intuiTunes

Enhancing The Portable Digital Music Player Experience

Stephen Pike
293828

May 2006

Abstract

The world of mobile music players is ever-growing in both popularity and storage size, allowing users to access more and more of their collection. With this access comes a problem – how do we select the music we want to hear from such a large collection? This project employs extended meta-data in order to construct musical contexts, enabling the user to quickly locate tracks which are appropriate to their current personal world. Introducing...

... intuiTunes

Project Dissertation submitted to the University of Wales, Swansea
in Partial Fulfilment for the Degree of Bachelor of Science

Department of Computer Science
University of Wales Swansea

Declaration

This work has not previously been accepted in substance for any degree and is not being concurrently submitted for any degree.

Name: Stephen Pike

Signed:

Date: 05/05/2006

Statement 1

This dissertation is being submitted in partial fulfillment of the requirements for the degree of BSc Computer Science

Signed:

Date: 05/05/2006

Statement 2

This dissertation is the result of my own independent work / investigation, except where otherwise stated. Other sources are specifically acknowledged by clear cross referencing to author, work and page(s) using the bibliography / references section. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this dissertation and the degree examination as a whole.

Signed:

Date: 05/05/2006

Statement 3

I hereby give consent for my dissertation to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed:

Date: 05/05/2006

Acknowledgments

I would first of all like to thank a few people for their time and support...

Thank you firstly to my peers in the lab, especially Dave and Stu for putting up with my coding rants for which I apologise. Thank you guys for chilling me out and helping me rationalise things. Thanks to Stu and Rose for proof reading this document.

A big thank you to my tutor and supervisor, Dr. Matt Jones, whose impeccable guidance, enthusiasm, humour and motivation have encouraged me and kept me going especially during the lows.

I have taken a lot away from this project and I hope that what I have achieved is useful to someone other than myself. The world of Human Computer Interaction is truly an exciting one and I am glad to have been a part of it.

Contents

Chapter 1

Introduction

Music players have been a part of our desktop computer systems for as long as we have been able to store music digitally. Portable music players have been around for a lot longer, dating back to the earliest battery powered Hi-Fi systems (or 'Boomboxes') of the mid 1970s, and later on the slightly more portable Sony Walkman of 1979[1]. As music moved from being stored in an analog format, to that of a digital one with the advent of Compact Discs (CD) and market impact in the early 1980s[2], portable music players changed too. The introduction of the Sony Discman in 1986 followed the move to CDs, and more and more music 'users' started recording their CDs onto computers in MPEG Layer 3 (MP3) format. Audio companies started to produce portable players capable of playing this new format (MP3) in 1999. The latest innovations from Apple, of course, are the highly successful iPod family.

Having all this portable music technology allows us to carry our music collections (or subsets thereof) with us wherever we choose. Pre-digital players such as the Walkman and Discman meant that users had to carefully select which collections they wanted to take with them, often resulting in the production of 'Mixtapes' – home-produced compilations of different tracks which the user wanted to have with them. In more recent years, as digital music has become more and more popular, many large corporations and other groups of programmers have been producing Portable Digital Music Players (PDMPs) such as Apple with the iPod, to allow the masses to listen to *all* of their music. However, as storage becomes cheaper, and therefore more available, and with broadband Internet access providing a means with which to acquire music (be it legally or otherwise), users' collections have been becoming larger and more eclectic.

With growing collections comes the problem of organising the content and finding effective ways of browsing and searching it. Current desktop music players have attempted to solve

[1] `http://pocketcalculatorshow.com/walkman/sony/`

[2] `http://en.wikipedia.org/wiki/Compact_Disc#History`

Figure 1.1: *An early Boombox of the 1970s[28]*

these problems, with applications like GNOME's Rhythmbox[3] and Apple's iTunes allowing 'smart' playlist creation[4]. However, these methods still require a fair amount of input from the user. What if we could automate and extend this, and provide it in a mobile environment?

1.1 Aims

This project aims to tackle the latter problem; how can we use a PDMP to help the user to retrieve content based on what they want to hear and also, how can we use the PDMP to gather extended data, without too much input from the user. By achieving this we also tackle the former problem by giving emergent organisation based on 'Idiosyncratic Genres' [9].

Specifically then, we intend to enhance the experience of the user by using extended data to compartmentalise their listening habits. This data is to be gathered as silently as possible, and used to maximum effect in creating idiosyncratic playlists. The user will have limited input in the gathering phase, but will have control over how it is used in the manipulation phase. This project will set itself in the context of a user who has access to the modern convenience of broadband Internet and has a large collection of digital music stored on a portable storage device. For simplification and ease of implementation this project will use a Personal Digital Assistant (PDA) as the basis for our Portable Device. In this way, the PDA becomes a PDMP, but with the added advantage that it is extendable to allow other data input methods via Bluetooth, Infrared, WiFi and through the serial and Compact Flash sockets. From a commercial point of view, only those data input methods of use would be

[3] `http://www.gnome.org/projects/rhythmbox/`
[4] `http://www.apple.com/itunes/playlists/`

included in the final PDMP design, but for this project as a proof of concept, it is enough to show that these methods are feasible.

The intended environment is one in which the user is mobile, and as such will be using their PDMP for different purposes, at different times of the day and in different locations. On top of that, if the user is a heavy music user (i.e. they listen to music for all occasions, no matter what), then they will also be listening in different emotional moods, personal situations and other non-physical/controllable factors (perhaps even as far as changing weather!). Taking all these factors into account will give us a rich set of meta-data to work with to decide what might be the best 'type' of music to deliver. We can for example, extrapolate from location given by Global Positioning Systems (GPS), (which is now commercially available at relatively cheap prices), that our user was at the gym or training, and as such we can produce a playlist for that context. Using the time of day and location we could adjust volume and produce a playlist of (say) 'chill out' music at home in the evening, or a playlist to keep you awake whilst driving at night. The actual selection of music and creation of playlists, will be done on the desktop environment, which could later then be moved back to the PDMP.

Given all these different factors, the main aim is to see what patterns appear in the data acquired, and use these patterns to work out useful/meaningful trends in music 'habits'. If this is achieved then we can enhance the user's music-listening experience.

1.2 Project Overview

To achieve our aims, we produced a prototype system. This system is split into two main phases: one based on the mobile device, and the other on the desktop. It works as follows:

Phase 1

- Given all the software and hardware necessary, the user deploys the desktop software onto their desktop machine.
- The user uploads their choice of music onto the PDA via the serial uplink.
- The user creates their profile on the desktop software, setting their moods and related colours.
- The settings are uploaded to the PDA
- The user then uses the PDA as their replacement music player, wherever they go.
- The user sets their mood as it changes, within the music player.
- The PDA logs location via a Bluetooth connection to a GPS (Global Positioning System) receiver, time, mood and track data to file.

As we can see, phase 1 mostly takes place on the PDA. The focus of this phase being on making the interface as easy to use as possible, and also in retrieving and logging the data in the most unobtrusive way to the user possible. The user will have their own selection of music for evaluation, as it is necessary for the user to know which tracks relate to their moods in order for the logged data to be accurate.

Phase 2

- The log files are downloaded from the PDA onto the desktop.
- The desktop software imports the log files.
- The user can now observe their habits.
- Using the software, the user can form 'contexts' for which they have certain moods, times and locations
- These contexts form playlists and can be saved, reloaded and played back
- Web-services (using Last.fm[5]) are employed to broaden the selection of music in the context, based on similar artists, and what the user has in their library.

Phase 2 is where we discover how useful our software is. If the user finds the extra meta-data a useful, and convenient (and not confusing) in creating these contextual playlists, then we have achieved our goal of enhancing the music player. If however, they do not see the point or understand how it could be used, then we should re-evaluate the implementation. Figure 1.2 shows an overview of the system.

[5] `http://last.fm`

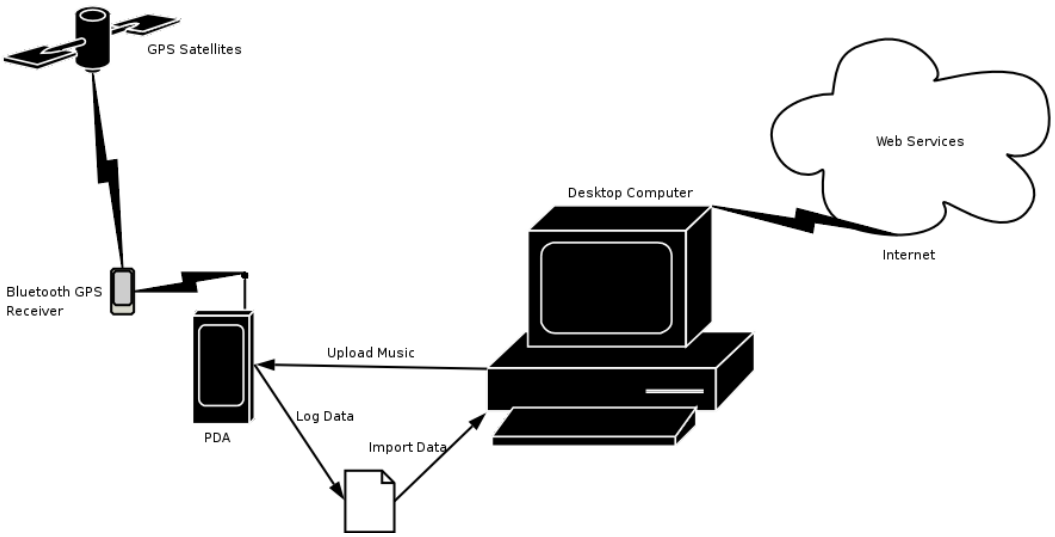


Figure 1.2: *An overview of the system. Data is logged on the PDA and uploaded to the desktop, where it is used to create contextual playlists, with playlists 'filled out' by recommendations from web-services*

Chapter 2

Background Information and Related Work

Following is an in depth look at background to the project, with a discussion of previous investigations into, and uses of the key concept areas.

2.1 Metadata

Meta data has long been used in a Computer Science environment to describe information 'objects'.

> **Metadata** (Greek: meta-+ Latin: data "information"), literally "data about data", is information that describes another set of data.[37]

Meta-data has been used in all types of media, a simple example- meta-data is used in the <head>tags of HTML documents to help search engines index pages by content (figure 2.1). Even in an office environment we can see the use of meta-data; simply by naming a

```
<HEAD profile="http://www.acme.com/profiles/core">
  <TITLE>How to complete Memorandum cover sheets</TITLE>
  <META name="author" content="John Doe">
  <META name="copyright" content="&copy; 1997 Acme Corp.">
  <META name="keywords" content="corporate,guidelines,cataloging">
  <META name="date" content="1994-11-06T08:49:37+00:00">
 </HEAD>
```

Figure 2.1: *An example of some meta-data added to an HTML page[36]. probably one of the simplest examples of the use of meta data, it adds information about the author of the page and essential key words that search engines may use to categorise this page, data that may not be apparent from the immediate content of the page.*

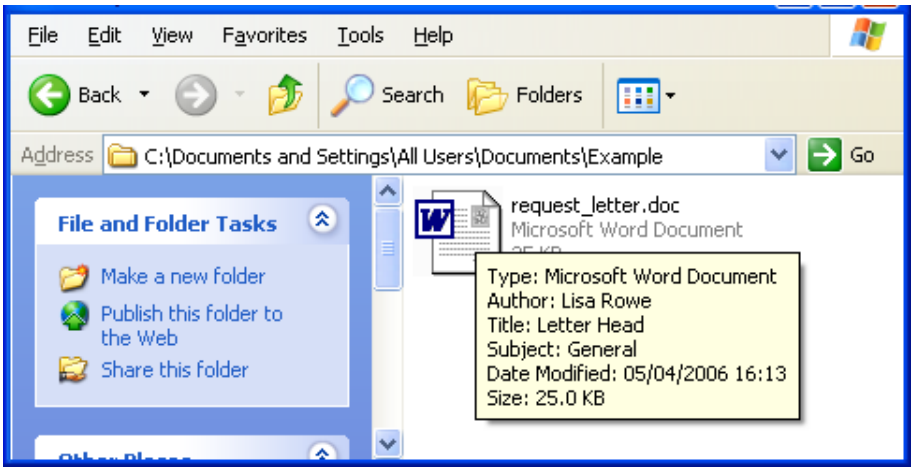


Figure 2.2: *Meta-data attached to a Microsoft Word Document, to quickly overview information about the document that otherwise would only be apparent by opening it.*

file we are adding meta-data which describes the file to other users, and also data is often attached to documents containing information about the Author, Time and Date of creation and sometimes even a short description of the contents (figure 2.2). However, meta-data is most useful for objects that have no 'natural' way to communicate what they contain to computers. For example, if you have a text document, it is plausible that a computer program could search the text and derive some meaning from it, and some context. Take a digital photograph, on the other hand, and it would be very difficult for a computer program to retrieve information about the photo. Current digital cameras will now automatically attach data to any photographs it takes, stating the data and time, the model of the camera used and other details about exposure settings and so on (as seen in figure 2.3). A caption can also be added by when the image is transferred to a computer.

2.2 ID3 tagging

This use of meta-data in digital media leads us onto digital music. Digital music has had meta-data attached to it in a similar way to that of digital photographs. In 1996, Eric Kemp [20] invented ID3 tags for Digital Music, namely the MP3 format. This allowed users to add meta-data concerning the Artist, Title, Album and Genre, amongst others, of a track to its MP3 file. Since then, ID3 has been updated to ID3v2, providing a richer set of data fields to store information in.

Having come into wide use, music meta-data was used many applications (mainly music players) to organise the music and inform the user about their music library. Here is an overview of some widely used applications that make use of meta-data in music:

- **iTunes** [7], created by Apple Computers makes considerable use of meta-data to or-

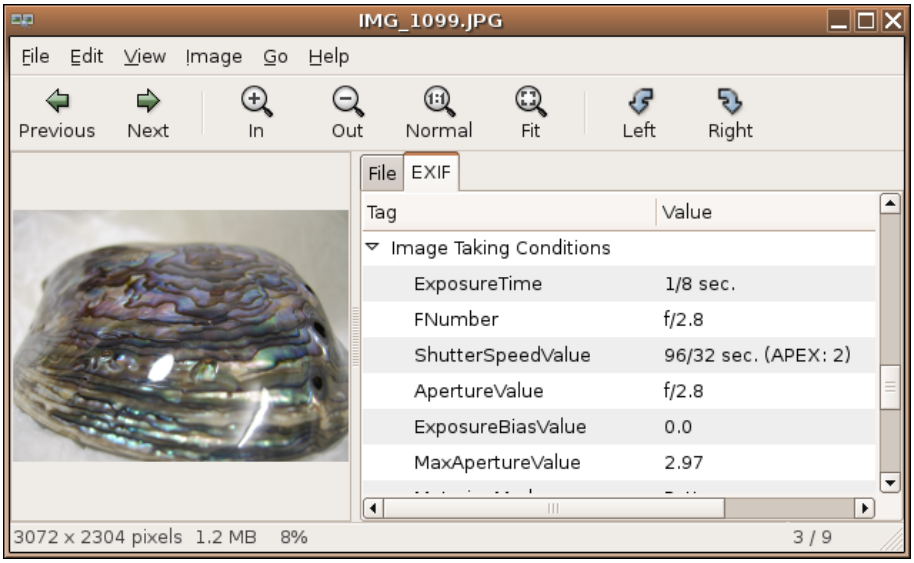


Figure 2.3: *EXIF Data displayed for a digital photograph, using Eye of GNOME Software. Here we see the meta data in relation to the Image Taking Conditions.*

ganise and inform the user about music. Some of the features of iTunes:

- Not only can the ID3 tags be used to organise both the virtual storage of tracks (i.e. how they appear in the iTunes Library), but also on the physical disk- iTunes will sort tracks into directories based on Artist and Album information gathered from ID3 tags.
- Lyrics and Album Artwork can be attached to tracks, adding to the musical atmosphere and experience.
- The ID3 tags are matched to the iTunes online store, to allow users to buy the music online by means of download.
- In iTunes 6, a feature named 'Just for you' has been added to the store, which uses meta-data collected about previous purchases and music in the iTunes Library to recommend new tracks to the user [6]

- **MusicBrainz** [25] is an encyclopedia of music that provides meta-data about music tracks to users, based on Relatable's TRM for acoustic fingerprint matching [38][32]
- **Gracenote CDDB** [16] is a large collection of meta-data of CDs. This can be added to by any user with a Gracenote client, and these clients are often included in music players (such as iTunes). The main purpose of this database is to allow users to put a CD into their computer, and their music player will automatically know what the track names and artist etc are.

