Amplitude Analysis in Musical Composition

Timothy Opie
Creative Industries: Music and Sound
Queensland University of Technology
Australia
Email: timopie@fastmail.fm

Abstract

This paper discusses the design, function, and initial results of a set of four computer music tools. The tools are the first step in a series of attribute studies based on acoustic ecology and composition with natural sounds. The application of the tools enables abstraction, magnification, and transformation of specific attributes defining a sound event. The study endeavours to extract information from sound events that can be categorised and used in a musical composition process.

Introduction

Listening to nature, and the sounds of the environment has led many composers to write musical works. Vivaldi's *Le quattro stagioni* (*The Four Seasons*) published in 1725 is a well recognised musical work that attempts to imitate nature. Many of the musical works of the Baroque period were imitations of nature. Later pastoral music imitated nature by depicting life in the country. Ludwig van Beethoven's s*ymphony No.6, Op.68 "Pastoralî*, written in 1808 is a well known example of this genre. Today, composers such as Barry Truax use granular synthesis to recreate and explore the sounds of nature. Recently I began using the sounds of nature as a subject for analysis, through which I could create musical works. This process began as an attribute study on amplitude.

An *attribute study* is a study which categorises an event into attributes, using the attributes as a focus for analysis, synthesis, abstraction, reduction, resynthesis, and modelling. In this attribute study, sound events have been categorised. The currently identified categories of the sound event attribute study include *amplitude*, *frequency*, *duration*, *timbre*, and *space.*

The idea behind using an attribute study is to analyse naturally occurring sounds, for example a crackling fire, rain drops, and ocean waves, and use the analysis as the basis for ecologically based musical compositions. The analysis endeavours to find patterns and structures, and categorise them into a typology of behavioural affordances that can be used as compositional devices. Some currently identified behavioural affordances include *stasis*, *movement*, and *gating*. The devices will be constructed as algorithms for computer assisted composition and sound generation, as well as rule sets for composers.

Barry Truax began using granular synthesis as a device in his research of acoustic ecology and soundscapes (Truax 1991). It allowed him to stretch time and observe the inner character of sound, "as if under a microscope" (Truax 1999). The attribute study will afford the opportunity to stretch time for each attribute individually allowing a greater depth of analysis.

Amplitude was the first attribute to be studied in this attribute study, and will be the focus of this paper. The other attributes are still in preliminary stages of study, however this study of amplitude is becoming quite mature. Amplitude was cited by Clynes (1986) as an independent dimension of sound. The choice of commencing the study with amplitude resulted from two simple factors. 1) The recording and analysis of amplitude is well documented and forms the basis of many sound functions. 2) There was no framework or structure from which to work, so it would need to be built from ground up. Starting with the simpler, documented task meant it was easier to set up and verify the framework for the attribute studies.

Four programs currently constitute the amplitude attribute study:
- *ExtractAmplitude* creates a data file containing the primary analysis information. The information is gathered by extracting amplitude values from an audio file.
- *ProcessEnvelope* uses the data file to create a scaleable envelope that can be applied to any input sound source, and saved as an audio file.
- *DataMelody* uses the data file to create a musical composition that can be saved as a MIDI or audio file. The program transforms amplitude data to pitch data, creating melodies.
- *CracklingFire* has a more esoteric function and is derived directly as a secondary abstraction of the data created in the first program. *CracklingFire* is modelled on the statistical information generated from the data file. It simulates a crackling fire.

This paper will examine these programs in more detail, and form a basic analysis of the initial re-

sults, including areas in which the programs can be used and extended. The other attributes to be studied, namely frequency, duration, timbre, and space, will not be analysed or discussed at any depth within this paper. In summary though the frequency attribute will focus on the base frequency of a sound event. The duration will focus on the duration of streams within a sound event. Timbre will focus on the sound event spectrum, and activity of the spectrum and transients. Space will focus on movement and location of sound events within multi-channel sound recordings. As each attribute study matures it will be incorporated into a larger encompassing structure that will combine the attributes to give a fuller and more in depth view of each sound event. The data collected from each attribute study will use a common format allowing the data to be interchanged, stretched and transformed from one form to another without having to create conversion processes. This will allow for a greater flexibility and allow for wider experimentation with the ecological soundscape.

