audio data to audio hardware buffers and to specify priority of the audio output stream in relation to other clients. This class provides options for changing the balance or volume, and to play or stop the audio track. The callback functions, documented in detail in [4], were not working as anticipated and use of the on-phone debugger did not help to identify the problem. Although the project team is confident that the problem could have been overcome, when it was realized that an implementation capable of rendering the audio in real-time would not be feasible in the imposed schedule it was decided that further investigation into the matter would not benefit the project, as real-time playback was no longer within the scope of the implemented project.

4.2 Audio Format Conversion

The class CmdaAudioConvertUtility can be used to convert audio sample data from one format to another. However, this class is not supported by the 3rd version of the Symbian S60 operating system nor is it or will it be by any later versions. As a result the class could not be used in this application. The use of this class would have been useful in making the application for the phone more robust.

It is hypothesized that the CmdaAudioConvertUtility class was dropped due to the increasing trend to lock down audio decoding in the day of prevalent digital rights management (DRM), as access to decoded PCM data from any input format may be concerning to those entities who believe to have an interest in the strict enforcement of DRM.

4.3 Frequency Domain Filtering

As the spatialization effects are implemented with digital filters, computational optimization is attainable by computing the response to filters in the frequency domain rather than in the time domain. In order to obtain the frequency-domain representation of an arbitrary signal, a fast fourier transform is desirable.

An attempt was made to implement the Fast-Fourier Transform (FFT) on the N80 for frequency domain filtering. Problems arose when trying to port open source FFT libraries (in particular the FFTW available from [5]) on the Symbian S60 platform. Standard C/C++ libraries that were necessary for the open source libraries were not supported by the S60 platform. Although the project team is confident that an FFT could be implemented or ported for Symbian, frequency domain processing was dropped from the scope of the project due to time restrictions from the imposed schedule.
4.4 Blockwise Processing

Because the digital filtering was processed in blocks, rather than on the entire input at once, reconstruction of the output from the separate blocks involved a bit of trouble shooting. Initial implementations suffered from a noticeable “clicking” where two blocks met, as the output was not exactly as would have been obtained by applying filters to the entire input.

This clicking was overcome by applying the overlap-and-add technique given in [6].
5.0 Implemented Solution

5.1 Spatialization Effect
The 3D spatialization effect that was implemented in Symbian C++ was a simulated stereo-over-loudspeakers effect. This effect simulates a listening environment in which two loudspeakers are behind the listener’s head, at angles of ±110°. The speaker to the left of (and behind) the listener’s head would play the left stereo channel; similarly the speaker to the right would play the right channel.

A representation of the spatialization effect is given in Figure 6 below.

![Figure 6: Stereo over loudspeaker effect](image)

Placement of the virtual speakers were behind the listener’s head rather than in front of the head to reduce visual and aural contradiction; when a person hears something “infront” of them but sees that the source is not actually there, the brain may tend to believe visual cues over audible cues. If the source is behind the head, such conflicts will not exist.

A block diagram of the implementation of this effect is given in Figure 7 below.
The spatialization effect was applied using a time-domain FIR filter implementation. The filter was applied in blocks of 20,000 samples (40,000 bytes for 16 bit WAV encoding) simultaneously to each of the input stereo tracks (i.e. loop infusion). The filtering was implemented as nested for loops, convolving the time domain signal with the impulse response of the desired spatialization location’s head-related transfer function.

In order to overcome artifacts introduced by blockwise filtering, the overlap-and-add method in [antonieu] was implemented. In order to add the overlapping values, the required number of samples to be overlapped at the end of a block (the FIR filter length less one) were computed by effectively zero padding the block. These results were stored in a buffer and, using a similar zero padding technique for the first samples of the next block, the end of the previous block was added to the beginning of the next block. This successfully implemented the overlap-and-add method and gave results that were numerically identical to what would be obtained by processing the entire input at once.

5.3 User Interface

The end result from the user point-of-view was an application with a single menu command (“Apply Stereo 3D”) which would expect a WAV file in a predefined path with a predefined name.

The properties of the expected file were as follows:
- .WAV format
- 16 bit encoding
- Stereo (rather than Mono)
- 44100 Hz sample rate
- File name “input.wav”
- File path C:\Data\Sounds\
The output file would have the exact same format, but would be named “output3D.wav”. Once the output file was rendered, the user could listen to it in the device’s built in media player.

The application was blocking, meaning that while the phone was rendering the audio no other applications or user interfacing would be available.

Although this interface was cumbersome and impractical, the scope of the project did not include an intuitive user interface. Ideally, the input file could support multiple formats, and the path and filename would be specified dynamically.

5.3.1 GUI

The GUI that was used was a basic Symbian GUI supplied by Carbide. Essentially, the typical “Hello World” application was used as a template. Customization of the menu provided a more suitable set of menu options. When the user selected the “Apply Stereo 3D” option, rather than displaying the trivial “Hello World” message, the spatialization rendering was initiated. A message would be displayed once the rendering was complete.

The application GUI is depicted in Figure 8 below.
5.3.2 File I/O

File I/O was accomplished using the RFile class documented in [API]. Much like standard c file I/O functions fread and fwrite, the file was incrementally read directly into a 40KB buffer. To obtain the PCM audio data from a WAV file, the 44B header simply needs to be stripped away, leaving little-endian 4B words representing 2 simultaneous 16bit samples (one each for stereo left and right).
6.0 Results

6.1 Audio Effect Results
The audio effects generated by the emulator were spatialized consistent with what was observed in pre-implementation MATLAB simulations. The resulting filtered audio created the effect of a loud speaker behind the right shoulder of the listener and another loudspeaker behind their left shoulder as was expected.