Applications like Gracenote require that a user has entered the data into the database manually. This is a reasonable expectation, but it can be frustrating and tedious for users who have a lot of CDs which aren't in the database. This is a common problem with using meta-data. The process of inputting data is both time expensive and can result in ambiguous information. Attempts have been made to automate organising images into similar categories based not on user-inputted data, rather by using visual data [30]. Similar attempts have been made with music using tempo detection [12] as well as using a more explicit classification by detecting the Meter and Rhythm patterns [3].

2.3 The mobile environment

Risto Sarvas et al.[31] looked into the process of automatic meta-data creation for images taken on mobile phone cameras. This primarily involves using location based information to determine the 'type' of image, and hence categorise it (e.g. a photograph taken outside on a weekend in the day could be categorised- "weekend", "outside" etc...). In this mobile context, applying such techniques to music meta-data instead of images is what we are interested in.

Enhanced PDMPs (i.e. beyond just playing music on a portable) are relatively new, and have been explored by Nigel Warren et al. [35], using the music in combination with GPS to produce a navigational system. The idea being that the music volume will change from left to right or level out, depending on the direction the user should be going to reach their destination.

2.4 Contexts

The context in which a user is listening to their music is really the key focus in this project. Hong et al. split context up into three key areas [18], in an attempt to better define what areas to consider when programatically implementing context: Computing context (that of the hardware and software environments), User context (that of the specifics to each user) and Physical context (such as location, time and so on). These areas allowed us to better consider what and how to create the application. Splitting the physical context down further was discussed by Abowd et al.[1]. They defined the context into five key areas; Who, What, Where, When and Why. Each of these areas play a different part in defining context and we attempted to make use of these categorisations, especially the 'Where', 'When' and 'What'.

There have been various attempts to help refine a selection of music for a user based on a limited context. For example, applications like MoodLogic[1] and MoodPlayer[2] use the user's mood to try and select songs to play. This achieves a limited context examination as it is

[1] `http://moodlogic.com/`

[2] `http://www.moodplayer.com`

only based on mood. Another point to notice with this application is that it uses a database from all users; while this expands the 'knowledge-base' it also means the track selection is based on other users' opinions and may not be relevant to a user with obscure tastes. Other investigations into music retrieval based on user-only context (i.e. the preferences of a particular user, ignoring the physical context) include that of Hoashi et al.[17] who used tree-structured vector quantization techniques based on TreeQ[14] and their own relevance feedback methods to select music on a per-user basis.

Chapter 3

Environment

Here, we discuss what equipment and tools were used, how they were used and why.

3.1 Hardware

3.1.1 Mobile

Given that this system is intended for the mobile platform, and to be deployed to a handheld, we used a Portable Digital Assistant (PDA) to emulate the functionality of a PDMP. Using a PDA allowed us to easily introduce extra functionality for testing, the most useful being the connectivity features of the device.

Features The PDA we used was a HP iPaq Hx4700, one of the latest models. The Hx4700 features a large screen (480 pixels wide, by 640 pixels high), with VGA and True Colour (32bit). It also features a navigation touch-pad and 5 programmable soft-keys. It has both SD and CF-2 slots available, which we used for disk expansion.

Screen Size & Colour Because the screen is quite large for a mobile device, and given that it supports 32 bit colour, there was lots of scope for making use of these features. The size of the screen meant that placing of components was simpler than expected (with regards to designing for a small screen). From the evaluation (section 6), the placing seems to have been about right; users managed to navigate the interface fairly well. We made full use of the colour availability with full colour icons for buttons, which added to the design, leaving the overall appearance much more pleasing to the eye.

Connectivity In order to acquire certain data it was necessary to use an external source. In order to gather Global Positioning System (GPS) data, for example, we used an external module. Using the PDA's Bluetooth we could connect to the module and retrieve this data. Doing this meant that the PDA did not effectively increase in bulk – the GPS module could

Figure 3.1: *The Hewlett-Packard Hx4700 on which the mobile phase of the system was implemented, showing phase 1 software running.*

be carried in a backpack or pocket and hence did not interfere with the 'portability' of the device.

The PDA also has the ability to connect to other sources via WiFi (2.4gHz Wireless), Infrared and over a serial connection. Initially we had intended to utilise the WiFi connection in order to retrieve information over the Internet, and this would still be used if the desktop application was ported back to the PDA (see section 8.3, regarding further work). We used the serial connection in order to synchronise music collections from the PDA with the desktop, and also synchronise settings between the applications.

3.1.2 Desktop

Although one of the main aims of this project was to make use of and explore the mobile platform and music players, certain restrictions meant that we had to use the mobile in conjunction with the desktop. The main advantage of using the desktop in phase 2, was the fact that table manipulation and internet accessibility are far more advanced under .NET 2.0 than .NET 2.0 Compact Framework.

In terms of hardware, the desktop could be made up of any hardware components providing the software is supported. The only essential extras for our study were the PDA cradle, in order to use the serial connection for music transferral and settings synchronisation.

3.2 Software

The main development environment for both applications consisted of Microsoft's Visual Studio 2005. Although in the case of the PDA software, the project was initially created under Visual Studio 2003 and later ported to 2005 when certain issues relating to library incompatibility were resolved.

Visual Studio Visual Studio is a development environment for use with Microsoft's .NET technology. .NET is a conglomeration of different languages, which all get compiled to an intermediate language, which is then interpreted by a Common Runtime Library (CRL). The CRL means that multiple languages can be used in one application, and also that a compiled binary written in any supported language can be used on any machine that has a version of the CRL and .NET framework available. Although there is support for multiple languages, we only concentrated on one for the bulk of our code, although several other languages were employed to create 'satellite' or helper applications.

The language used was C#. This is Microsoft's competitor to Sun's Java, employing similar syntax and semantics. The overriding reason for using this language over others available (such as Visual Basic – VB) was down to a familiarity with Java, making the transposition to C# less painful than it could otherwise have been. There is also a lot of support for common advanced uses of C#; for example the existence of libraries for both GPS and ID3 manipulation , which (while not fully capable of our needs) meant the process of getting the underlying functionality of the software complete, was a lot faster than it could otherwise have been.

Satellites As we will see in section 4.2, it was necessary to create some satellite applications which carry out small tasks that are needed in the main application. Interpreted languages such as Python[1], PHP[2] and Javascript were employed in order to do this, as well as some HTML. For these applications a Linux based web server running Apache 2[3] was used which was connected to over the Internet. Google's mapping API[4] was also used in conjunction with these languages, in order to produce maps of where tracks were listened to.

[1] `http://python.org`
[2] `http://www.php.net`
[3] `http://www.apache.org`
[4] `http://www.google.com/apis/maps/documentation`

Chapter 4

Core Technical Aspects and Implementation

In this section we look at the underlying aspects of our system, and how they were used and manipulated in order to create the final application.

4.1 ID3

Firstly, ID3 Tags. As mentioned in the background (section 2.2), ID3 tags were invented by Eric Kemp and consist of small blocks of meta-data that are added to MP3 files. The meta-data provides information about the Artist, Title, Album, Genre and so on with relation to the track.

4.1.1 Technical Specification

The ID3 tag standard has progressed through various levels since its inception in 1996, and at its most current status stands at ID3v2.4 (Version 2.4). The way ID3 tagging works is to prepend (and sometimes append) header fields into the MP3 file. The specification, as given in [21] shows how the frames and headers are added in, and what defines a frame and its header. In figure 4.1 we can see the structure of the entire ID3 tag. Figure 4.2 shows this in practice.

4.1.2 Extraction

Given the specification above, we can extract ID3 tags from our music files. Using a prebuilt library, it was possible to extract ID3 tags from most MP3 files in our collection. In order to do this a simple call was made to the library which returns the relevant field (Title– '`TIT2`', in the example):

```
AssocArray result = new AssocArray(4);
ID3v2 tag2 = new ID3v2();
tag2.Load(path);
...
return tag2.Frames["TIT2"].Data(); //ID3v2.3+
```

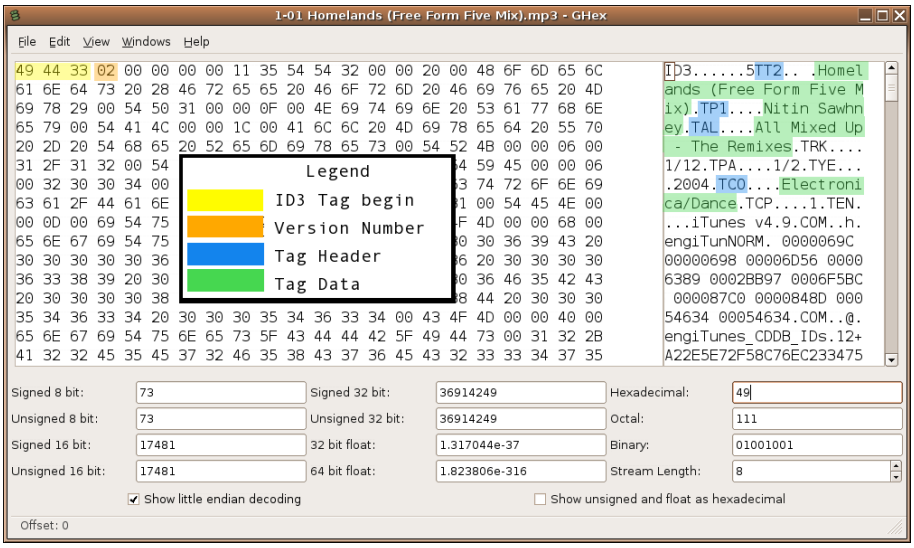


Figure 4.1: *Showing an MP3 file in hexadecimal format using GHex2 Software. We can see the ID3 Header ID (Hexadecimal- 49 44 33), followed by a version number (in this case 02, the default iTunes tag standard). Following that are the frame headers and their content.*

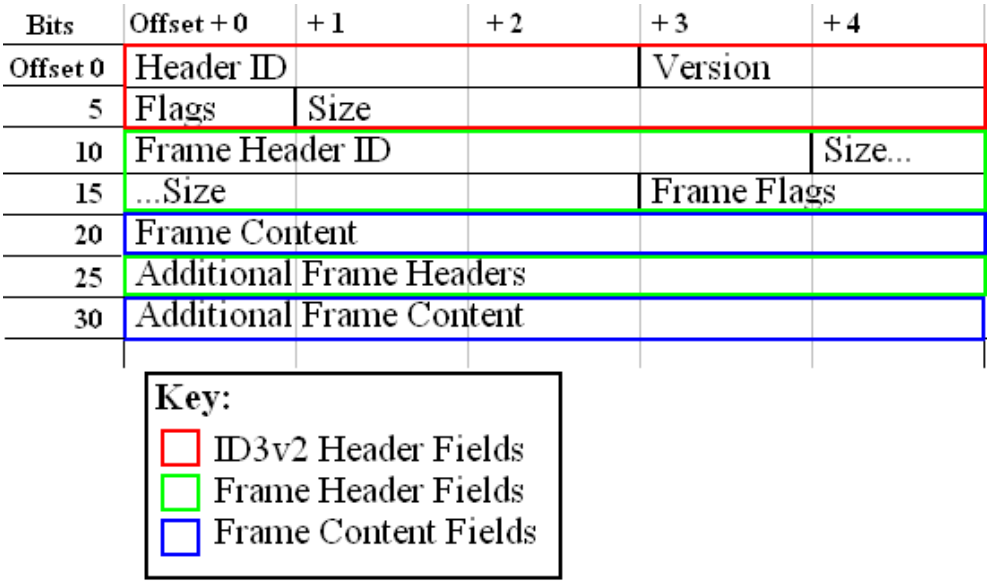


Figure 4.2: *Showing the buildup of data in an ID3 Tag. Adapted from [2]*

4.1.3 How ID3 was used

Having this meta-data to hand, which is common to the majority of MP3 files, meant that we could immediately access a wealth of data about users' listening habits, based solely on the 4 main tags; Artist, Title, Album and Genre. Following extraction of the fields, they were used on the portable application to display track names in the playlists, and the current playing track on the main screen (see the screenshots section– 5.2). This data was then logged to file as the user listened to music, for use later in the desktop application. The desktop application used the logged ID3 data to build up a database of track information, which allows the user to sort by these core fields (as one would expect to be able to with even a simple music player), along with our extended data.

4.1.4 Usage constraints

While ID3 tagging has been standardised and standard libraries have been made in order to take advantage of this, there were some data that did not conform. The main issue was that iTunes defaults to storing tags in ID3v2.2 format, which has been made obsolete by the newer versions 2.3 and 2.4. Unfortunately, most of the music in the test collection was ripped using iTunes, and hence had tags that were largely unusable. The library mentioned above did not support ID3v2.2 tags, and so it was necessary to adapt it to do so.

The largest difference between ID3v2.3+ and ID3v2.2 is that the frame headers differ in size. While v2.3+ uses 4 bytes, v2.2 uses only 3. Due to this factor amongst others (including the lack of a flags in v2.2 frame headers) the ID3Frame module had to be extended. To do this, an interface was created for which both the old ID3Frame module and our new one could implement:

```
public interface IID3Frame
{
    int Size();
    string Data();
    string ID();
}
```

This very simple piece of code saved re-writing the entire library by allowing us to use different version of the ID3Frame module. Thereafter, a copy of ID3Frame was made and modified to correct the header and frame header sizes.

```
byte[] id = new byte[4]; // original header id
byte[] id = new byte[3]; // new header id
...
byte[] header = r.ReadBytes(10); // original ID3 header
byte[] header = r.ReadBytes(6); // new ID3 header
...
```

Finding a solution to this problem did not entirely resolve the problem however. While most ID3 tagging utilities do conform to some standard of ID3, others do not. This became apparent as some tags were being extracted in a peculiar manner, showing data that did not appear to be in the headers. Windows Media Player adds tags both at the start and the end of the MP3 file. This was probably done out of implementational ease, as it is far easier to

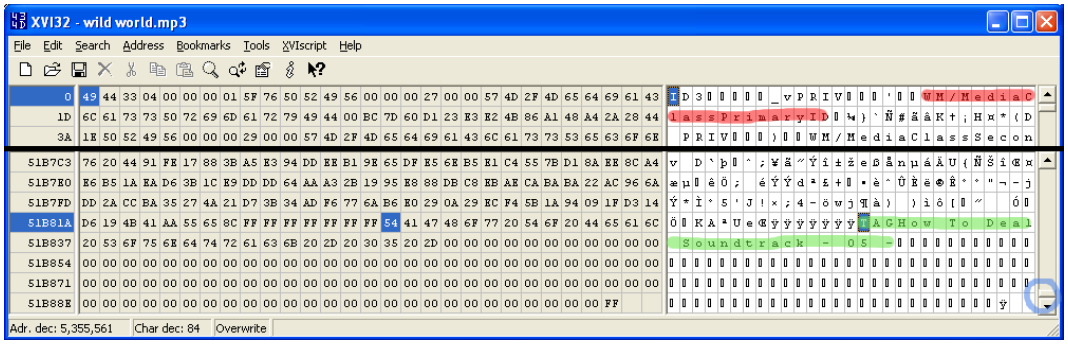


Figure 4.3: *Showing a hex breakdown of a file tagged by Windows Media Player (the red highlight indicates Windows Media meta-data), and a tag appended to the file (highlighted in green). A quick search in the hexadecimal data for either '3DI' (the footer identifier) or 'SEEK' (to locate an appended tag) proves fruitless, showing this ID3 tagging to be non compliant to ID3v2.4, which it claims to be (from byte 4)*

append data than to insert it. While the ID3 standard does support this, it has some rules on how it should be implemented. Referring to section 5 of the ID3v2.4 standard[21] tells us that we MAY append a tag, but that a SEEK frame must be given in the prepended tag in order to locate it. Section 3.4 also tells us that if we are tagging at the end of the file, a footer is REQUIRED, which is basically a copy of the header but with a reversed identifier (i.e. '3DI' as opposed to 'ID3'). In figure 4.3 we see a tag appended to a file, but with no corresponding footer or seek tags to locate it. Unfortunately standards compliance is not always adhered to in proprietary organisations, and in order to get round this problem, we used directory naming structure as a fallback for tagging.