Ecologically Based Composition

Degeneration of the audio landscape, sound pollution, and sound intrusions in general have only in recent decades been acknowledged as a threat to our environment. R. Murray Schafer began the first studies into this area, and founded the World Soundscape Project (Truax 1999). This research project made and analysed recordings of soundscapes in Canada, making assessment on keynotes[1], sound events, and soundmarks[2]. Barry Truax was amongst those on his research team.

Truax has researched approaches to listening and analysing soundscapes, as well as composing with soundscapes. He developed the technique of composing with sound at the micro-time level in real-time. Using his PODX system, Truax composed using granular synthesis to stretch sound events, enabling the listener to hear the multitude of sounds and characteristics present within the event (Truax 1992).

Truax (1999) stated: "In terms of the soundscape composition, the added duration also allows the sound to reverberate in the listener's memory, providing time for long-term memories and associations to surface….By keeping the attack portion relatively intact and stretching only the body of the sound, each signal retained its recognizability, but allowed listening associations to be savoured, along with the inherent musicality of its constituent harmonics".

This compositional technique is apparent in his piece *Ocean*, which features the crashing of waves on a shoreline. The listener may recognise the sound event immediately, but there awaits a journey of undiscovered nuances and sounds that would otherwise be impossible to hear. Truax's CD *Pacific Rim* contains many similar examples (Truax 1992).

Analysis of perception from an ecological perspective assumes the knowledge that the environment is highly structured. Objects and events from within the environment need to be considered within that structure, how it relates directly to them, and what it affords (Windsor 2000, Gibson 1979). For example the sound of a waterfall may afford different things to every listener from any species:

- place for washing
- place of fun and play
- dangerous place
- life supportive habitat
- source of power
- food or water resource
- soundscape recording

These affordances will not change the sound event, but they will change the individual interpretation of the sound. Using natural sounds in a compositional context may set up a series referential events, which may extend the meaning or perception of the music or soundscape. One of the aims for my own compositions is that the listener treats the sound not as a sound object[3], but as a sound event, allowing personal exploration of the environment from which the sound event was derived.

Damián Keller and Barry Truax wrote a paper about their work with ecologically based granular synthesis in 1998 (Keller & Truax 1998). The method for their process consisted of sampling simple sounds such as scraping, breaking, and bouncing. They would then select fragments from their samples and use these in a granular synthesis process in order to resynthesise the entire sound. They then experimented with the control parameters of the synthesis process in order to create variants on the original sound, so as to create an array of different scraping sounds based on the

1 "Keynote sounds are those which are heard by a particular society continuously or frequently enough to form a background against which other sounds are perceived" (Truax 1999).

2 "A term derived from 'landmark' used in soundscape studies to refer to a community sound which is unique, or possesses qualities which make it specially regarded or noticed by the people in that community" (Truax 1999).

3 Pierre Schaeffer - *l'objet sonore* (Schaeffer 1966)

same small fragments of sound. The project was carried out further and completed by Keller in 1999. Keller's main tools for this project were Csound and Cmask (Keller 1999a). Keller produced the CD *touch'n'go / toco y me voy* as part of his research on ecologically-based granular synthesis (Keller 1999b).

The Attribute Study Framework

The methodology behind the attribute study is still under development, but the analysis tasks and some of the compositional tasks can be described in terms of the programs that have been written to perform the tasks. All of the programs share a number of commonalities. These will be discussed in this section.

All of the programs were written using the jMusic library, written by Andrew Brown and Andrew Sorensen (Brown & Sorensen 2000). jMusic is an open source Java based sound synthesis and compositional tool, and can run in most computer platforms.

The programs all use the SUN (.au) audio file format, and assume that the files are SUN files without verifying. Some programs however do have the additional option of saving a musical file in the MIDI format.

All the programs in this attribute study and those that follow will use the same data file format. The format is quite simple and allows transportability between the different attribute studies categories. This is a key notion as it will later allow the exploitation of the data. The simple nature of the format also allows editing. This means the composer can easily change data, delete sections, add or create new data, or even simulate an entire analysis sequence without performing any analysis.

The data file format itself is a text file. The first line is reserved for the name of the program that created it, so it can always be traced back to the original attribute. The next nine lines are reserved for parameter values that were set during the analysis process. These include the window size, sample size, frequency ranges, and other values that are yet to be determined. These are all included in every file, not only as a guide to the creation of the file, but more importantly as a standard control method for scaling and transforming the data. When the composer wishes to scale something, they need only set the scaling factor, and the program will work out how to scale it, based on the original values. The rest of the data file is reserved for the analysis data. Every value is on a separate line. This makes it much easier to process, as well as view in a text editor, or spreadsheet program.

