Computer Science Specialist Project - Interim Progress Report

David Halliday

December 9, 2004

Contents

1 Introduction

1.1 Problem Details

More people around the world are using electronically stored music for a number of reasons including ease of use, portability and ease of sharing/distributing.

The most popular format of digital music people are storing on their computers is MP3. These files can store additional information about the track including artist and track names. However these aren't always entered properly and if someone has a large number of mp3 files in one directory it is hard to locate the desired file.

My aim is to make an application to edit and search through music files and organise them into folders by a user defined system.

1.2 Precise Objectives of Project

The Project is to construct a GUI tool to read and edit extended information on audio files and then sort the files by these details into sub directories.

1.2.1 Make Platform Independent

I want my programme to be able to run on all the common operating systems (windows, Unix, Linux, Mac). This means I have to find a language that will allow me to develop it in such a manner.

1.2.2 Learn Chosen Programming Language & Libraries

Whatever language I choose I will have to learn how to use the language and its libraries in many areas especially as I have no GUI programming skills outside of using RAD (Rapid Application Developmeant) tools such as Borland Delphi or Microsoft Visual Basic.

1.2.3 Support Multiple file formats

I will primarily base my project on working with MP3 files as they are the most common, however there are a number of other file formats that store music including wma (Windows Media Audio) and ogg vorbis.

1.2.4 Provide functionality to organise music files

This is to be done by extracting information from text held in the file containing things like the track and artist names. The idea is to sort them according to the

users specifications such as:
\Band Name\Album Name\

1.2.5 easy & efficient GUI

The GUI has to be easy and efficient with the important and commonly used functions easily accessible.

2 Research & Progress

2.1 Academic Background & Research

2.1.1 MP3 Files

Information found from [Nilsson, 2004] MP3 files have two methods of storing the track information. both known as “ID Tags” which are:

ID3v1 This is the oldest and most simple. It works by adding 128 bytes to the end of the file that contains ASCII text for the track information it works on the basis:

Song Title	30 characters
Artist	30 characters
Album Title	30 characters
Year	4 characters
Comment	30 characters
Genre	1 character

The first 3 bytes contain the letters “TAG” so that it can be identified as being there or not.

ID3v1.1 The ID3v1 format managed to not include information on the track number. Because of this it had to be adapted to hold the track number. This was done by removing the end of the comment and adding an extra field giving the following layout:

Song Title	30 characters
Artist	30 characters
Album Title	30 characters
Year	4 characters
Comment	28 characters
Blank Space	1 character (mandatory space)
Track No.	1 character
Genre	1 character

ID3v2 This is more complex but circum-navigates all the shortfalls of the ID3v1 format. It works by attaching information to the start of the file and allows for variable sized fields getting past the 30 character limit. Useful when you're working with "It's The End Of The World As We Know It (And I Feel Fine)" by REM or ("It's The End Of The World As W" on ID3v1/ID3v1.1)

After this research I decided I would build my system to support only ID3v1 to start off with as it is the simplest and I only need one format for my other functionality to work then I can easily add further tag support at the end.

2.1.2 Programming Languages

Java is the first language that comes to mind when developing for platform independent applications. However due to its nature a Java programme is generally slower than a fully compiled programme. Java is also good for GUI tools as there are a whole suite of built in libraries and facilities for GUI development.

C and C++ are another possible choice in that they have better performance than Java as fully compiled binary. Both languages in their basic form are platform independent however there are no graphics libraries built into the standard C and C++. Only third party libraries that can be installed which have functionality for all sorts but not all are available on all platforms. Some libraries may be similar or their functions may double up so to get round this separate versions of classes could be used (in different files) with the same public variables and functions for different operating systems.

2.1.3 WxWidgets

Fortunately after researching the different possibilities I found a library called wxWidgets (formally called wxWindows but had to change its name for legal reasons where another company had rights over the name, (possibly a double glazing firm) which is good for GUI programming with a number of common functions and dialogues.

The library works with C, C++ and python however can be expanded to other programming languages. It has been under development for a number of years with a number or GNU and commercial programmes being developed under it. One programme that I looked at. , that was developed under it, is "audacity" which is a platform independent audio editing application.

This library had a lot of features and resources within it for controlling the GUI and for other areas that are not normally platform independent such as file reading and writing functions.

The functions from this library that are most important to me at this stage are:

basic GUI The library allows you to quickly construct simple GUI applications starting with a basic window which can then have things added to it such as:

- **Text Boxes** these are powerful enough to become a basic editor for text or RTF files
- **List Boxes**
- **Combo Boxes**
- **Drop Down Boxes**
- **Image Boxes**
- **tick Boxes**
- **Menu Bars**

File procedures Functions to create, copy, rename, delete, read and write to files and directories.

2.2 Progress

The Project up until a week ago consisted of an application made for testing and learning purposes for C++ and the wxWidgets library. It was coded in a single text file with lots of repeated code and excessive comments. The application has more recently started being split up to use more object oriented programming techniques for speed and ease of programming. As I continue to better structure the programme the level of functionality increases as classes and functions are able to interact together better. For ease of programming I have separated some of the functions and classes into separate files:

mp3org.cpp contains the headers and wxWidgets equivalent of main.

IDlinks.cpp contains ID numbers for different event driven functions.

mp3frame contains the class and function information for the frame object which controls the GUI.

mp3file contains the class and function information for working with the physically stored files.

2.2.1 C++

After choosing to do the project in C++ I found very quickly that my basic knowledge of C++ was going to need a little revision and expanding. I was able to get more information from [Liberty, 2002] which gave me all the C++ i needed to work with the library.