Whilst not a perfect solution, there was only so much time that could be devoted to implementing all the ID3 tagging methods, and this was a solution that, while sub-optimal, achieved most of our requirements. iTunes (and other music applications) sort the music in their libraries into specific directory structures, and this enabled us to extrapolate data where ID3 tags were unavailable or unreachable. Below is an example of how applications like iTunes can store their music files.

```
c:\...\Library\Artist\Album\Title.mp3
```

So, where we had a blank field for an ID3 tag, we simply look at the file path and determine it from that. Obviously the problem here is that there is no indication as to the genre of the music, and so using this method removes a quarter of what we wanted out of track data.

4.2 GPS

The Global Positioning System (GPS) consists of a set of satellites that orbit the earth and provide properly equipped persons on the ground to locate their latitude and longitude to within 10 metres.

4.2.1 Technical Specification

In order to use the GPS system one needs a GPS receiver. For our project we used a Bluetooth enabled external receiver. The receiver itself gathers data from up to 8 satellites at any one time, though commonly only 4 are visible in built up areas. From information given by these satellites, the receiver can estimate (accurate to 10 metres) its latitude and longitude. The satellites themselves can be used to get a highly accurate co-ordinate reading, however commercial application of the GPS is limited by the American Military, and hence this inaccuracy is introduced.

NMEA Strings The standard output from the receivers are NMEA (National Marine Electronics Association) Strings. NMEA Strings are part of the standard NMEA 0183 controlled by the NMEA in the US[39]. These strings are used as a simple method of communicating locational data to various parties, originally intended for marine vessels but now used throughout the world due to the popularity of GPS. NMEA Strings consist of comma separated values ('sentences') of various types, an overview of which is given below.

```
1: $GPGGA,092204.999,4250.5589,S,14718.5084,E,1,04,24.4,19.7,M,,,,0000*1F
2: $GPGLL,4250.5589,S,14718.5084,E,092204.999,A*2D
3: $GPGSA,A,3,01,20,19,13,,,,,,,,,40.4,24.4,32.2*0A
4: $GPGSV,3,1,10,20,78,331,45,01,59,235,47,22,41,069,,13,32,252,45*70
5: $GPRMC,155127.000,A,5136.5943,N,00358.7267,W,0.96,131.24,171105,,*1D
```

[34] Sentence 1 shows fix data, showing the position of a satellite position fix. 2 is a position sentence showing longitude and latitude. 3 shows active satellites (which satellite numbers are used on which channels). 4 shows which satellites are visible, but are not necessarily being used. Sentence 5 shows Time, Latitude and Longitude – it is this sentence that we are interested in. Decomposing sentence 5 we get a better idea of what it all means:

`$GPRMC`	the sentence identifier
`155127.000`	the time in UTC (Coordinated Universal Time) format
`A`	this simply says that the sentence is valid
	(the alternative, somewhat counter-intuitively, being '`V`' for invalid)
`5136.5943`	our current latitude
`N`	hemisphere indicator ('`N`' = North, '`S`' = South)
`00358.7267`	our current longitude
`W`	to which side of the Greenwich Meridian Line are we? ('`W`' = West, '`E`' = East)
`0.96`	speed in knots over ground
`131.24`	degrees, course over ground (bearing)
`171105`	UTC Date
	two blank fields here, are the magnetic variation (degrees) and direction.
`*1D`	a checksum for the sentence

4.2.2 Use

Having the ability to access this locational information means we can add extra meta-data to each track's play log. In other words, every time a track is played we can now log exactly

where it was played. By examining where a track is played, we can start to define the current context for that track. This effectively addresses one of the five important fields of context as defined by Abowd et al.[1]

Tables There are various stages to using this data effectively, the first being the ability to log it in a format which is usable to both man and machine. Using degrees, minutes and seconds we can purvey the meaning to the user (provided they are familiar with this format, and what it means) and also retain accuracy for use by computers. Obviously the chosen format is not the most apparent to the average user, and this issue will be discussed later on in the evaluation. However, assuming we have the data logged on the portable software, it can then be uploaded to and manipulated by the desktop software. In order to manipulate it, there are several stages through which it must pass, and these are discussed in 4.2.3. The GPS data, split into its component longitude and latitude and used in a tabular format in the final desktop software provides a very quick and simple sorting facility. By sorting on column headers the positions which lie near one another become near one another in the table and hence we can quickly see where tracks were played in similar locations.

Maps The most obvious way in which to display data concerning co-ordinates is by plotting them. This allows us to get an overview of where we were and what we listened to 'at a glance'. Unfortunately simple co-ordinates points are not particularly self-apparent, and so in order to give them some meaning we plotted them to Google's maps facility. The intention here was to allow users to see what they had listened to and where, and with the added meta-data about their moods, give them a better understanding of their own listening habits.

4.2.3 Manipulation

Given that we can read a valid NMEA string we have to be able to parse it and use it. In order to parse it we modified an existing library which handles both connecting to and receiving from the GPS Unit, and also handles NMEA string parsing. Figure 4.4 shows the code needed in order to call an instance of the receiver and what it does with the data it receives.

What this actually outputs is a formatted co-ordinate, showing the latitude and longitude in degrees minutes and seconds:

```
51°36.9930"N, 003°58.1584
```

This is then passed directly into our log for use with our mapping methods. These methods, which make use of Google's Map API[15], were implemented first in Python, in order to examine how the API works, and then later ported to PHP for dynamic generation with an XML file.

```
try
            {
                nmea = new NMEAReceiver(8); // 1
            }
            catch (IOException ioe)
            {
                lbDebug.Items.Add(ioe.Message); // 2
            }
            if (nmea != null)
            {
                nmea.PositionReceived += new
  NMEAReceiver.PositionReceivedEventHandler(gpsEvent); // 3
            }
```

Figure 4.4: *Here we see the creation of a new receiver on COM port 8 (1), the error handling if opening on this port fails (2), and the method to be called when we receive a new position event (3). The* `PositionReceived` *method simply updates a variable with the most recent valid GPS co-ordinates.*

4.2.4 Conversion and Display

So, how did we achieve the conversion from this human readable degrees, minutes and seconds notation, to something that the Google API can read and interpret? Several tools were used, discussed below.

1. The data was written to a C# DataTable with some added HTML formatting (for use in the final display), and using the inbuilt methods, converted to an XML string

```
StringWriter s = new StringWriter(); // new stream to write to
DataTable dt = dtViewDynamic.Clone();
// create a new table based on an old one

// ... fill table ...

dt.WriteXml(s); // write the table to the stream as XML
s.Flush(); s.Close(); // tidy the stream
string s2 = s.ToString(); // convert it to a string
```

2. Next, encode the XML string to base 64. The reason for this is so that we can pass it as a POST parameter to the PHP script, and base 64 encoding removes the need for escaping characters

```
byte[] encbuff = System.Text.Encoding.UTF8.GetBytes(str);
// convert the string to a byte array
return Convert.ToBase64String(encbuff);
// return a base 64 string encoding of the byte array
```

3. Then, create a new HTML document that contains a hidden and automatically submitting form. The form contains only one field – that of a text area – and this takes the value of the base 64 encoded XML string. This form gets posted to our PHP script.

```
<html>
<body onLoad="javascript:document.forms[0].submit()">
        <h1>Please wait...</h1>
        <form method=POST action=
                "http://maps.synfinity.net/intuiTunes/mapIt.php">
                <textarea name="xml" style="display:none">
                        ENCODED STRING HERE
                </textarea>
        </form>
</body>
</html>
```

4. The PHP script takes the input encoded string and decodes it back to plaintext:

   ```
   $p =& new xmlParser(base64_decode($_POST["xml"]));
   ```

   This then gets saved on the server and read into an XML parser. The parser outputs an array structure representing the XML tree. We can then extract the GPS co-ordinates and other data from this array.

   ```
   foreach ($p->data[0]['child'] as $track){
   $artist = str_replace('"','\\"',$track['child'][2]['content']);
   $title  = str_replace('"','\\"',$track['child'][3]['content']);
   $album  = str_replace('"','\\"',$track['child'][4]['content']);
   $genre  = str_replace('"','\\"',$track['child'][5]['content']);
   $gpslon    = $track['child'][8]['content'];
   $gpslat    = $track['child'][9]['content'];
   $mood   = str_replace('"','\\"',$track['child'][7]['content']);
   // ...
   }
   ```

5. Given that the GPS co-ordinates are still in a format which is unusable to the Google API, we must convert it. This is done by a fairly tedious process of extracting parts of the co-ordinate and multiplying and dividing them in order to decimalise them. So, our original latitude-longitude tuple;
   `(51°37.0163, 003°58.1612)`
   becomes;
   `(-3.9693533333333, 51.61693833333333)`
   This decimalised format is what we pass to the Google API, which uses JavaScript:

   ```
   markers.push(createMarker(new GPoint(-3.9693533333333,
   51.61693833333333),"ARTIST","TITLE","ALBUM","GENRE",
   "<font color=#8000FF>MOOD</font>"));
   ```

The resulting output is shown in figure 4.5.

4.3 Moods

One of our main interests with regards to the User's context is that of their current mood (the 'What' from [1]). The theory here is that a user will change what they listen to based on what mood they are in; many studies have attempted to detect elements of music to automatically suggest music to the user. Feng et al.[12], based their mood-music matching on tempo and articulation. Field et al.[13] used a meta-data schema with keywords and integer ratings to match specific known tracks to a mood or category profile.

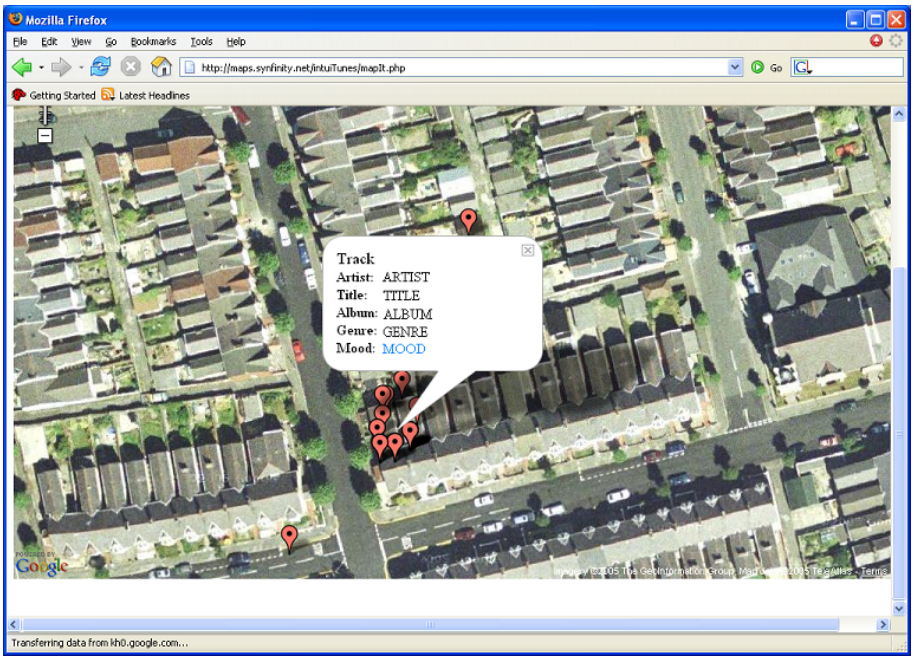


Figure 4.5: *The resulting Google map, with a bubble showing track data for one track in the database, and other tracks markers shown.*

4.3.1 Definition - What defines mood?

So how do we define a mood? How can we provide an interface to users that is both accurate for a particular user and 'portable' across users – in other words, how can we find a way to define a mood and apply it to all users? Unfortunately there is no single answer to this question. The initial problem being that for each person, a different mood will be described differently by their music tastes. Thus, we cannot just sample a set of users and say, for example, all heavy metal is angry music – for some it is the music of choice for when they are in a very positive mood. Sampling of this fashion would prove fruitless.

Instead, we need to look not at defining which moods relate to which music, but rather, how can we let the user associate their mood with specific music. In this way we allow for different tastes to be represented. Looking at the research of Rosalind Picard[27] who describes mood by valence and arousal, allowed us to conceive a mood model. Put more simply, valence models how positive we are feeling, ranging from 'happy' (high valence) to 'sad' (low valence) and arousal is to what extent we feel this (e.g. high arousal and high valence would be very happy- possibly a mood induced by a party, for example). As the terms 'valence' and 'arousal' are not used in everyday communication of our moods to one another, we thought colour to be an appropriate model. The horizontal plane, ranging from blue to red, modeling valence and the vertical, from intense colour to faded out – arousal. Figure 4.6 shows our initial mood panel.

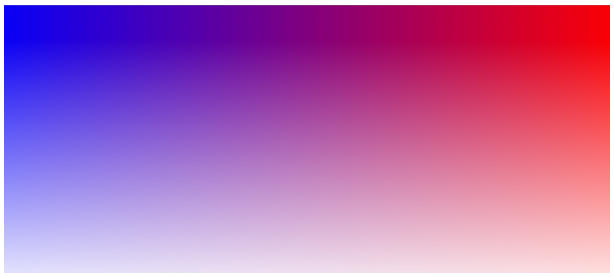


Figure 4.6: *The prototype mood panel, with colour representing valence and intensity representing arousal. This model turned out to be ineffective.*

After carrying out some paper prototype analysis with a set of potential users it was apparent that the model was inappropriate. An in-depth look at the evaluation can be found in section 6.2, however, we shall summarise here what we found and the replacement strategy used.

- **User's Ideas** – Before showing participants the prototype, we asked them how they though mood could be represented. Generally, participants thought that a good way to represent different moods was either by association to colours, or by 'emoticons' (as used in online chat clients) that characterise various moods or emotions. This confirms our initial ideas that colour is a fair model for mood.

- **Our Intentions vs. Theirs** – With a set of positions on the prototype mood-panel in mind, we asked participants to point out where they would place various moods, ranging from depressed to 'wanting to dance'. The participants didn't really agree on one area of the mood panel to represent each of the moods that we asked them to try and mark. Each user had a different idea of what each colour meant to them. One participant even said that they found it impossible to represent 'content' with the Red to Blue fade scheme. This means that our model does not suffice.

- **Bettering the Model** – We then asked them how we could improve the model. All the participants agreed that the colours were too specialised. What to one participant was the colour for anger would mean happiness for another. As the system will be used on a per-user basis rather than as part of a collective user-base, this fact does not matter too much. However, participants felt that the colour range was limited and would rather have a less 'analog' way too choose a mood – they would rather have a limited number of set moods and be able to define colours for them.

From this simple evaluation we determined that a better model was to allow users to select 8 moods (including a required 'Normal' mood), and attach those mood labels to 8 different colours. This allowed the users to personalize and qualify what a colour meant to them in terms of mood, avoiding ambiguity introduced by the linear fades of our initial model. Figure 4.7 shows an updated model with sample data loaded.

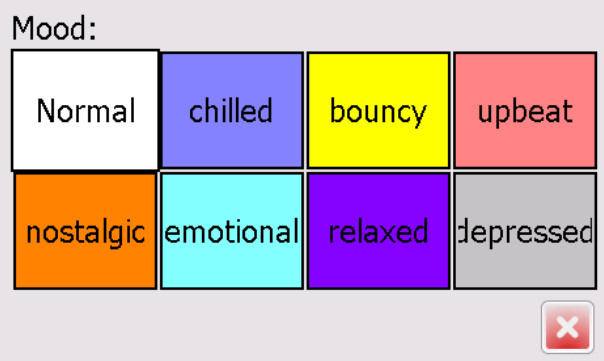


Figure 4.7: *The new mood model, with user 8 defined mood labels and colours, allowing both concrete specification of mood and personalisation. ('Normal' has been selected here, the default)*

Figure 4.8: *Tracks showing only one mood even if they have several attached to them.*

4.3.2 Uses

Obviously having this extra meta-data field allows for some extra insight into a user's musical habits. Our users selected their mood as it changed and the tracks that played were logged along with this field, and the others mentioned in the sections above, to file. Uploading these logs into the desktop application allowed the users to visualise what they listened to and when.