The programs all use a similar interface. The programs are all executed in the console (shell) with the same syntax structure and similar command parameter options. The command parameters themselves are all identical across each program, but those that are irrelevant to a particular program are left out to reduce confusion.

To run any of the programs one would type in:

```
java program_name [command] \
   [parameter] [c] [p] [c] [p] ....
```

The command line options are as follows:

Command	: Parameter
`--help, -h`	: the help page (no parameter needed).
`--window, -w`	: sample size of the window in samples - default: int 128
`--datafile, -d`	: name of file for data input or output - default: String EnvInfo.dat
`--filename, -f`	: name of audio file to be analysed or processed - default: String input.au
`--length, -l`	: length of audio to analyse or process, in seconds -default: double 60.0
`--type –t`	: input source type for processing, either audio file or generated audio: type 0 = sine wave type 1 = white noise type 2 = audio file - default: int 0
`--scale –s`	: scale factor to use on data file - default: double 1.0
`--frequency –p`	: frequency (pitch) of base note in Hz - default: double 440.0
`--outputfile –o`	: name of audio output file - default: String output.au
`--samplerate –r`	: sample rate - default: int 44100
`--channels –c`	: number of channels -default: int 1
`--minfrequency –b`	: minimum (bottom) frequency for composition in Hz - default: double 50.0
`--maxfrequency –t`	: maximum (top) frequency for composition in Hz - default: double 2050.0
`--format –f`	:output format: audio(0) or MIDI(1) - default: int 0

Examples:

```
java ExtractEnvelope --help
```

```
java ProcessEnvelope -t 2 -f in-
put3.au
-o output5.au -s 1.4

java DataMelody --type 1 --scale 0.5
--length 20.0
```

The Data Extraction Process

ExtractAmplitude - The amplitude extraction process used in this study reduces the entire audio sample to a list of significant amplitude values. Firstly it windows the audio sample into segments of equal length, and records the maximum value of each segment in a data file. The data file containing all segment peak values is written out as a text file, and can be read with any text editor.
If the composer does not wish to analyse the entire audio file, they can set the length parameter, measured in seconds.

The window size default is 128 samples per window. When analysing a mono audio file at the sample rate 44,100 Hz, this equates to about 3ms per window. According to Dennis Gabor this value is well below the threshold of hearing. Gabor stated that a human ear requires 21ms to hear a single discrete sound event (Gabor 1946). At the default value, the data is reduced to 1/128th of the original size, without discernable information loss, and with head room for higher amounts of unnoticeable data loss.

The function of *ExtractAmplitude* can be viewed visually, as seen in figure 1. The figure shows the absolute values of the audio file, the windows, and the peak amplitude for each window.

The results of this program were simply verified by reading through the data file and checking that the flow of numbers bore resemblance to the audio file. The results would be much better verified with the second program.

Creating Envelopes

ProcessEnvelope - Once a data file has been created the audio sample file is no longer required. The data file contains all the amplitude information necessary. The most obvious function for the amplitude data is to use it to create envelopes.

ProcessEnvelope creates a breakpoint envelope with points designated evenly by the window size, and the peak amplitude for the current window. The envelope itself is a simple ramp based envelope applying a linear interpolation between values. Figure 2 displays an image of the peak amplitudes, with an envelope superimposed. The envelope is then applied to some sound source. This source could be an audio file, or a sine or white noise wave, generated from within the program.

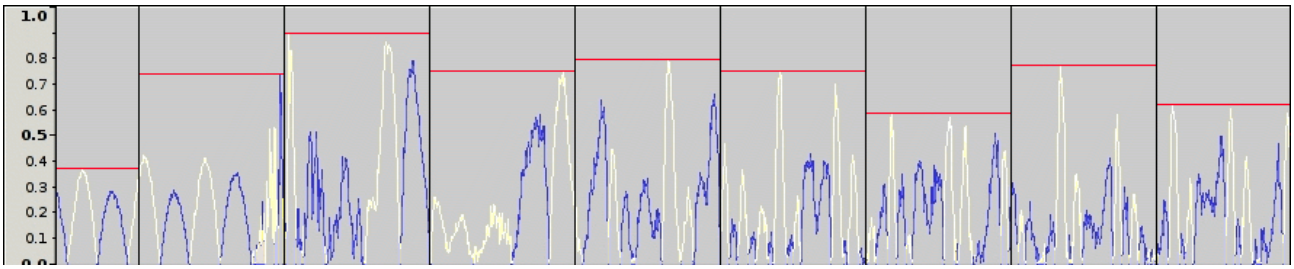

Figure 1) Audio wave with amplitude peaks – graphical representation of the ExtractAmplitude function

