Converting Electronic Music to Sheet Music
TJHSST Senior Research Project Paper
Computer Systems Lab 2009-2010

Hugh Smith

April 7, 2010

Abstract

Electronic music has been steadily expanding over the past years. Many file formats have come into use, including WAVE, MP3, Ogg Vorbix, and many others. This project hopes to take any one of these file formats, and, based on the pure audio wavelength data (what the computer must see to play the song), convert it to a sheet music version. **Keywords:** music, analysis, mp3, wav, sheet, music

1 Introduction

Problem Statement and Purpose This project will involve reading the audio data from an electronic audio file and converting it into a file format called ABC. The ABC format will be discussed in more detail later in the paper. This process will be difficult because converting music files to basic sound data can be very complicated, especially in the case of MP3 and other "compressed" formats.

2 Background

I need to have a good understanding of how C++ works. Also, I need to know musical composition, and how virtual music files are put together. The reason for knowing these things is so I can perform the operations stated above in the fastest time. With bigger music files, the analysis portion of this project could take a long time, so I need to be able to optimize the process. I know some previous research has been done in this area, by some TJ students and other researchers.

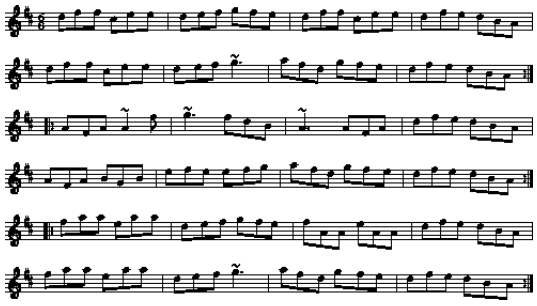
3 ABC Format

ABC notation is a way to represent sheet music in a text file. The format is simple, with the letters of the keyboard serving as the notes of the piano. You can add information such as the title, the arranger, the performer, and other things to the music. In addition, there are lots of programs available for converting this notation to a

PDF document, so it can be actual sheet music. ABC notation is best for rendering single-melody songs, so it will be very useful for my project. The general format of an ABC file is as the following example:

```
T:Paddy O'Rafferty
C:Trad.
M:6/8
K:D
dff cee|def gfe|dff cee|dfe dBA|\
dff cee|def gfe|faf gfe|1 dfe dBA:|2 dfe dcB|]
~A3 B3|gfe fdB|AFA B2c|dfe dcB|\
~A3 ~B3|efe efg|faf gfe|1 dfe dcB:|2 dfe dBA|]
fAA eAA|def gfe|fAA eAA|dfe dBA|\
fAA eAA|def gfe|faf gfe|dfe dBA:|
```

This creates a sheet music file like so:



ABC format is a well-used format, with many websites offering versions of their MIDI or MP3 files in ABC format as well. An example is The Session (http://thesession.org/). This website is a collection of Irish traditional folk music. This is useful, because, most folk music has one melody, and a single line at a time. For my project, these types of songs will be best to test my program on, as there is a low probability of getting the song wrong if the program is actually working.

4 Description

So far, I have a working program that reads in a .WAV file and prints out information gleaned from the information chunks of the file. It also saves the file, in chunk form, to a buffer, and then writes it out to a copy of the file, just to show that it can, and the structure of a .WAV file can be easily imitated. The structure of a .WAV file is pretty simple; it consists of three main sections, and each has a number of chunks contained within it. The chunks contain information such as the type of file it is, the bitrate, the sample size, the number of channels (mono, stereo, etc.). These are used by music players, mostly. It also contains information about the type of compression the file uses; if there is one, some extra chunks are added to tell more about it. This will probably be one of the hardest parts to complete, as I need to be able to convert from the compresion into actual data. This program is still useful in the second quarter of my project, because I will need to convert the files to audio data. I have done some more research on this, from my main research book "Elements of Computer Music," by F. Richard Moore.

5 Elements of Computer Music

All waves are defined by two things: frequency and amplitude. Basic physics tells us the dynamics of these waves. Sound waves are exactly the same. They travel fast, albeit slower than light, and through most

non-vacuum mediums. Computer sound files work the same way, in fact. Energy is converted into data by the use of a transducer. An analog signal is then sent out, which the computer can interpret into sound.

6 Sound Waves

All sound waves can be expressed as the sum of many waves. Many, in this case, really does mean a lot of waves - think of how many different waves it takes to express a short sample of a human talking. Music expressed in sound waves can be even more complex. Electronically generated music is easier to deal with in this regard, because you do not have to deal with random fluctuations in tone or volume based on instrument or player error. Now, dealing with an uncompressed .WAV file, the file has to store the information about the sound wave in numerical form. Deciphering this takes some work. However, it is possible with something called the Fourier Transform.

7 Fourier Transform

The Fourier Transform is a way to separate a complex waveform into many different sine or cosine waves. It is a complex algorithm, and as I had not learned it in any of my math classes so far, I had to learn it on my own. The formal formula is as follows:

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)\, e^{-2\pi i x \xi}\,dx,$$

As you can see, it is rather complex. I have tried to convert this into code, and with a little help with a website, I believe it works. However, there is one major problem with my project.

8 The Problem

And that is that it is extremely, extremely slow-running. The problem is in the sample size in the .WAV file. For the Fourier Transform to work accurately, it has to analyze each of the "sample points" in the file. For the five-second .WAV clip that I am using to test the program, the program has to do approximately 10+ billion calculations. Obviously, this does not work out very well. On my jacked-up computer at home, it takes at least forty minutes to run. I don't even want to think about how long it would take on a single syslab computer at school. This is the major problem with my project - that it simply does not finish running in time.

9 The Solution

I have come up with a solution, though, and that is to use MPI to make the process much faster. Since I am taking parallel Computing in parallel with the research lab, it will be relatively easy to incorporate the theories I have learned in that class into this one. Although it is not working yet (mostly because of the differences between C and C++ implementation of MPI), once I get it working, the program should be running at normal-ish

speeds.

10 Conclusion

I have progressed much on my project since last quarter. I have a clear idea for my project, and I have almost implemented it all. I have a working Fourier Transform program working, that analyzes the raw data in a .WAV file and eventually produces an array of numbers that defines the amplitudes and frequencies of the many waves evident in the file. The two things I have left to do is add code to convert the pitch values into notes, and implement MPI to make the program faster.

11 Bibliography

http://www.sonicspot.com/guide/wavefiles.html
https://ccrma.stanford.edu/courses/422/projects/WaveFormat/
http://thesession.org/ Elements of Computer Music (F. Richard Moore)
http://www.dspdimension.com/admin/dft-a-pied/