In order to utilise the mood colours, we used the column in the database table which holds the mood data, and coloured the cells to match that of the related mood. For example, all tracks that had been listened to in an 'angry' mood (with the associated colour, red) will have a red cell in their mood column (Figure 4.8). Unfortunately, problems arose when tracks had been listened to in multiple moods, only showing the first mood that the cell-painter encountered.

To solve this problem, it was necessary to extend the C# grid cell and column classes, to overload the cell-painter and hence show all the colours which were related to that track.

```
public class DataGridViewTextBoxMoodCell : DataGridViewTextBoxCell
    {
        ArrayList _Colours;

        public DataGridViewTextBoxMoodCell() : base()
        {
            _Colours = Colours._Colours; // 1
        }

        protected override void Paint( ... ) // this line has been truncated for display
```

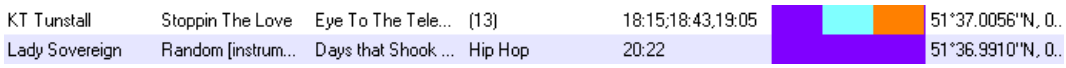


Figure 4.9: *A track with more than one associated mood, represented by multiple colours in the mood column cell.*

```
    {
        base.Paint( ... ); // this line has been truncated for display
        int i = 0; // 2
        foreach (string mood in value.ToString().Split(';')) // 3
        {
            int width = cellBounds.Width / value.ToString().Split(';').Length; // 4
            graphics.FillRectangle(
                  new SolidBrush((Color)_Colours[Int32.Parse(mood)]),
                  cellBounds.Left + (width * i), cellBounds.Top, width,
                  cellBounds.Height); // 5
            i++; // 6
        }
    }
}
```

At 1 we are setting up our array of colours. These are provided in a static class which gets written to on initialisation, with our list of user defined colours. At 2 we are resetting which mood we are currently looking at (if there are multiple moods for this column). 3 starts an iteration through the moods, which are stored as a semi-colon separated list of integers. At 4 we set how wide this colour should be, based on the size of the cell to colour divided by the number of moods we have to fit in. 5 carries out the painting of the cell, keeping the colour within the vertical bounds as well as our limit for width. 6 simply increases the current mood counter.

In combination with an extended DataGridViewTextBoxColumn, we overrode this column's paint methods and the resulting colour display can be seen in figure 4.9. This allows the user to get an overview of their mood and track relationships at a glance, as with the map data.

Chapter 5

Implementation

In this section we will look at how the software was put together, including design choices and features.

5.1 Features

The main focusses of the desktop software were to allow users to manipulate the meta-data, be able to view it, and then create contextual playlists from it. One other aim was to create a recommendation system to enhance and extend the created playlists.

5.1.1 Meta-Data Manipulation

The first target was to allow users to load up the meta-data log files into the system. This was achieved through the use of the XML format when logging data– the ability to save to and load from XML is incorporated into the C# DataSet and DataTable classes, allowing use to quickly load up vast amounts of data, and process it very easily.

Given that we could load the logs into a DataTable, it was then necessary to process it. 'Processing' was carried out in the following fashion:

- New data (a track that the table did not have any data about) had to be added to the table, using the file-path as a unique identifier. The ID3 data was taken directly from the logs, as was the data for the mood in which it was listened to, the GPS co-ordinates, and the time. If this entry was a completed track (as opposed to one which was started and stopped/skipped before completion) then we set the play count to 1, otherwise 0.
- 'Update' data, (a log entry for a track that was already in the table) was used to append moods, times and co-ordinates (if they differed from current data). Again, where a track had been fully played, the play count was incremented.

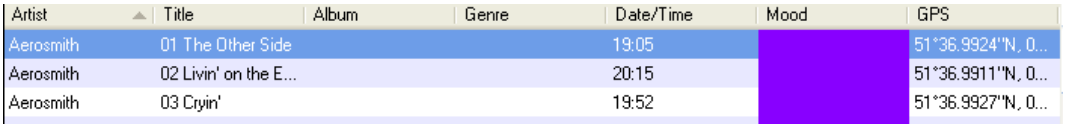


Figure 5.1: *The layout of the track grid, with track data on the left and our added dynamic data to the right.*

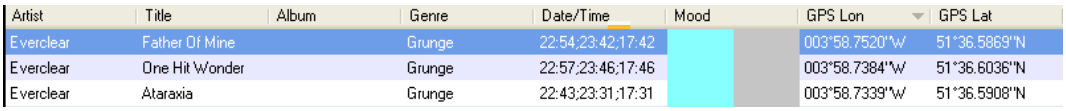


Figure 5.2: *The layout change after GPS has been selected – the column splits showing latitude and longitude separately for better sorting*

Having loaded all the data into the table (and rendering the mood cells as described in section 4.3.1), it was available for the user to browse. The table layout shows all columns, starting with ID3 data, with our added meta-data following (figure 5.1). The table could be sorted by any of the fields, with multiple field sorting available. This allowed the user to, for example, sort by mood but keeping alphabetic sorting on artist then sub-sorting alphabetically on album and a further alphabetic sub-sort on track title. In this manner, the tracks were kept in a logical sub-order – Artist, Album, Title.

Due to the nature of GPS co-ordinates – co-ordintaes being a tuple, as opposed to a datum – sorting by GPS required the co-ordinates to be split into latitude and longitude. This event occurs when a user clicks on the GPS column header, and is reversed when a column header for a different field is selected (figure 5.2). Splitting this field allowed numeric sorting to be based primarily on either latitude or longitude, with a sub-sort on longitude or latitude (respectively) and further sub-sorting as described above.

5.1.2 Context Playlists

Simple sorting in the fashion described, allowed like-data to be grouped together in the table, and hence allow the user to locate similar tracks. Being able to locate similar tracks means that a context can be built manually by selecting a few rows of the table that are nearby and add them to a new table, to create a playlist.

In order to create a context (a playlist of similar tracks), the user selects a track which fits that context and presses the add button. If there is no current context, a new one is created and the user is prompted to name it - for example, one of our test users in the evaluation created a context called 'working at night'. This context is then expanded by selecting similar tracks, based on mood, location, time or a mixture thereof, which is assisted by the sorting mechanisms.

5.1.3 Recommendations and Last.FM

To expand the contexts, we used a web-service from Last.FM[1]. Last.FM is a web site that allows users to sign up and submit what they are listening to in real time (or in batch, for periods when they are offline). This data is used along with collaborative filtering techniques in order to match artists – giving a relationship between artists:

```
Artist X is similar to Artist Y
```

Last.FM provide this data free to non-subscribers and we used it in order to expand the contexts based on a selected artist.

The user selects a track in the context grid, and can either right click or use the 'Context' menu on the menu-bar to access 'Get similar by... Artist'. This initiates an HTTP GET to the Last.FM web-service with the selected artist as a parameter, which returns an XML file (figure 5.5) containing a list of similar artists to the input artist. The top 5 are extracted from this list by loading the XML into a C# datatable. The user's local music store is then scanned for any of these 5 artists, and upon finding a match, the list of tracks is added to the 'Recommendation' panel (figure 5.4).

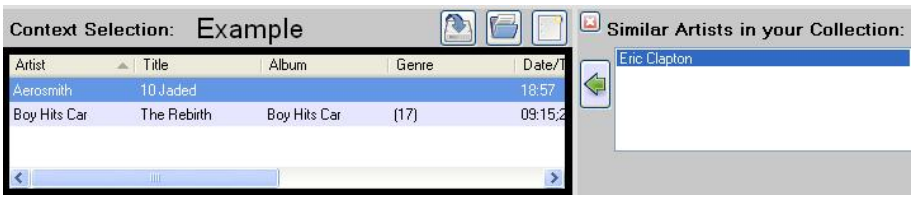


Figure 5.3: *Here we have retrieved a list of similar artists to 'Aerosmith' and found that 'Eric Clapton' is in the local music store*

5.1.4 Display

The last use for our added meta-data is to enable the user to visualise it. This will allow them to get an overview of their listening habits, and locate similar tracks by looking at

[1] `http://www.last.fm`

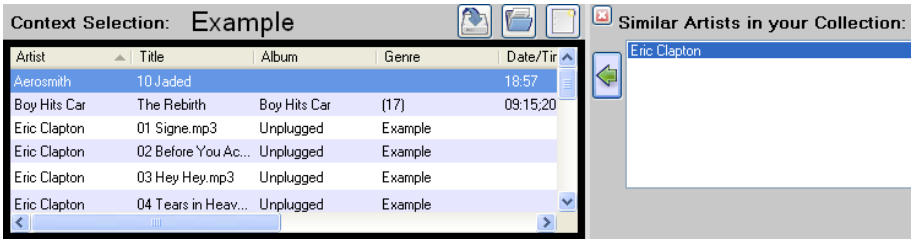


Figure 5.4: *The user can now select this and add the tracks to the context*

```
<similarartists artist="Zero 7" streamable="1" picture="http://static.last.fm/proposedimages/sidebar/6/1319/2608.jpg"
mbid="c23b637b-97c6-41eb-8ef6-6c724efc80a8">
    <artist>
        <name>Air</name>
        <mbid>cb67438a-7f50-4f2b-a6f1-2bb2729fd538</mbid>
        <match>100</match>
        <url>http://www.last.fm/music/Air</url>
        <image_small>
        http://static.last.fm/proposedimages/thumbnail/6/2/3782.jpg
        </image_small>
        <image>
        http://static.last.fm/proposedimages/sidebar/6/2/3782.jpg
        </image>
        <streamable>1</streamable>
    </artist>
```

Figure 5.5: *An XML file containing similar artists to 'Zero 7', taken from Last.FM's web service.*

the data that lies near by on a visualisation. In order to do this we took the three fields and created paper prototypes of how we thought they could best be represented. These prototypes were evaluated (see section 6) and modifications were made, the result of which is demonstrated below.

- **Time** – Time was represented using a clock like diagram, with numbers for hours marked at intervals. However, due to the nature of the data – covering the full 24 hours – we needed to be able to represent any hour. This was achieved by using a spiral, with the larger regions of the spiral devoted to the hours of the day where the average users are most likely to listen to tracks; and the smaller areas assigned to the early hours of the morning (figure 5.6).

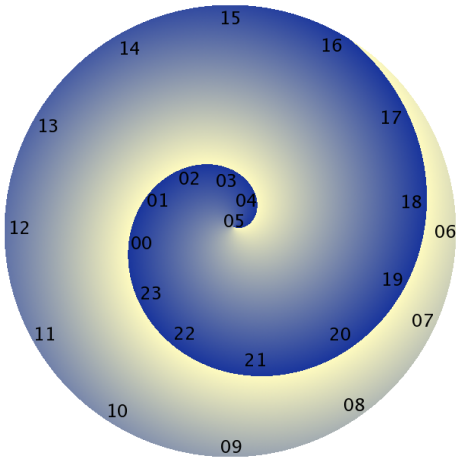


Figure 5.6: *The Time-Spiral, representing the full 24 hours in a clock-like fashion. Larger areas are used for parts of the day where more music is likely to be logged.*

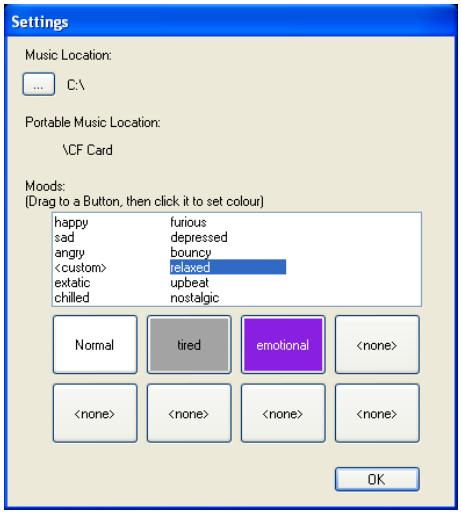
- **Location** – Using the GPS co-ordinates we plotted tracks on Google Maps (see section

4.2). The most simple and effective manner to display locations is of course with a map, and Google's API provided an elegant method to do this in conjunction with the GPS data.

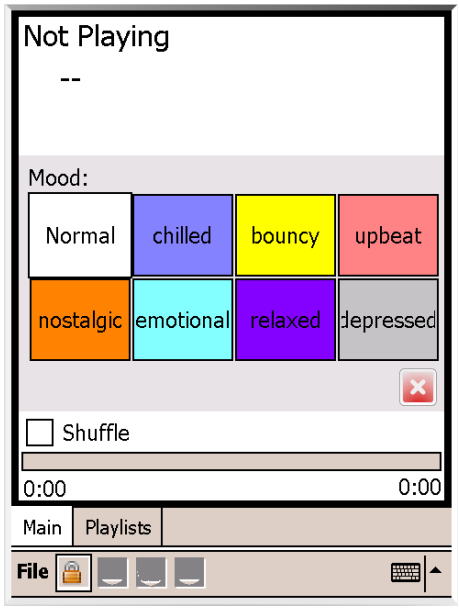
- **Mood** – To overview mood, the Mood column of the data grid could be sorted, grouping like mood-colours together. With our first attempt at mood representation using a colour fade (see figure 4.6) it was possible to plot the tracks directly back onto the mood panel, as we were using a co-ordinate system to locate a given mood. However with the 8 block mood selection, this would prove to look messy and the fact we are using distinct colours (as opposed to a fade) means this column sorting method that is more suited to the task.

5.2 Screenshots

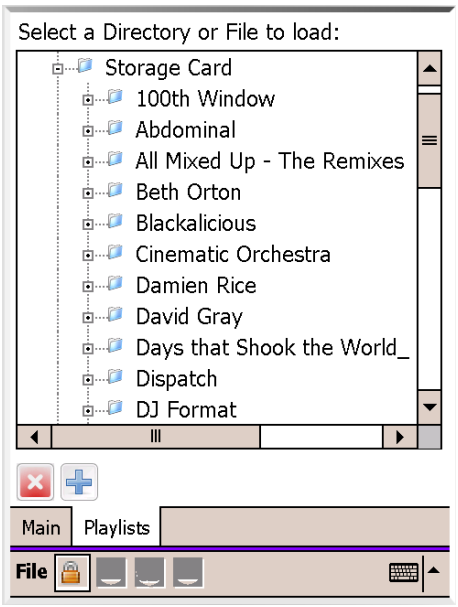
Here we walkthrough the application with a series of screenshots.



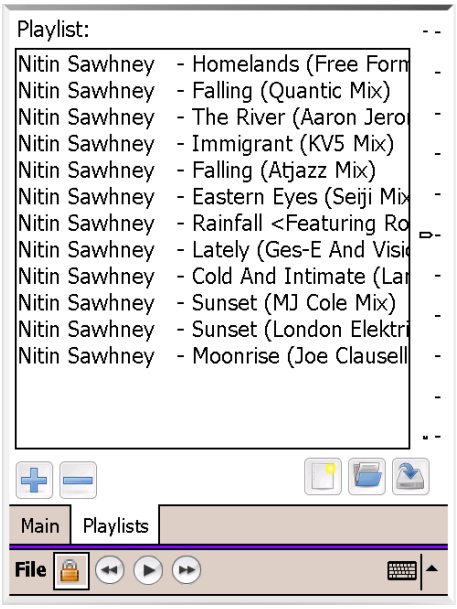
Firstly, the user is presented with a settings dialog in order to set their moods and colours by dragging and dropping from the list to the buttons. These settings are then transferred to the PDA.



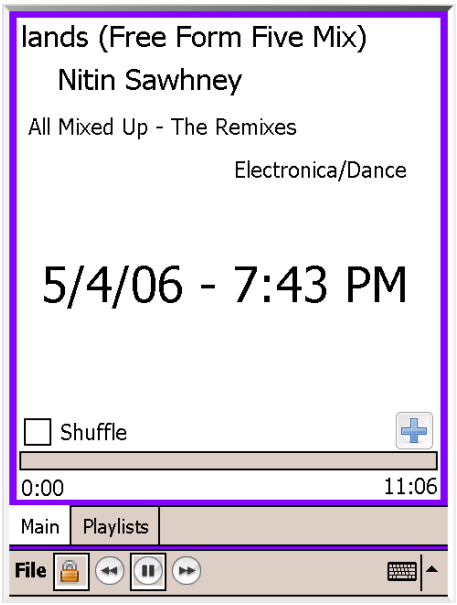
The user then starts the PDA application and the settings are carried through, showing the mood panel as it was on the desktop.