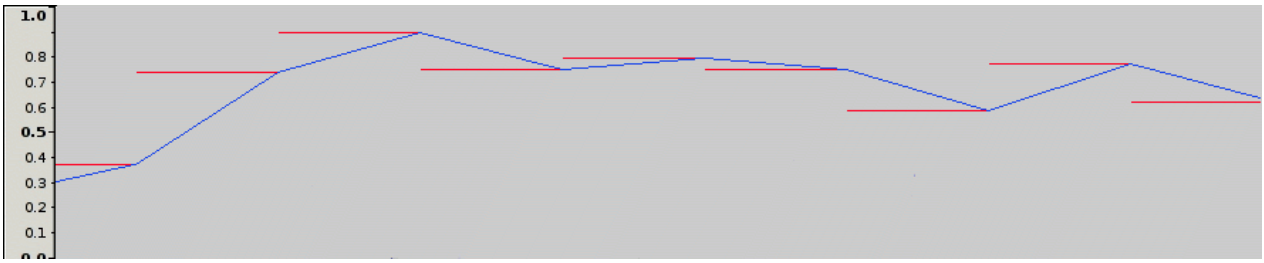

Figure 2) Envelope – graphical representation of the envelope created by ProcessEnvelope.

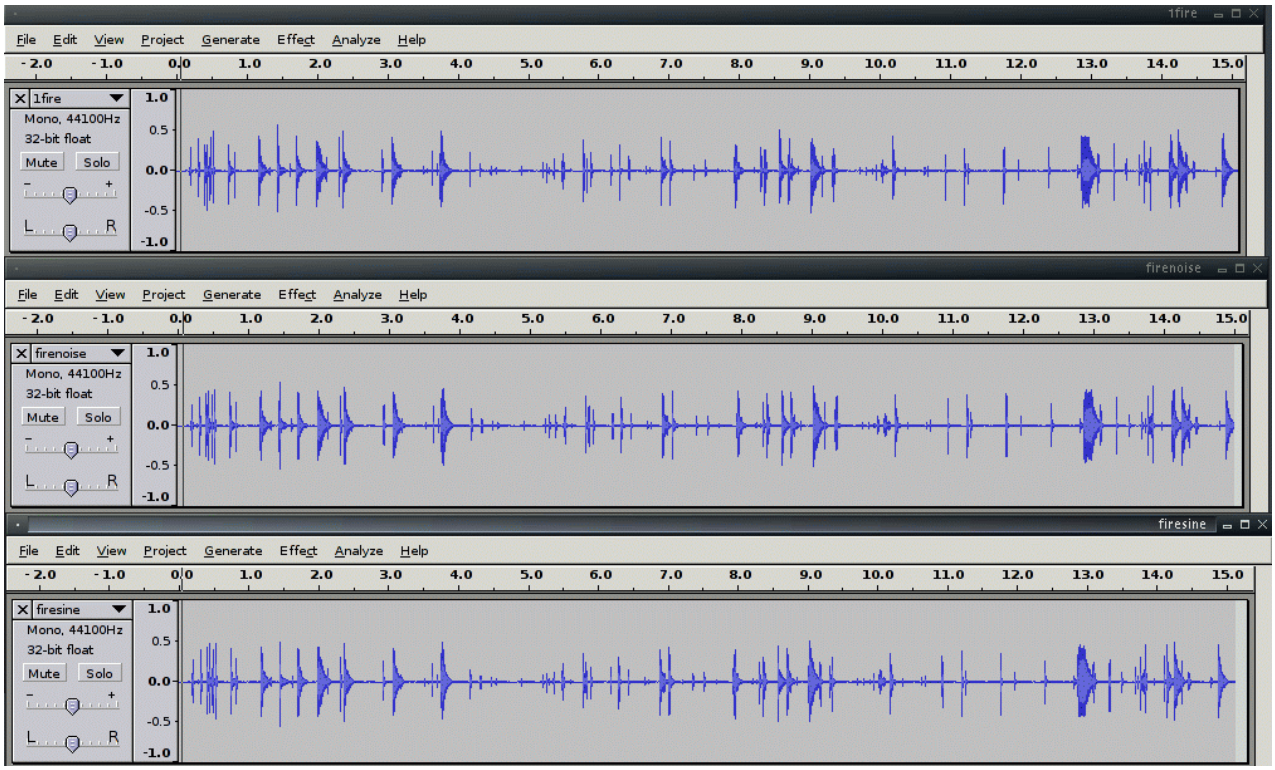

Figure 3) Graphical comparison of three audio files.