2.2.2 Use of wxWidgets Library

I had difficulty at first with the installation of the library as I am primarily developing the software on a computer running Debian Linux and when I used apt-get (the Debian package installation programme) it downloaded and installed most of the programme but left parts out as the dependencies were not correctly set in the archive. Once this problem was solved there was no difficulty getting it to work with my chosen compiler (gcc).

I have had to learn the use of this library from scratch, fortunately I have been able to find tutorials and sample code to help me get started. The tutorials I have used are a basic hello world programme [Roebling, 2004] and then a more comprehensive tutorial [Beech, 1999]. Once I had worked my way through these I was more confident with looking at the full WxWidgets manual (with less in the way of instructions) [Smart, 2003]. This has a full list of classes functions and descriptions of their uses.

The areas that I have had to spend a lot of time looking at are the file manipulation functions and the listing functions.

The file manipulation took me quite a long time as I made the mistake of telling the programme to read 128 bytes after the end of the file rather than 128 bytes before the end of the file which returned a number of seemingly random and confusing characters.

The listing took some time to do as there were two different list functions in the library. The first is simple to use but with less functionality. I In many ways it would be sufficient except that it can only hold up to 3000 items, which in a collection of MP3s that numbers over 3000, (as mine does) it would be insufficient. So I had to use the more complex list function that has lots of extra features. However once it is fully implemented it should allow me to easily incorporate more functionality into the list.

2.2.3 Makefile

To aid me in quickly developing my project I modified a makefile from the wxWidgets tutorials that would pass all the required switches to the compiler so all I had to do was type “make” at the command line to compile my project. I got further information from [Sobell, 1997]

2.2.4 achieved Functionality

This is a list of the functions that have been created in the "testing" and "actual" versions of the programme:

Check For tag Check to see if an MP3 contains an ID3v1 tag.

Read & separate tag extracts and separates the separate parts of the ID3v1 tag.

Display an "open file dialogue" using a library call to create an "open file dialogue" and extracts the filename from it to be used by the application.

Directories work copy a file into a subdirectory with a specified name (and create the directories if it does not already exist).

Draw a list Draw a list in the GUI and list all the files in a given directories.

3 To Be Done

3.1 Restructure Programme

I have begun to restructure the programme and separate some of the functionality into its own classes such as all the file manipulation and the functions for drawing the window are now in their own classes. This makes the code more efficient, easier to read and update.

3.2 Add more file format support

I aim to support a number of files I am currently supporting MP3 files with ID3v1 tags, I aim to support MP3 files with all three tag styles (ID3v1, ID3v1.1, ID3v2) as well as other file formats including ogg vorbis and windows media audio.

3.3 Multiple & Batch editing

Currently all the functions in the application can only be done one at a time, however through the use of object oriented programming and manipulation of the list control, I aim to be able to make most if not all functionality in the application to be performed in batch as well as one off. This would allow the user to change a section of information (such as band name) on multiple files simultaneously the same way as they would with just one.

3.4 Improve & extend GUI

The current GUI is still very much based on the experimental/testing status of the programme. I need to reconsider and redesign the GUI for ease of use and accessibility. I can get further improvements through feedback from friends. One way I aim to add more features to the GUI is to add a number of lists to separate and organise different aspects of the audio description tags.

4 appendices

4.1 bibliography

References

[Beech, 1999] Beech, D. (1999). *WxWidgets tutorial*. `<http://www.bzzt.net/~wxwindows/icpp_wx1.html>`. An intuative tutorial on using many features of WxWidgets.

[Cadenhead, 2001] Cadenhead, R. (2001). *Sams Teach Yourself Java in 24 Hours*. Sams, 2nd edition. only used for java code on how to read from ID3v1 tags.

[Liberty, 2002] Liberty, J. (2002). *Sams Teach Yourself C++ in 24 Hours*. Sams, 3rd edition.

[Nilsson, 2004] Nilsson, M. (2004). *ID3v1*. `<http://www.id3.org/id3v1.html>`. Website documenting how the ID3 Tags work on music files (ID3v1, ID3v1.1, ID3v2).

[Roebling, 2004] Roebling, R. (2004). *hello world tutorial*. `<http://wxwindows.org/hello.htm>`. hello world tutorial for beggining using WxWidgets.

[Smart, 2003] Smart, J. (2003). *WxWidgets Manuel*. WxWidgets, `<http://wxwindows.org/manuals/2.4.2/wx.htm>`. A full guide written in HTML containing details of all the classes and functions and their uses.

[Sobell, 1997] Sobell, M. G. (1997). *A Practical Guide to Linux*. Addison Wesley, 1st edition. A little out of date but usefull for information on linux and Makefiles.

4.2 code

4.2.1 ID3v1 reading code

i was able to get a good headstart by getting the following Java code from a book [Cadenhead, 2001] where there was an example of reading strings from files that used MP3 files:

```
import java.io.*;
```

```
public class ReadID3
{
 public static void main(String[] arguments)
 {
  try
  {
   File song = new File(arguments[0]);
   FileInputStream file = new FileInputStream(song);
   int size = (int)song.length();
   file.skip(size - 128);
   byte[] last128 = new byte[128];
   file.read(last128);
   String id3 = new String(last128);
   String tag = id3.substring(0, 3);
   if (tag.equals("TAG"))
   {
    System.out.println("Title: " + id3.substring(3, 32));
    System.out.println("Artist: " + id3.substring(33, 62));
    System.out.println("Album: " + id3.substring(63, 91));
    System.out.println("Year: " + id3.substring(93, 97));
    System.out.println("Comment: " + id3.substring(98, 127));
    System.out.println("genre: " + id3.substring(128));
   }
   else
   System.out.println(arguments[0] + " does not contain ID3 info.");
   file.close();
  }
   catch (Exception e)
  {
   System.out.println("Error -- " + e.toString());
  }
 }
}
```

4.2.2 My Full Code

From here Follows my entire code listing.