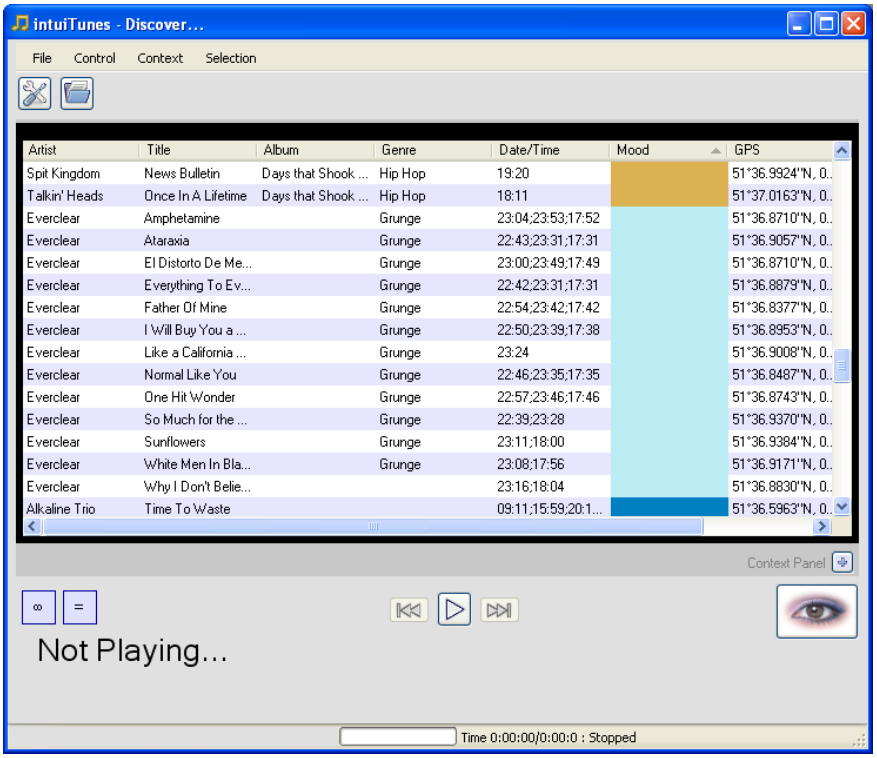
The user then selects the tracks or directory of tracks and adds them to the playlist.



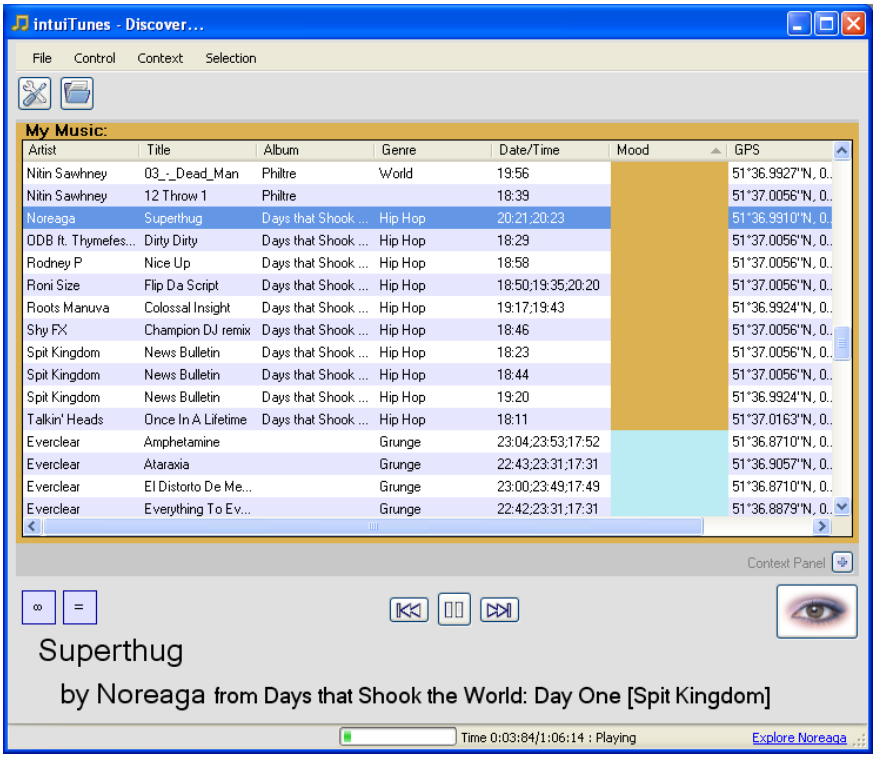
The user can modify the playlist by using the slider to move tracks, and the plus and minus buttons to add or remove tracks. The lists can also be saved, cleared and loaded.



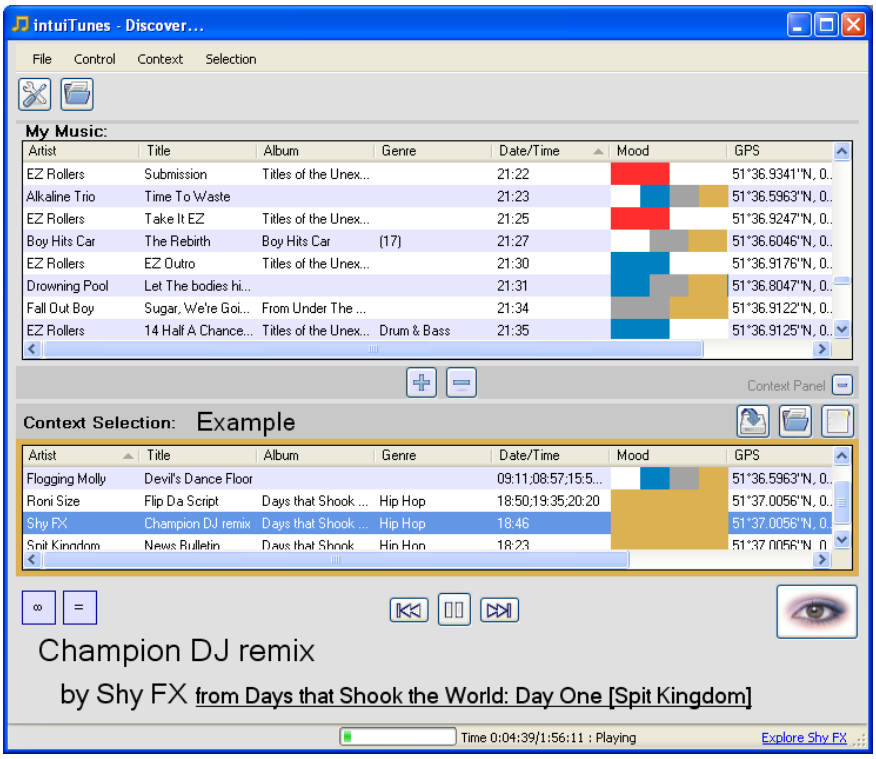
The application can then be used as a normal music player, silently logging data in the background.



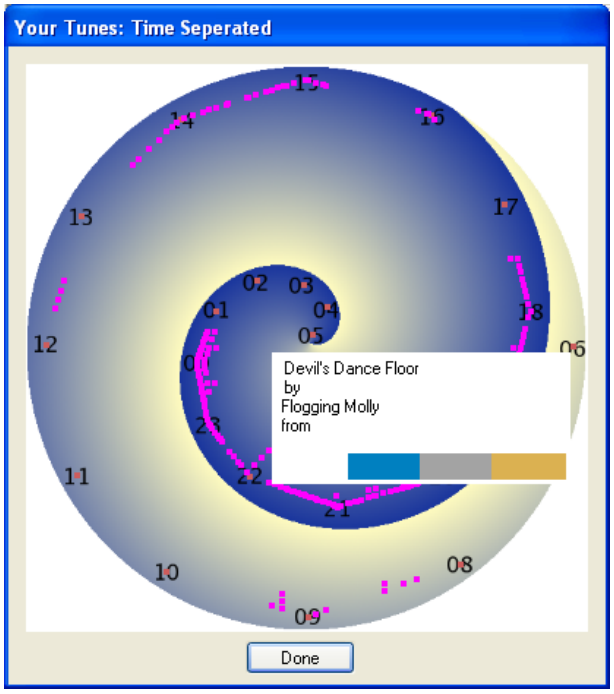
This is the main screen of the desktop application, after the user has loaded up some log files transferred from the PDA.



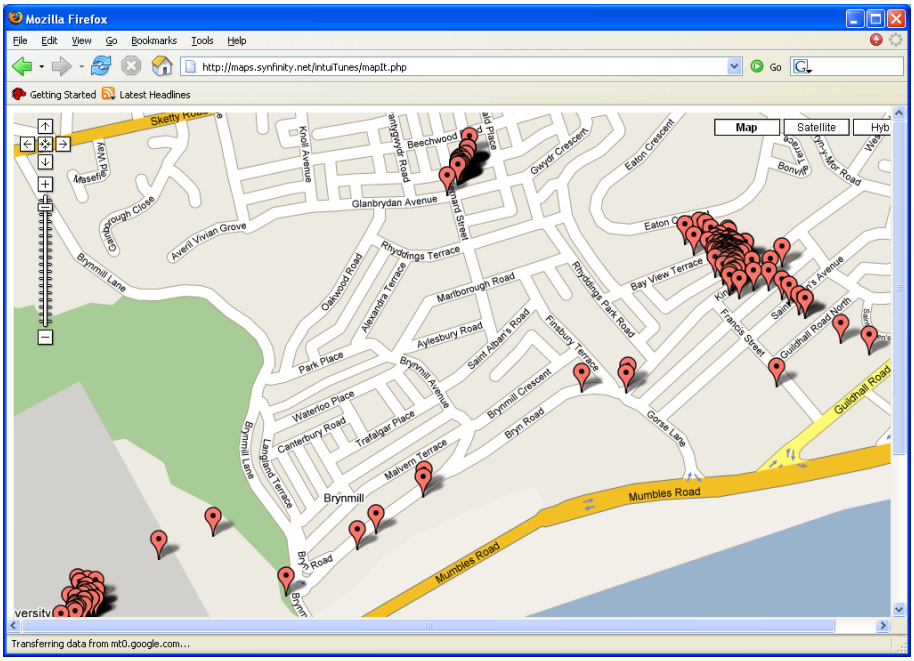
The back panel re-colours itself to match the current track's mood. Status information about the track is shown at the bottom.



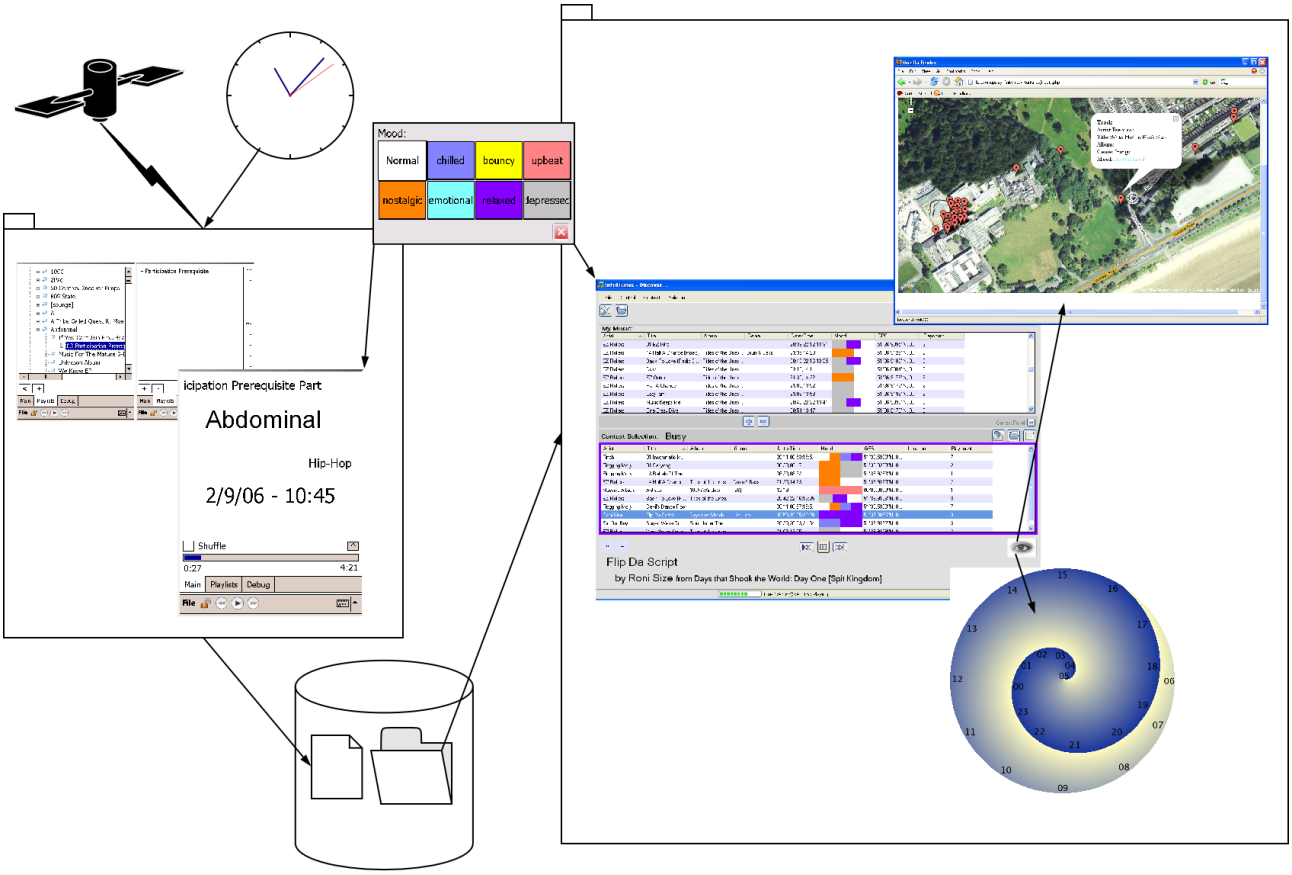
Contexts can be created and then played.



The time overview, showing a hover box with track information.



The GPS data overview, with clickable markers showing track information.



An overview of the system with screenshots, the PDA on the left and the desktop on the right.

Chapter 6

Evaluation

In order to establish whether our application achieved its aims, we carried out several evaluation steps. In doing this we discovered problems that should have been immediately obvious but were hidden by familiarity – in other words, the fact that we knew how to operate the system meant that we could not always see its failings, and this is rather the point of an evaluation.

6.1 A Discussion of Methods

There are many tried and tested ways in which software can be effectively evaluated, ranging from very quick and simple methods to those which require a lot of time and resources. For the project, we used a few methods which worked with the time scale available, and while perhaps more time should have been devoted to the evaluation process, what *was* achieved during this time raised some interesting points for consideration.

There are several things to consider when selecting an evaluation method. It is important to gain enough information from the users in order to correctly interpret what is happening, and this can be done in various ways. Some of these methods are better than others, and indeed, some can have the opposing effect, 'injecting' information into the users that is not really there (for example by asking leading questions) – known as experimenter biasing.

To gather enough information to perform a valid evaluation, especially with interface design, we need to know what the users are thinking. By simply watching users, the experimenter may become frustrated if they are not performing the task to expectations (especially if the feature being tested is the brainchild of the experimenter). To conquer this problem, we can ask the users to talk through what they are thinking and doing as they do it[10]. As discussed by [22] however, this is not an optimal solution as the user may feel self conscious and uncomfortable while doing this or just stop talking. If the experimenter talks through it with them then they may again introduce bias.

One way in which to avoid experimenter biasing, and to avoid the issues above, is by asking more than one user to participate at a time[26]. This allows the two users to effectively combine their thoughts to solve a task. If one of the users is finding the task difficult, instead of the experimenter prompting them, the users engage in dialog which hopefully brings ideas to light.

Another completely different method involves removing the experimenter completely and simply have the users carry out self reporting. This method, inviting users to log their thoughts at intervals is known as Experience Sampling[23] and has been used in mobile evaluations, for example by Csikszentmihalyi[8].

6.2 Paper-prototypes

The first part of the process was to evaluate our ideas. In order to do this we produced some low fidelity paper prototypes of the type of screens we intended to use for our visualisation of the three extended meta-data types (i.e. Time, Location and Mood). These prototypes were used along with a set of questions. A copy of the questionnaire used can be found in appendix A.