Probably the most important function is the scaling factor. An envelope can be created at any scale. If the composer wants an envelope that is twice the length of the audio file from which they originally created the data file, then the scaling factor is set at 2.0. The scaling factor works by multiplying the window size by the scale factor, and setting the breakpoints intervals using the result. So if you have a window of 128 samples, and you scale by a factor of 0.3, the break points will be set at 38 sample intervals.

After applying the envelope to a sound source, there were two things in particular that I noticed.
The first audition revealed what was expected. The amplitude characteristics were translated across the medium to the extent that the original data source could still be perceived. Figure 3 shows a graphical comparison of the original audio file, and two that have been created using the *ProcessEnvelope* program. The first graph is the original file; the second was generated with a white noise source; the third with a sine wave source at 440Hz. The comparison displays 15 seconds of audio, and reveals the shape of all three files is almost identical. The differences in the shape can be accounted to the timbral differences in the source file.

The second audition revealed something unaccounted for, an amplitude modulation (AM) effect. The small window size used especially if scaled smaller, would act as a modulator if the envelope fluctuated rapidly. An averaging filter was used to try and reduce the effect, but the AM sidebands were still noticeable. Instead, increasing the window size during the data collection process helped overcome this effect, although it did it at the cost of data acuracy. The default value of 128 samples per window is a compromise between data and effect. If the fluctuation rate is low there is no audible effect.

Amplitude Composition

DataMelody – The first level of abstraction from the data chosen for this project was to transform the amplitude to pitch. The reason for doing this was twofold. Firstly I wanted to begin experiments in interoperability, as it will become more prevalent and useful as I complete further attribute studies. Secondly I wanted to begin the compositional process, and create music and musical patterns with my first findings. The function of this program is to use the values of the data file to create a melody. This process is carried out systematically traversing the data file and converting the amplitude data to a pitch data within a set pitch range. The new composition can either be written out as a MIDI file, or an audio track.

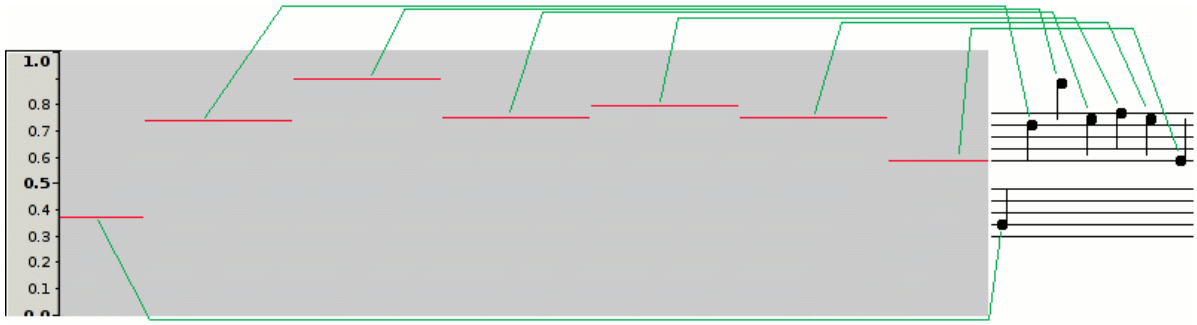

Figure 4) Melody – The data file has now been converted to form a melody.

The scale factor in *DataMelody* works by setting the length of the note. Once a note is completed, the next note in the sequence begins. The note duration is set using the real duration of the window size, and multiplying it by the scale factor. The tempo is set at 60BPS, so that the composer can work with a real-time scale.

If the default window size is 128 samples, and there is 1.0 (no) scaling, then the note value is 2.9ms at 60BPS. Composing a piece with such short notes actually puts it at a sub *granular synthesis* level of microsound control, yet it is being controlled with macrosound level terminology and tools. When using a granular synthesis approach the result is a mass of timbre. This is sometimes a little noisy, as the envelopes are the dominant feature of the synthesised sound rather than the contents of the grain. When stretching the time scale, the timbre stretches, emerging as a melody.