6.2 Performance Results
The method used for filtering was not very efficient. This was due to the restrictions of the software-based implementation and of the time-domain filter implementation. For a 30 second audio WAV file the program took approximately one minute to apply the audio effects desired. For a larger file this computational time was even more pronounced.

6.3 Awards and Accolades
During the formal project presentations on March 30, 2007, the project was recognized with two awards: the Schneider Electric Relevant Innovation Award for the student design team which best demonstrates an understanding of market needs by applying technology to deliver a quality solution and third place in the IEEE award based on the poster presentation.
Although the implemented result did successfully apply the desired 3D spatialization effect, many improvements and expansions would be made with sufficient time and resources. This section outlines some of the key features that should be considered for future extensions of the project.

### 7.1 Input Format Support

The implemented application only allowed for a very specific input file format, due to limitations that were encountered while working with the CmdaAudioConvertUtility class as previously discussed. The WAV format is impractical for a few reasons, with the most significant reason being the storage requirements to store a library of WAV files. WAV files are uncompressed and thus are very large in size, with an average song ranging in size from 60 to 100 MB (while MP3s average around a measly 3 to 5 MB each). The WAV format was a good academic starting point, but a practical application would require support of MP3s at a minimum.

### 7.2 Frequency Domain Filtering

To improve upon the execution time of applying the spatialization effects, a frequency domain implementation of the FIR filters (the HRIRs) would be required. The current implementation is a time-domain convolution one, which is $O(n^2)$. An equivalent result can be obtained with a $O(n)$ multiplication in the frequency domain, with a $O(n\log(n))$ FFT required to obtain the frequency domain representation of the input (the filter transfer function could be statically programmed, saving unnecessary computation). A frequency-based implementation ranks high on the priority list of improvements that should be made in future project extensions.

### 7.3 Dynamic File Specification

To have a user interface that would be practical for the application of this project, dynamic input file specification would be required. Allowing the user to dynamically specify on which file to apply the spatialization effect would be a very powerful feature yet would be relatively simple to implement.

### 7.4 Output Buffering

As previously discussed, the ability to buffer and playback the resulting spatialized audio as it is rendered would be greatly beneficial to the practical applicability of this project. If frequency domain filtering is implemented and successfully improves the execution time of the audio rendering to the point where buffer underflows could be avoided, this feature should be added.
7.5 Additional Spatialization Effects

Currently, the implementation of externalization used is a fixed position of 2 speakers at a fixed angle behind the head. It would be entirely possible in future applications to allow the user to specify the angle and radius of the perceived sound source on the fly as they are listening to the playback. As well, to provide a better externalized sound, a room modeling filtering aspect can be applied with the externalization routines to simulate the reverb and wall reflections from the physical environment. This virtual listening environment would enhance the externalization effect even more. As discussed earlier, multiple channels can be modified to provide a multi-point surround sound experience. The principles applied to headphone externalization can also be applied to loudspeaker listening if the user can specify dimensions concerning the listening environment such as the radius between the speakers and the distance to the listener. This, coupled with the ability for future phones to transmit audio over Bluetooth, could make the phone into a multi-use multimedia device that can be used to playback in any environment.

7.6 Equalization

One of the issues with the measured HRIR method of achieving externalization is that the filtering does not have a flat magnitude response. This can cause the filtered sound to lose some of its perceived quality by lacking in some frequency ranges. The reason for this is derived from the microphones used to measure the impulse response in the dummy. The microphones will not have a completely flat frequency response so the tonal coloration that they do will be passed onto the HRIR as well as the externalization effect. As well, the impulse response measurement is taken inside the ear canal and thus the effect of the audio traveling through the ear canal is passed on into the HRIR. When the sound is played back over headphones, the effect ear canal is then effectively doubled. These problems may be remedied using additional equalization filters on top of the externalization filtering. The effect of the frequency response of the microphone can be adjusted for by boosting the frequencies that are attenuated and a sophisticated model of the ear canal could be used to compensate for that. A less intensive solution, which was used for the purposes of this project, is to use the existing equalization options in the media player that is playing back the externalized audio. An EQ preset that boosts the lower frequency can compensate for some of the tonal discoloration but can't completely cancel out the loss.

7.7 Video Application

A practical application of the project would be in the spatialization of audio for video. Simulated surround sound in particular would greatly improve the experience of watching movies on mobile devices.

It is possible to obtain the audio layer of a video file programmatically; applying spatialization effects and merging the audio back in with the video would be a feasible task (however, some consideration would be required to ensure the delay resulting from
the filter does not cause a noticeable delay between video and audio – i.e. the result must be re-synchronized).

A block diagram of a possible simulated 5.1 surround sound is given in Figure 9 below.

![Figure 9: Simulated 5.1 surround sound conceivable implementation](image)

This feature would greatly expand the applicability of the project.
The simulated stereo-over-loudspeaker effect was successfully implemented for the Symbian S60 in Carbide C++. The implementation is however restricted in terms of execution time, input audio format and in the ability to render and playback the results in real-time. These issues could have been addressed with sufficient time and resources, and could perhaps be considered in future expansion of the project.

The project was considered by the group members to be a success in that an adequate level of deliverables were provided and, most importantly, a great deal about embedded system application development was learned.
9.0 References

9.1 Cited References


9.2 General References