The participants consisted of three pairs, two who were non-technical (i.e. have no technical background or experience) and were non-PDMP users (i.e. they only use music players in their homes, which aren't portable), two were non-technical PDMP users, and two were technical - one of which was a PDMP user and the other wasn't. These groups were selected to represent a range of potential users, including the early adopters (the technical group) to those that may be enticed to switch (current users) and those that may start to use the device as a result of enhancements.

6.2.1 Participants' Ideas

The first step was to evaluate how the participants thought the three data types could best be represented. In order to do this, we asked them to describe their ideas. Following is a summary of what they thought.

- **Time** – Participants thought that the most obvious way to represent time was either with a clockface, which could be 'segmented' to clearly define hour boundries, or to use a time-line style diagram.
- **Mood** – Generally, participants thought that a good way to represent different moods was either by association to colours, or by 'emoticons' (as used in online chat clients - figure 6.1) that characterise various moods or emotions.
- **Location** – The two main ideas that came out here were to either use a Map (as this is what we are used to using to locate places) or to simplify the representation of a

location into a categorisation of location. i.e. use icons to represent 'home' or 'in the car' or 'out shopping' for example.

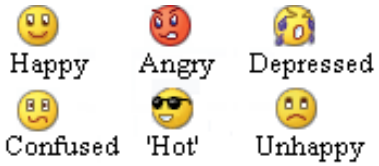


Figure 6.1: *'Smileys' or emoticons, could be used to represent mood. These emoticons, taken from Microsoft's MSN Messenger Client are used in text based chats.*

6.2.2 Evaluating our ideas

Given that the participants' ideas were not so different from our own, we proceeded to ask them to which of our mock-ups they would use to represent certain things. For example, the participants were asked to show five moods on the mood panel ranging from 'Depressed' to 'Feel like dancing', and similar questions for time and location covering a range of possible data. Their thoughts, summarised:

- **Time** – All three groups of participants preferred the Spiral (figure 6.2) over the slightly more abstract day to night fades (figure 6.3).
- **Mood** – As discussed in section 4.3.1, we saw that the participants found it difficult to make use of the model we created, often appearing confused by the model. Some participants decided on one area of the panel to represent a mood, then after a discussion with other the participant had forgotten where they had originally said, demonstrating the ambiguity of the design.
- **Location** – All three groups of participants agreed unanimously that the easiest map to locate the 3 places was the simple picture-map (figure 6.4A). Although, one participant commented that the picture-map doesn't show the beach particularly well (as it simply shows land then sea). Another point that came up was that it might be good to use a satellite-map when the picture is zoomed in closer.

6.2.3 Critical Analysis

The next step was to ask the participants what they thought (if anything) was wrong with the designs, and how they could be improved. From the actions described in 6.2.2, we can see there were some obvious problems. Summarised below is an overview of what participants thought

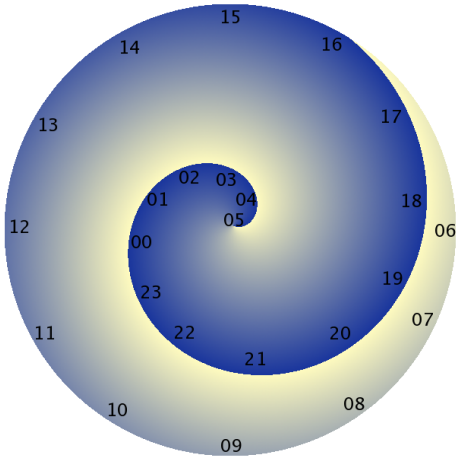


Figure 6.2: *The time spiral that was decided on for the final time display panel.*

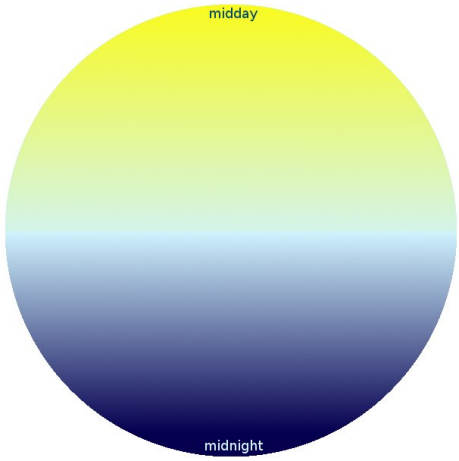


Figure 6.3: *This time display, showing day fading to night with the majority of the space devoted to midday, was rejected from the paper prototype evaluation.*

Figure 6.4: *The three possible mapping types, A showing a flat map, B with a hybrid satellite and flat map, and C – just satellite imagery. It was agreed that the flat map was the easiest to make sense when zoomed out. Images from Google Maps*

- **Time** – The night to day fade (blue and yellow) was too ambiguous, with the supposed centre of the day's events being unrealistic- a few of the participants commented that they regularly listen to music into the early morning, and so there was not enough space devoted to it. They all agreed that the spiral representation was more appropriate as it was closer to a well known time display already (i.e. that of a clockface). Other methods mentioned included a timezone selector style fade (as is used on a cartesian map of the earth to show day and night.
- **Mood** – As we saw in section 4.3.1, a better solution discussed was one in which users could select their own colours, and attach moods to them to provide a more definite choice.
- **Location** – Participants thought that the flat-map was easier to read than the satellite version (as mentioned under 6.2.2), but it was thought that the satellite map might be interesting when zoomed in close (although one participant commented that "I don't need to see a picture of my house, I already know what it looks like").

6.2.4 Summary

From this study it was clear that our initial ideas about what made a 'good' display were flawed. Trying to be too abstract for the time display and too general for the mood panel achieves negative effects. It is better to tailor the interface to each user and use well known themes or ideas to portray commonly acknowledged information. In response to the comments the following changes were made to the displays and the probe:

- The probe mood panel was converted to 8 mood buttons which are coloured and named to reflect the intended mood. These are set by the user in the desktop application prior to deployment with the probe, providing a matched colour-name mood between both applications, tailored to the specific user. (See section 4.3.1).

```
************** Exception Text **************
System.NullReferenceException: Object reference not set to an instance of an object.
   at intuiTunes_Discover.fmDiscover.bntSaveContext_Click(Object sender, EventArgs e)
   at System.Windows.Forms.ToolStripItem.RaiseEvent(Object key, EventArgs e)
   at System.Windows.Forms.ToolStripMenuItem.OnClick(EventArgs e)
   at System.Windows.Forms.ToolStripItem.HandleClick(EventArgs e)
   at System.Windows.Forms.ToolStripItem.HandleMouseUp(MouseEventArgs e)
   at System.Windows.Forms.ToolStripItem.FireEventInteractive(EventArgs e, ToolStripItemEventType met)
   at System.Windows.Forms.ToolStripItem.FireEvent(EventArgs e, ToolStripItemEventType met)
   at System.Windows.Forms.ToolStrip.OnMouseUp(MouseEventArgs mea)
   at System.Windows.Forms.ToolStripDropDown.OnMouseUp(MouseEventArgs mea)
   at System.Windows.Forms.Control.WmMouseUp(Message& m, MouseButtons button, Int32 clicks)
   at System.Windows.Forms.Control.WndProc(Message& m)
   at System.Windows.Forms.ScrollableControl.WndProc(Message& m)
   at System.Windows.Forms.ToolStrip.WndProc(Message& m)
   at System.Windows.Forms.ToolStripDropDown.WndProc(Message& m)
   at System.Windows.Forms.Control.ControlNativeWindow.OnMessage(Message& m)
   at System.Windows.Forms.Control.ControlNativeWindow.WndProc(Message& m)
   at System.Windows.Forms.NativeWindow.Callback(IntPtr hWnd, Int32 msg, IntPtr wparam, IntPtr lparam)
```

Figure 6.5: *Example stack trace produced in the pilot test run. These were used to further remove bugs from the code, which were mostly caused by simple lack of variable checking.*

- The time display was changed to the spiral image. A track played at a specific time will be plotted at the corresponding location on the spiral between the representative hour markers. This gives us a larger area to display tracks where we expect the greatest number of tracks to be played, and this time range has been extended; addressing the problem of an unrealistic day centralisation.

6.3 Pilot

After making the relevant improvements to the display methods, as discussed above, the software was completed and made ready for a pilot test. The purpose of this testing was to iron out and major bugs and address any immediate interface problems that became apparent.

The software package was given to a colleague with good technical grounding and an understanding of the basic concepts behind the software. This was necessary so that during the testing he could quickly and easily report bugs, using the diary feature screen in the PDA software itself, and to a text file on where the software caused an exception and shutdown.

6.3.1 Software Bugs

The bugs that were uncovered during this process were mainly to do with a lack of bounds and variable checking. An example stack trace can be seen in figure 6.5.

The pilot study proved incredibly useful in ridding the software of simple bugs that had been overlooked due to 'sensible' debugging - in other words, the way in which we tested the software as it was being coded, was to follow logical paths through it. Following seemingly illogical paths (as users do) reveals many errors, an example that came out during the pilot study being that the user attempted to delete a playlist when he had not created one, causing a NullReferenceException (seen in figure 6.5).

6.4 Prototype Evaluation

The final evaluation of the prototypes was then carried out. This was done in a similar way to that of the pilot study, the users being given the pack of software and hardware. They were asked to carry out the evaluation in two stages, to match the application phases- phase 1 with the PDA probe software, and phase 2 with the desktop manipulation software.

6.4.1 User Selection

As with our paper-prototype evaluation, this stage involved a selection of participants, some of whom were familiar with portable music players, some were in-touch with recent developments in musical and mobile technologies (our 'technical users'), and users who are familiar with portables but not nescessarily technologically minded (our 'non-technical users'). We also had another category of participants; those who had not used the probe software. The participants who used the probe also used the desktop application, while the other users were only asked to evaluate the desktop application. The purpose of this extra set of users is to see if they could make sense of the data produced by the first sets of users. If they could, then we can be more certain that our interface is one that is intuitive, and its behaviour is not implied by the surrounding context. (For, the first users knew what the project was about, having been asked to record their mood and so on).

We aimed to get a spread of users as mentioned above, however, there were some constraining factors involved in the selection process, based on technical problems that have yet to be overcome in the code. A few key points helped us choose ideal candidates;

1. The users had to have their music stored in one location on their desktop
2. The music store had to match the native iTunes directory and naming structure, that is (for example):

   ```
   C:\path\to\library\<ARTIST>\<ALBUM>\<track no.><TITLE>.mp3
   ```

   Where the items in angle braces are specific to the track.

3. As implied by the `.mp3` extension in item 2, the files had to be MPEG Layer 3 (MP3) format.

4. The user had to be happy for us to install our softare on their computer

5. The user had to be happy with the ethical issues associated with this project; namely the (anonymous) storage of data about their listening habits. (See ethical sign sheet in the appendix C).

6.4.2 Phase 1

The first stage was to let the user use the data probe to capture data about their listening habits. There were a couple of pre-requisites before this could happen however;

- The user had to install the Desktop application on their computer and go through the setup screen to specify their 8 moods, and associated colours. This also allowed them set the path to their local music store for use in phase 2.

- The user had to select 4GB of music to put onto the portable. This limit is associated to the size of the Compact Flash card we used, and while it should not affect the data for the test period, it should be noted that if this system was deployed commercially or on a fullscale test (i.e. over a period of weeks or months) then the music storage should be equal to that of the user's 'active collection' (that is, all the music they listen to).

After those had been achieved, the user was simply asked to use the PDMP in their daily routine for two days. The probe logged data during this time and stored it on the portable. The user was asked to keep a diary log of their experience, including any error reporting or interactive annoyances that they encounter, and this was assisted by a prompt that appeared randomly during use, asking for them to make notes – a method known as Experience Sampling[23], discussed in 6.1.

6.4.3 Phase 2

Phase 2 involved the users from phase 1 and also our extra users who had not used the probe, and were unaware of the main purpose of the project. As we mentioned before, the reasons for this extra set of users are twofold; firstly, as they will be viewing the collected data and its manipulation and use from an external point of view, their thoughts on how *they* might use the system might be different to how we see its purpose, and may offer some intriguing improvements or features to add. The other reason is to aid the first set of users in expressing their thoughts about the interface, and what they are doing as they do it. The methods described in section 6.1 concerning unbiased testing come into effect here, and essentially the second set of users act as sounding boards for the first users' feelings and thoughts on the system. [19]

To evaluate how well the system performed in its intended fashion we asked users to carry out various tasks, and recorded how well they managed to achieve these tasks. Where there were different ways in which the task could be achieved, we noted which method they used. The task list can be seen in the appendices. Following is a summary of all 5 user's actions and thoughts regarding this phase.

1. In order to import the log files, users tended to go for the Menu bar first, ignoring the Import button. Users attempted to multiple select items, something that had been overlooked and proved an annoyance to the users. A solution to this is simple, though a better one would be to remove this step altogether, and automate the transfer and import.

2. Users successfully managed to locate similar tracks by sorting the columns using the column headers as intended. Some users commented that the mood colours were the most obvious groupings, and that perhaps the times and locations were too 'numerical'.

3. We asked users to create a context based on some of the tracks they had listened to. Again, users tended to use the menu-bar over exploring the buttons and reading tooltips. Where users *did* discover and use the context-panel expansion button (and thereafter the buttons on the panel) some commented that the chosen icon (a plus sign) implied somethings else. As it had been used elsewhere for an 'add' function, this lead to some confusion. Two other methods for adding tracks to the context came up through the users' actions - the first being a drag and drop type interface, and the second – multiple selection of tracks.
   A bug also arose from this task- the context panel did not expand when a new context was created using the menu-bar, leaving the users a little confused as to what had just happened.

4. The users were then asked to use the recommendation system to add in other tracks to the context, but only if the selection was relevant to that context. Users went straight for the menu-bar again, but did have to search for the item, suggesting that it was not apparent enough on its own. Also, when the users closed the recommendation panel, they could not find a way to re-open it (as there was no way to) which caused some confusion – *"Oh. Where did it go?"*. The other issue that came out here was that not all the tracks by the recommended artist were relevant to the context. To conquer this we can simply raise a dialog with a list of tracks to add, before the add occurs.

5. This question was one of the most important of the evaluation – what did the users make of the three data types; how useful are they in creating a context playlist, and how accurate and obvious was the manner in which they were displayed. The first problem here was that users did not realise they *could* visualise the data. Although two of the users went straight for the button with an eye on it, the others seemed unsure and looked to the menu-bar, to which we had neglected to add the appropriate item. Also, using the same button to visualise all three data-types caused confusion.

Despite the problems in finding the visualisations, the users all managed it eventually. The users were asked to rate (out of 10) the following: 'Usefulness in creating a context playlist', 'Accuracy of the data shown', 'How obvious is the visualisation – how clear is it what representation means?'. These ratings are omitted due to statistical invalidity – 5 users are not a large enough sample to garner a respectable values.

Mood performed best in usefulness and obviousness, and was rated highly in accuracy (although as we will see in section 7, this could be improved). Time data was generally rated highly, but 'Location' was thought to be less useful and accurate. However the data presented on the map was obvious in what it meant.

6. Users had no problem saving the context, again choosing the menu-bar method over the buttons.

7. All the users double-clicked the item they wanted to play, as opposed to selecting and pressing the play button.

8. Having been asked to explore the application and comment on anything they discover, 3 of the users said that they found the immediate links to Last.FM's artist pages interesting. 1 user commented that the button layout for the context save/load/open was not the same as the save/load/open button layout for the playlists on the PDA software. 1 user said a search would be useful.

9. We asked the users to rate the applications overall Usefulness, Relevancy (to their music listening experience – i.e. does it 'fit' with how they listen?) and Interestingness. Table 6.1 shows the average ratings.

	Average
Usefulness?	7.8
Relevancy?	7.4
Interestingness?	8.6

Table 6.1: *Table showing the average rating of the whole system, in terms of how useful it is, how it fits in with the users' music listening (relevance) and how interesting the extra data is.*

We then asked the users to rate the entire experience, commenting on how they might use the system if they had it permanently, the benefits and drawbacks of the system as it stands, and whether they think the system achieves its purpose (i.e. does it enhance the experience and help in organising and selecting their music?). A discussion of the main areas follows.