The minimum and maximum pitch can be set, and the amplitude data will be transformed and scaled to fit within the limits.

Figure 4 shows a graphical representation of the function of *DataMelody*.

Abstracting Fire

CracklingFire – Loading the data file into a spreadsheet program allowed for a second level of abstraction to be explored. The data file used was extracted from a crackling fire, the same data file as used in figure 3. Once the data was imported, it could be graphed and rearranged. Figure 5 is a simple graph of the data in the original order. Figure 6 is a close up graph of 2 of the peaks from the data. It displays a very sharp attack, with an exponentially shaped release. This information is very easy to recreate mathematically. Viewing the distribution of the amplitudes reveals extra useful information. Figure 7 is a graph of the distribution, and reveals a very simple exponential graph.

Using this information the program *CracklingFire* was created. It was used to model the crackling fire. The first process required modelling the amplitude shape. The distribution equation dictated when an amplitude spike would occur, and how high it would spike. The spike attack and release equation dictated the shape of the amplitude spike.

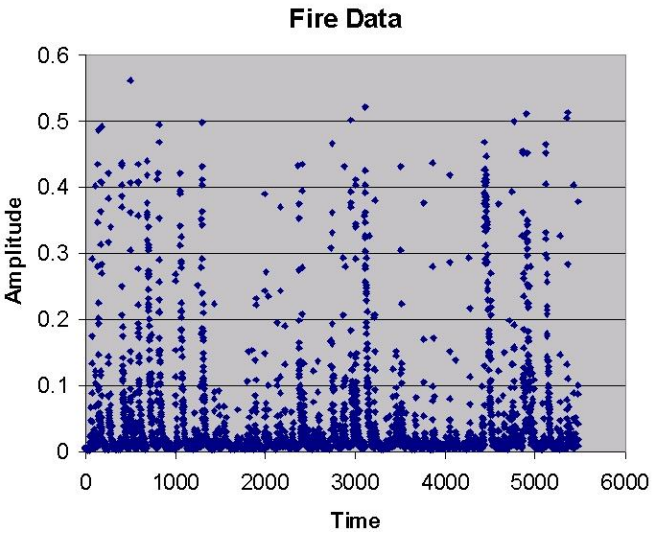

Figure 5) Graph of sequential data

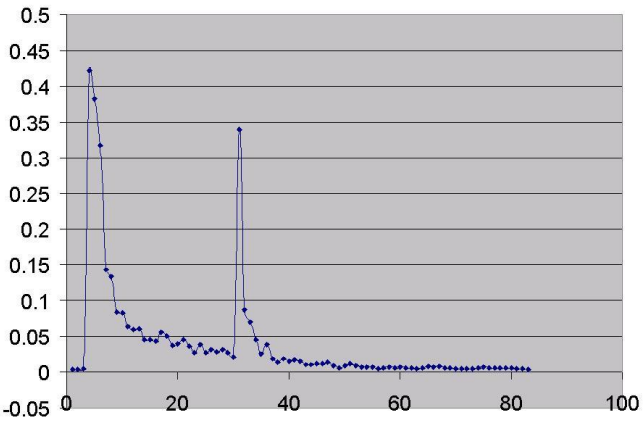

Figure 6) Graph of 2 peaks

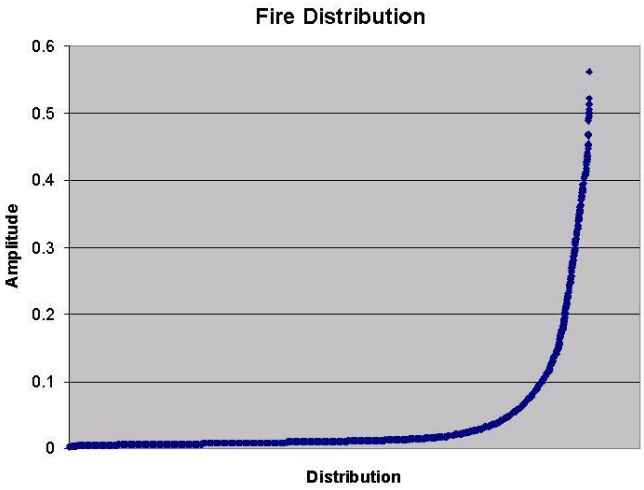

Figure 7) Graph of amplitude distribution

Once the amplitude of the piece was calculated, white noise was chosen as the content. This choice was a compositional decision, based on previous experiments which proved that white noise would work effectively.