- **Overall uses for the system** – All the users said that the system would prove very useful for creation of new playlists, allowing them to categorise their music and playback given contexts without having to search through their music, or think too hard about what music they want to hear.

- **Drawbacks** – Users commented that having to have all the equipment in order to get data was irritating, and on several occasions users had left the GPS unit near a window when inside a building (in order to acquire a reading) and later walked off without it. One user had concerns about privacy- commenting that they weren't sure the mood selection was discrete enough to use in public. Two users made similar comments regarding the effects of the computer doing selection work instead of the user–

  > *"It's good that [the system] can find other music in my collection that relates to my playlist, but I think I might end up listening to the same things and not listening to the rest of my collection."*

  Another comment was that they would prefer if it was available as a sort of 'plug-in' for iTunes, as they had only just got to grips with iTunes and would find switching difficult.

- **Purpose** – All the users agreed that it did indeed help in creating contextual playlists, and that they found accessing related tracks much easier. However, on the whole users did not think the system really helped with regards to music in different locations; this may have been due to several factors as discussed in section 7.

Chapter 7

Analysis

In this section we look at the data acquired in our evaluation, the reasons why some features did not perform as well as others, and ways in which to improve the current system.

7.1 Added Meta-Data

Firstly, a breakdown of how our added fields contributed or otherwise to the system.

7.1.1 Location – GPS

While a lot of work went into data conversion and manipulation thereof, it was apparent from the evaluation that perhaps the work had been misdirected. From the user ratings (out of 10) the locational data was deemed far less useful than the other data. Users commented that for the most part, they listened to all their music in the same place during the study, and that even if they had have moved around, it would not really make a difference to their listening habits.

The fact that most of the users in our evaluation remained in one place for the duration of the study may be attributed to the selection of users – those asked being 3rd year undergraduates – who all had deadlines approaching and therefore mainly stayed in their rooms and worked whilst listening to the music. This was a failing in the selection process, but another problem was that the evaluation period was too short; 2 days is not adequate as a data gathering period – in fact, some sources recommend a minimum of one week[24]. Unfortunately it was a limitation of time that caused this period to be shortened.

The manner in which the GPS data was displayed was perhaps not optimal for the average user. Although it could be sorted to group like locations, the data was still in an (essentially) raw format, with simple GPS co-ordinates. A better solution perhaps would be to carry out some co-ordinate evaluation, allowing users to set up a list of known co-ordinates and name them. A small prototype application was mocked up in to achieve this, making use of

Figure 7.1: *Here we see the bounds of an area being set up in (a) and after the fourth bounding point is added, a bounding box is drawn (b).*

Google maps API. Figure 7.1(a) shows the user having clicked 3 points to bound an area (in this case, the University of Wales, Swansea). In figure 7.1(b) a fourth point has been clicked and a simple bounding box has been drawn. This would then be incorporated into the configuration of the main application to have defined locations by searching for GPS co-ordinates that lie within a bounding box.

The above modification would allow us to sort the data by 'Named Location' as opposed to just GPS co-ordinates, giving some meaningful reference to the user, and allowing them to quickly see where they listened to a track, without having to find the plot-mark on the overview display. This may also increase the users' perception of 'accuracy' for this data field – this is largely due to GPS scatter, points being up to 10 metres out and also the recorded co-ordinates are incorrect if a good satellite lock has not been obtained.

While users did not find this field particularly useful or accurate, they did find it interesting, rating it highly. This seemed to be largely due to a novelty factor of seeing a location close-up on Google maps (which some of the users had not seen before). However, if the data is portrayed in a more readable manner (as suggested above) then users may find it more useful – the 'interestingness' factor causing users to play a little more with the feature and perhaps then finding a use for it. This is speculative, however, and would need to be investigated further.

7.1.2 Time

The time data was deemed to be more useful than that of location, achieving an overall high rating in terms of 'Usefulness'. The users also thought that the logged data was quite accurate, although some users did comment that there appeared to be a series of tracks when they had not even been awake, but admitted that they may have left it running over night.

From the study, it seems that time data is a sensible field to include when considering contextual playlist creation. When asked to create a context playlist in phase 2 of the evaluation, two of the five users created one which was related to time (e.g. 'Late night coding', 'Evening chillout'). This supports the suggestion that a key element of context is the 'when'[1].

While all users said that the display technique was obvious in what it was portraying, it became apparent that over a period of use the display panel would become cluttered. A workaround for this could be achieved by grouping times that are similar into a category, instead of using actual times. For example, we could put all tracks that were played between 5pm and 8pm into an 'evening' category. This would mean that our display would be less accurate but perhaps more useful – the time data in the track grid was visually displeasing, and a simpler representation may prove more effective. This would, however, lose the system's ability to allow the user to effectively 'play back' an entire day (in a diary fashion, similar to [11] which uses mood and bodily expressiveness as opposed to music) using the time data (although our system does not accommodate this inherently, it could be easily added in and users could achieve this as it is with a small amount of effort).

7.1.3 Mood

Mood was considered more useful than any other data type in creating contextual playlists. The majority of users rated this field higher than the others in terms of usefulness.

It seems from this data that mood is far more relevant to people's listening habits. While we listen to music at different times of the day and in different places, it seems that our mood is affected by these other two factors. If we imagine a scenario where we are listening to very relaxed music at home late at night, the fact that we are at home and it is late are important, however, these factors are tied in with being tired. Given that we might be tired in a situation where we are not at home and it is not late, would we still listen to relaxing music? Quite probably. Similarly, if it were late at night and we were at home having a party, we would not likely be listening to relaxed music, our mood would be one of excitement and this would be the deciding factor in our listening habits. So, it is fair to say that while the other two data types are important, mood is by far the overriding factor in what we choose to listen to and this is supported by it being perceived to be more useful.

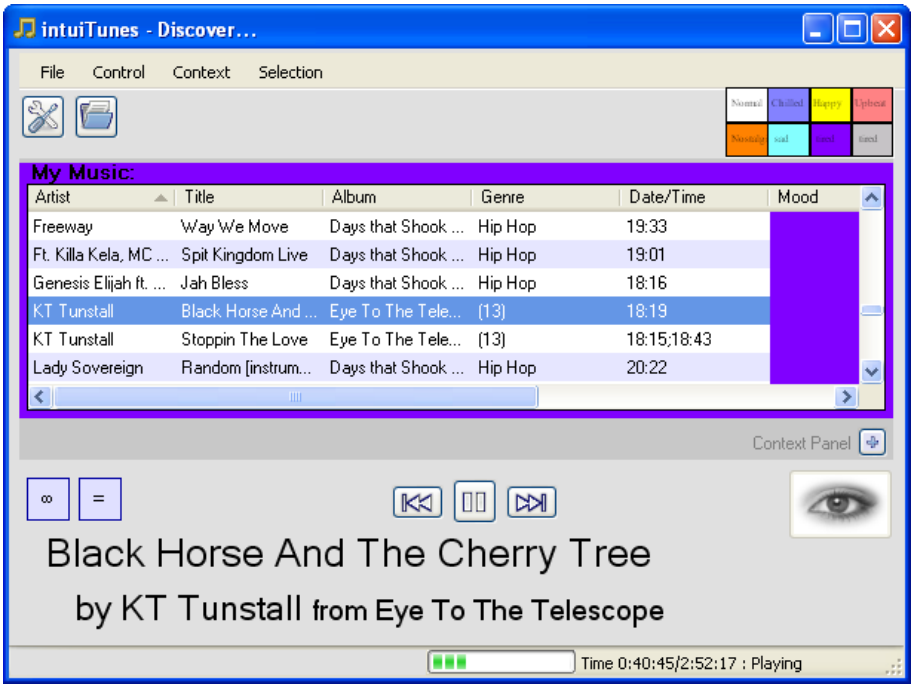


Figure 7.2: *A mock up of a mood key in the top right of the screen, showing the user what their colours represent.*

Mood scored lower than expected in terms of accuracy. Users commented that they often set their mood and then forgot about it, their mood might change but they would not update the selected mood. One user said she would prefer to be questioned every so often to ensure the mood being recorded was correct. This solution would work, however, it means that the software would be interrupting the user and this is something we tried to avoid – the data gathering process is supposed to be as transparent as possible. Another reason why it scored low in accuracy was because the users were not sure if it was accurate simply because they had no reference – most had forgotten what the colours meant to them, and wanted the describing text to refer to. A simple way to overcome this is to have a 'key' at the top of the screen, simply showing each colour with its mood label.

One other point that came up out of the evaluation was that attaching a mood to a track was not necessarily enough to define it. It would perhaps be better to keep a percentage rating for each track, and update it every time a mood is associated with it. Currently, the naïve implementation simply attaches a new mood to the track if it is not already attached, however, a better implementation would be to have a total mood count (this would be the same as the play count) and give the track a percentage rating for each mood, exemplified by the following pseudo code:

```
for each mood in moods:
      track mood percentage = track moodcount / track playcount *100
```

By doing this we could then sort not only into mood categories, but also in order of relevance (i.e., higher mood percentage means a track is more relevant to that mood). In this manner, we could see how much a track is related to a given mood, allowing users to produce more accurate playlists.

7.2 Interface Design

Obviously an important part of this project is design. We want our users to be content with using the software and want to avoid frustrating them with poor design choices. Through the evaluation, it was clear that some of the decisions made were not the best. In this section we will discuss the main failings and how to overcome them.

Text vs. Images Throughout the design of both the PDA application and the desktop software, icons were used on buttons to represent their actions. This is a useful design tool when working on small screens as it allows us to replace what are potentially long words with a small iconic representation – saving space. However, we must be careful in choosing the icons as a poor choice can lead to confusion.

From the evaluation, it was clear that users tended to use the menu with descriptive text as opposed to exploring the iconic buttons. However, the 'visualisation' button had no menu equivalent and this did cause some confusion. It is good practice to provide all functionality in the context menus, and give the most common actions keyboard shortcuts for the more advanced users. This was generally the case with the interface, however, this was occasionally overlooked as with the visualisation button. Jef Raskin points out in his book 'The Humane Interface' that iconic representation is regarded as being worthwhile, but is often made redundant as text descriptors are needed to tell the user what the icons mean[29].

While the icons provide a splash of colour to the interface, if they are not used or not obvious, then they do not add to the interface. If we remove them, we would have more room for the other, more important, controls such as the track list. However, if the users become familiar with their meanings, and if they are placed close to where they are needed then they reduce the amount of mouse movement required to perform a task. This is of course bettered by keyboard shortcuts. The problem is, by only having one method of carrying out a task, we assume all our users work in the same manner; they of course do not – every one is different and what works for one may not for another. So, by selectively providing recognisable icons and making all actions accessible from the context menus, providing keyboard shortcuts where actions are common, we avoid limiting our users.

Proximity and Placing One other point to notice about using buttons and icons in place of (or even along with) context menus, is that placing is important. In both the PDA and

Figure 7.3: *On the left are the buttons as displayed on the desktop application, and those of the PDA software on the right. The inconsistency caused one user confusion.*

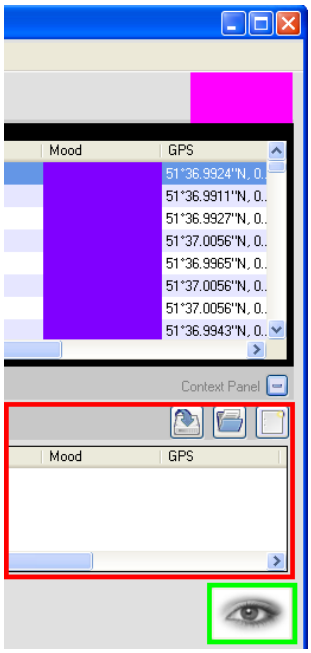


Figure 7.4: *The button (highlighted in lime green) which acts on the top track grid was closer to the context panel (highlighted in red) when the panel was expanded. This caused some users confusion – the button would be best placed at the top, highlighted in magenta*

desktop applications, buttons were used for open, save and new (with regards to playlists and contexts). However, the buttons were in a different order on the PDA to the desktop software (figure 7.3), an unnecessary inconsistency which could easily irritate users (and in the evaluation a user picked up on this).

Proximity was an issue that we had not fully considered also. The visualisation button was located next to the track grid when the context panel was hidden, but when it was expanded the button appeared closer to the context playlist. This implied to some of our users that the button acted on the context as opposed to the main track grid (figure 7.4). This is a simple case of moving the button, however without the evaluation it would have been overlooked.

Chapter 8

Further Work

After evaluating what the project did and did not achieve, we now look at ways in which it could be extended.

8.1 Meta Data

The most obvious way to make the contexts more accurate would be to increase either the amount of meta data (i.e. add more fields) or increase the accuracy of the current meta data. For example, we could use a better co-ordinate system than GPS to gather more reliable data in the location field, this would reduce the rather large scatter that is produced when staying in one place for any extended period of time, as seen in figure 8.1. A better system, which is currently under development and slated for release in 2010[4] is the European Union's GALILEO. This system is intended to provide near pinpoint accuracy, hence removing this 10 metre scatter.

To extend our meta data fields, we can look to two areas; the first being the immediate factors directly attributed to the user (i.e. their 'personal environment') and those which are out of the control of the user (i.e. the environment in which our user resides). Some examples of the sort of fields we might consider including are discussed below.

- **Tempo and Pace** – This is an example from the user's 'personal environment'. We might include a digital pedometer which feeds information to the PDA about walking pace. Using algorithms to detect tempo for each song, we can then match music to different paces. This could further be used to encourage the user to speed up their walking pace (perhaps they are listening to a playlist designed for exercising) or to slow down (a relaxing walk).
- **Pulse-rate and Activity level** – A slightly more invasive meta data gathering technique might involve a small wrist band which reports the user's pulse-rate to the device. This could give the software insight into the user's activity level (or possibly

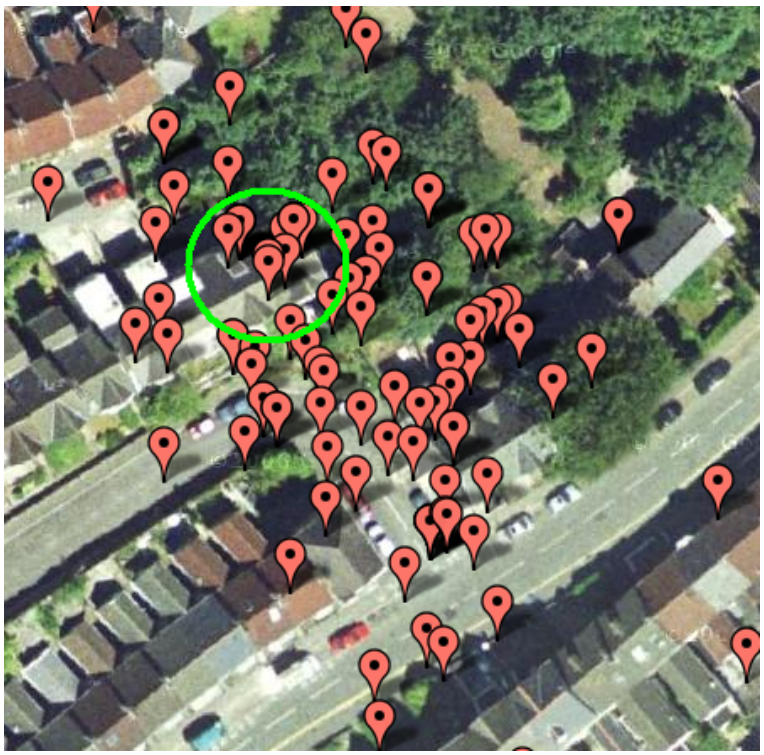


Figure 8.1: *GPS data mapped to Google Maps – the receiver had not received an adequate lock, coupled with the 10 metre inaccuracy causing the points to be scattered far from the actual location (circled)*

stress level!). Obviously this method may detract somewhat from the user experience as it involves potentially uncomfortable equipment to be worn.

- **Weather and Feeling** – An example from the external environment data; web-services exist that can look up a weekly weather forecast. This data could be used in a similar way to the user's mood, logging what the user listens to when its sunny, raining or snowing for example. An example– a user may listen to more music in the 'Ska/Reggae' genre in summer.