This program allows the composer to create an ever changing crackling fire soundscape. Initial listening tests have proven it can pass for an imitation crackling fire, although changing the timbral content would really enhance the effect. Using a recording of a burning fire may also improve the quality of the soundscape.

Future Vision

This first attribute study in amplitude is the initial foundation for a much larger project involving other characteristics of audio such as frequency, duration, timbre, and space. The main goal of the project is that all the aspects from a naturally occurring sound can be analysed and used to create one full composition, rather than just a melody, or amplitude model. In order to make this more meaningful and interesting, each of the analyses results needs to be interchangeable with any other, from any audio aspect. All the results combined can be used either to recreate a sound entirely based only on analysis of it, or more interestingly they can be shuffled around to create hybrids of sounds, new sounds, and new compositional ideas.

Another tool that will be useful in this analysis that is yet to be pursued is a data mining process in order to find patterns within the data file that can be used as compositional substructures, and musical motives. Finding smaller groupings within a larger data file may prove much more beneficial from a compositional prospect than using the entire data file. A data mining procedure is being investigated, and the results will be tested at a future date.

Aside from the technicalities of the project, a major aim is to produce interesting musical compositions. The melodies created so far have, in my opinion, been quite interesting. It is a good sign that when the later scheduled audio aspects have been analysed, they will provide rich and very interesting musical compositions.

Conclusion

An ecologically based perception of soundcapes and composition offers new means and categories with which to analyse, resynthesise and create sounds and compositions based on naturally occurring counterparts. In order to begin the creation process a project was laid out that set up a number of attribute studies to analyse and work with different attributes of sound. The first of these attribute studies is well under way, and involves the analysis of amplitude.

The initial process of the attribute studies was to set up the framework whilst beginning on amplitude analysis. Four different tools have been created that analyse and use amplitude data in various ways. The framework so far is proving effective, and is set up in such a way that future attribute studies can use the same program structures, reducing production time, and maintaining uniformity.

The programs themselves have been found useful, and promise affluent and abundant compositional ideas, especially when combined with future attribute studies.

References

Brown, A. and Sorensen, A. 2000. "Introducing jMusic", *Proceedings of the Australasian Computer Music Conference 2000*, pp. 68-76, ACMA.

Clynes, M. 1986. "Generative Principles of Musical Thought Integration of Microstructure with Structure", *Journal for the Integrated Study of Artificial Intelligence, Cognitive Science and Applied Epistemology*, Vol. 3, No. 3 pp. 185-223.

Gabor, D. 1946. "Theory of Communication", *The Journal of the Institution Of Electrical Engineers*, Volume 93(3), pp 429-457.

Gibson, J. 1979. *The Ecological Approach to Visual Perception*, New Jersey, Lawrence Earlbaum.

Keller, D. and Truax, B. 1998. "Ecologically-Based Granular Synthesis", *ICMC Proceedings 1998*, Ann Arbor, MI: ICMA.

Keller, D. 1999a. *touch'n'go: Ecological Models in Composition*, master of fine arts thesis Burnaby, BC: Simon Fraser University.

Keller, D. 1999b. *touch'n'go / toco y me voy*. Enhanced Compact Disc. Burnaby, BC: Earsay Productions.

Keller, D. 2000. "Compositional Processes from an Ecological Prespective". *Leonardo Music Journal*, Vol 10, pp 55-60

Schaeffer, P. 1996 *Traité des Objets Musicaux*, Paris, in Truax 1999.

Truax, B. 1991. *Pacific Rim*, CSR-CD 9101, Cambridge Street Records.

Truax, B. 1992. "Musical creativity and complexity at the threshold of the 21st century", *Interface,* 21(1), 29-42

Truax, B. 1999. Handbook for Acoustic Ecology, Second Edition. Cambridge Street Publishing.

Windsor, L. 2000. "Through and Around the Acousmatic: The Interpretation of Electroacoustic Sounds", *Music, Electronic Media, and Culture*, pp 7-35, Aldershot, Ashgate Publishing Ltd.

The audio files were viewed and compared using Audacity. Screenshots were captured using XView. The Gimp was used to create the function overviews.