8.2 Automation

One of the initial aims of this project was to reduce user interaction to a minimum. Due to the technical hurdles encountered during functional prototyping this was not possible, instead we adopted an approach whereby the user was able to create playlists themselves with the aid of the extra data, instead of the software creating these playlists itself. One way in which this project could be extended is to carry out this automation of playlist creation.

A more advanced solution might consider using automated mood reading techniques. This would complete the system – removing as much human interaction as possible, allowing the user to enter a natural state and simply enjoy the music. This could be done in a number of ways, for example, Kristina Höök et al. considered the bodily movements of a person, using common gestures to read their moods[33]. Walt Disney have a patent application which covers (clauses 27, 28 and 29) reading a persons mood by body temperature, pulse rate and perspiration levels, which could be considered in a fully automated system (as mentioned in 8.1)[5].

Bootstrapping One of the problems with this automation process (depending on how it is implemented) is – how does the program know what to base its choices on, if it has no data initially – also known as a bootstrapping problem (this name is taken from the hardware startup sequence where the hardware needs to know where to look for its data, but it needs data to tell it this). There are several ways to get round this, the simplest being remove the problem by providing the software with data; essentially giving it a training package. This method means the user would have to train the software before using it – a very frustrating process if they just want to listen to some music.

A much more elegant way to achieve the same effect is to use a random element along with collaborative filtering and perhaps some machine learning. Consider this scenario...

> *Richard picks up his intuiTunes unit for the first time, selects a track and presses play. The unit instantly plays the track and displays a panel with a mood indicator asking Richard to select his mood. He taps 'Happy' and the panel fades, leaving the screen filled with the track status – artist, title, genre and coloured with his*

'Happy' colour. The screen also has two buttons, a tick and a cross. The track finishes and the unit selects another track randomly. This track does not fall into what Richard would call 'Happy' music. He taps the cross and the track stops. The unit quickly cross references the genre and artist of the two tracks it now has information about, and selects a track which is similar to the initial track (perhaps the same artist, or genre, or from the same album). This process continues, and by process of reinforcement and negative feedback the unit forms a database from which it can construct contexts. New contexts where perhaps Richard is in the same mood but a different location use this previous data to try and perform intelligent selection, also selecting random tracks to expand the selection.

This form of feedback allows us to profile the music without interrupting the user too much – they simply say if a track is inappropriate or if it is a perfect match. Note that for a track to be acceptable, they need not reinforce that fact. This can be furthered by detecting skips and playcounts. For example, if a track is constantly replayed, this can be considered similar to 'ticking' the track (reinforcing the selection). If a track is skipped over in a given context, then perhaps the selection choice is borderline. Fuzzy sets could be employed to take advantage of this sort of data, where there is a sort of 'wavering' of what is deemed acceptable and what is not (for, it is not always the case that exactly the same context invokes the same choices).

8.3 Re-mobilising

Part of the automation process mentioned above removes the need for the desktop software. If all the processing and selection takes place then there is no need for the data to be loaded and manipulated on the desktop – fully mobilising the system.

However, as we intend to allow the users to visualise their habits, this would also need to be ported back to the PDA in order to fully mobilise the system. This would be fairly simple in the case of both mood and time data, however, in order to produce Google mapped images, we would require a permanent wireless Internet link. This wireless access would also be required in order to extend the recommendations by the collaborative filtering provided by the web services from Last.FM.

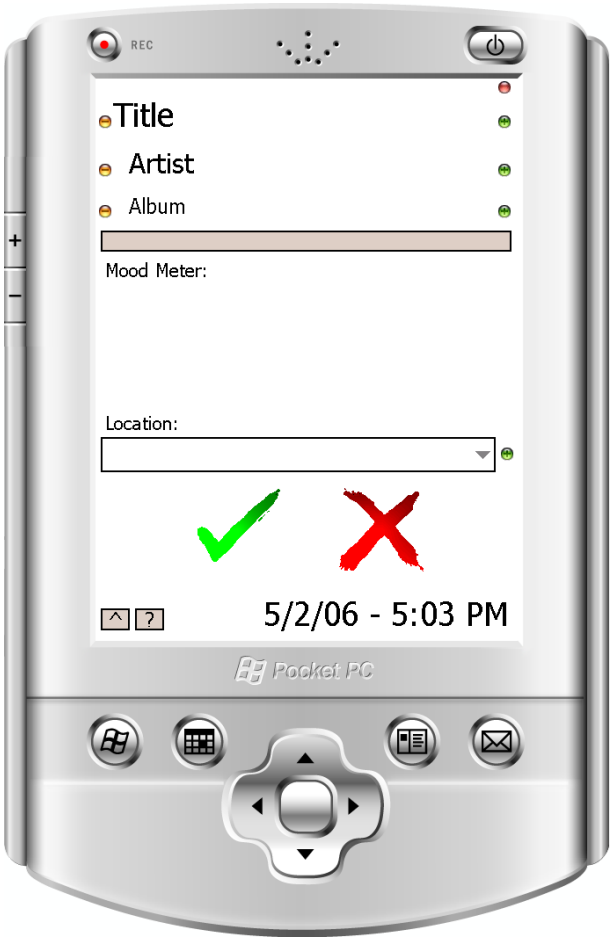


Figure 8.2: *A mock up of the interface which might be used in a fully automated prototype. Here we see the feedback buttons (a tick and a cross) enabling the user to quickly reinforce or counteract the selection.*

Chapter 9

Summary and Conclusion

This project aimed to enhance the experience of portable music player users by allowing them to access their collection in a different manner – that of context. In an effort to ease manual filtering through potentially large collections of music, we provided an extra layer of data to filter on, namely 'where', 'when' and 'what'. These three extra fields, defined by GPS co-ordinate data, clock time and user mood (respectively) were logged using a portable data probe based on a PDA, and could later be accessed and manipulated by the user in the desktop environment. The users were then able to construct contextual playlists based on sorting the collected data, and then save, load and play those contexts back later on. This approach was intended to save users from manually picking through their collections in order to find all the tracks that fall into the category of (for example) 'late night coding at home'.

It was found that while the mood field plays a strong role in defining a context, location information was deemed less important (and in fact, none of our users used this data to create a contextual playlist), though this may be due to a poor, less readable representation of the data. Time data was also considered less important than that of mood, but was still used by the users to create contextual playlists. Mood data was the most useful in creating contexts, and the majority of users liked having the added field available.

The extended data could be visualised to gain an overview of listening habits, intended as an aid for the users to understand their different musical contexts. The visualisation methods were found interesting by the users but the majority said they would not use them other than out of interest – they would not be useful in creating contextual playlists.

Further to this study, the software could be fully automated to reduce the user interaction to a minimum, selecting tracks without any intervention after a period of 'real-time' learning, allowing the user to listen to their music while the software makes intelligent guesses as to what they might want to listen to. Automating mood reading would be a most desirable extension to this project.

We have seen that the portable digital music player can be enhanced. Moving traditional, stationary, machines into the mobile environment has forced a shift in perspective – to create software that considers the context in which it currently resides, and by fully exploring how this can be done and implementing these methods, we can produce a more symbiotic relationship between people and technology.

Bibliography

[1] Gregory D. Abowd and Elizabeth D. Mynatt. Charting past, present, and future research in ubiquitous computing. *ACM Trans. Comput.-Hum. Interact.*, 7(1):29–58, 2000.

[2] ample.sf.net. Id3 header graphic, April 2006. `http://ample.sourceforge.net/images/id3v2-big.gif`.

[3] Pedro Cano, Markus Koppenberger, and Nicolas Wack. An industrial-strength content-based music recommendation system. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 673–673, New York, NY, USA, 2005. ACM Press.

[4] European Commission. Galileo project home-page, May 2006. `http://europa.eu.int/comm/dgs/energy_transport/galileo/index_en.htm`.

[5] The Walt Disney Company. Us patent application number 20060005226, January 2006. `http://tinyurl.com/cv3aj`.

[6] Apple Computer. Apple special event, October 2005. `http://www.apple.com/quicktime/qtv/specialeventoct05/`.

[7] Apple Computer. itunes, October 2005. `http://www.apple.com/itunes/overview/`.

[8] M. Csikszentmihalyi. Flow: The psychology of optimal experience, 1991.

[9] Sally Jo Cunningham, Matt Jones, and Steve Jones. Organizing digital music for use: an examination of personal music collections. In *Proceedings of ISMIR 2004 - 5th International Conference on Music Information Retrieval*, October 2004.

[10] T. Erikson and H. Simon. Protocol analysis: Verbal reports as data. 1985. As cited in [22].

[11] Madelene Lindström et al. Affective diary designing for bodily expressiveness and self-reflection. *Proceedings of the ACM CHI 2006 Conference, Extended Abstracts. (To Appear)*, April 2006.

[12] Yazhong Feng, Yueting Zhuang, and Yunhe Pan. Popular music retrieval by detecting mood. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 375–376, New York, NY, USA, 2003. ACM Press.

[13] Adam Field, Pieter Hartel, and Wim Mooij. Personal dj, an architecture for personalised content delivery. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 1–7, New York, NY, USA, 2001. ACM Press.

[14] J. T. Foote. Treeq package, April 2006. `ftp://svr-ftp.eng.cam.ac.uk/pub/comp.speech/tools/treeq1.3.tar.gz`.

[15] Google.com. Google maps api, April 2006. `http://www.google.com/apis/maps/documentation/`.

[16] Gracenote.com. Cddb, October 2005. `http://www.gracenote.com`.

[17] Keiichiro Hoashi, Kazunori Matsumoto, and Naomi Inoue. Personalization of user profiles for content-based music retrieval based on relevance feedback. In *MULTIMEDIA '03: Proceedings of the eleventh ACM international conference on Multimedia*, pages 110–119, New York, NY, USA, 2003. ACM Press.

[18] Dan Hong, Dickson K. W. Chiu, and Vincent Y. Shen. Requirements elicitation for the design of context-aware applications in a ubiquitous environment. In *ICEC '05: Proceedings of the 7th international conference on Electronic commerce*, pages 590–596, New York, NY, USA, 2005. ACM Press.

[19] Sami Hulkko, Tuuli Mattelmäki, Katja Virtanen, and Turkka Keinonen. Mobile probes. In *NordiCHI '04: Proceedings of the third Nordic conference on Human-computer interaction*, pages 43–51, New York, NY, USA, 2004. ACM Press.

[20] ID3.org. Id3 tagging, October 2005. `http://www.id3.org/`.

[21] ID3.org. Id3 specification, April 2006. `http://www.id3.org/id3v2.4.0-structure.txt`.

[22] Matt Jones and Gary Marsden. *Mobile Interaction Design*, chapter 7, pages 197–207. John Wiley & Sons Ltd., 2005.

[23] Matt Jones and Gary Marsden. *Mobile Interaction Design*, chapter 7, page 143. John Wiley & Sons Ltd., 2005. See also, [8].

[24] Matt Jones and Gary Marsden. *Mobile Interaction Design*, chapter 5. John Wiley & Sons Ltd., 2005.

[25] MusicBrainz. Musicbrainz home, October 2005. `http://musicbrainz.org/`.

[26] J. Nielsen and R. Mack. Usability inspection methods. 1994. As cited in [22].

[27] Rosalind W. Picard. *Affective Computing*, chapter 3, pages 98–101. The MIT Press, 1997.

[28] PocketCalculatorShow.com. The birth of the boombox, April 2006. http://pocketcalculatorshow.com/boombox/birth1.html.

[29] Jef Raskin. *The Humane Interface*, pages 128–174. Addison Wesley and ACM Press, 2000.

[30] Kerry Rodden, Wojciech Basalaj, David Sinclair, and Kenneth Wood. Does organisation by similarity assist image browsing? In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 190–197, New York, NY, USA, 2001. ACM Press.

[31] Risto Sarvas, Erick Herrarte, Anita Wilhelm, and Marc Davis. Metadata creation system for mobile images. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 36–48, New York, NY, USA, 2004. ACM Press.

[32] Relatable Software. Relatable trm, October 2005. http://www.relatable.com/tech/trm.html.

[33] Petra Sundström, Anna Ståhl, and Kristina Höök. emoto: affectively involving both body and mind. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 2005–2008, New York, NY, USA, 2005. ACM Press.

[34] TeleType.com. Unofficial nmea string documentation, April 2006. http://www.teletype.com/pages/support/Documentation/RMC_log_info.htm.

[35] Nigel Warren, Matt Jones, Steve Jones, and David Bainbridge. Navigation via continuously adapted music. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1849–1852, New York, NY, USA, 2005. ACM Press.

[36] wc3.org. Meta data profiles, April 2006. http://www.w3.org/TR/REC-html40/struct/global.html#h-7.4.4.3.

[37] Wikipedia.org. Meta-data, October 2005. http://en.wikipedia.org/wiki/Meta_data.

[38] Wikipedia.org. Musicbrainz, October 2005. http://en.wikipedia.org/wiki/MusicBrainz.

[39] Wikipedia.org. Wikipedia nmea entry, April 2006. http://en.wikipedia.org/wiki/NMEA.

Appendix A

A copy of the questions asked during the paper prototyping stage of the evaluation.

Section 1 – User's Thoughts

Q1: Given the following categories of meta-data, describe a means by which you could visualise this data in a simple, graphical manner.

a) Relating to Time
b) Relating to Mood
c) Relating to Location (given precise Global Positioning System co-ordinates)

The goal here is to establish unbiased ideas as to what the user might see as a 'good' way of displaying data.

Q2: Using the provided images (see Images Section, below), mock up how you might display:

a) 3 tracks:
   i) one played at 6AM
   ii) one played at 6PM
   iii) one at 12 Noon
b) 5 tracks:
   i) one played when you feel sad
   ii) one played when you feel like dancing (or similar)
   iii) one played when you feel angry
   iv) one when your mood is not really distinguishable
   v) one played when you feel content
c) 3 tracks:
   i) one played at University
   ii) one played on the beach
   iii) one played at home

The goal here is to establish whether, given a selection of displays, the user can easily distinguish the differences between different displays and make sense of them.

Q3: (Showing the users the intended models) Criticise/Comment on these models, and explain what you believe they represent

a) Show Time display
b) Show Mood panel
c) Show Google maps (interactive version?)

The goal here is to promote open discussion about each display, which will hopefully draw out their immediate weaknesses that we may have overlooked.

Appendix B

A copy of the questions asked during the full prototype stage of the evaluation.

1. Load up the logs created by the probe
2. Find tracks that are related to each other:
    1. by moods.
    2. by the times they were played at.
    3. by the locations they were played at
3. Create a new context of your choosing, using the tracks that have been loaded.
4. Use the recommendation system to expand the context, but do so only if the recommendations are relevant.
5. Try the overviews of the dynamic data types and comment:
    1. are they useful
    2. are they accurate
    3. are they obvious
6. Save the context
7. Create a new context (as above) and start it playing.
8. *Explore* the rest of the application, what do you find?
9. If you do find anything extra, rate it's:
    1. usefulness
    2. relevancy
    3. interestingness

from 1-10 where 10 is 'Most' and 1 is 'Least'

From here we will ask the users their opinions of the system, covering these main areas:

1. Did they feel the data manipulation features were useful/relevant/interesting?
2. How might they find uses this system (ignoring current technological constraints) if they were given one to use for a year?
3. How would they benefit or otherwise from the features?
4. What do they think the purposes of the features are?
5. After being told the purpose, does it achieve it's purpose?
6. How do they rate the experience (in words)?

Appendix C

A copy of the consent form that the users signed before taking part in the evaluation.

About...

This study is a look into enhancing the experience of using Portable Digital Music Players in everyday life, as a part of a 3rd year project in Computer Science Human Computer Interaction. It is carried out by Stephen Pike under the supervision of Dr. Matt Jones who can be reached matt.jones@swansea.ac.uk.

Thankyou for taking part in our user evaluation. Before we continue, please read this document carefully and if you agree with all the details in this document, sign and date at the bottom.

Anonymity

All data collected during the period of testing will be anonymised. Logfiles regarding listening habits and any other feedback data collected during this period may be used in analysis, and as such may appear in whole or in part, in the final documentation. Raw data will be destroyed after a period when it is no longer in use by ourselves (University of Wales, Swansea).

Requirements

By agreeing to this document, we will require that a piece of software (intuiTunes-Discover) will be installed on your Personal Computer. This piece of software requires that you have a collection of music on your PC, and that this collection is stored in the following format:
C:\Path\To\Your\Collection\<artist>\<album>\<title>.mp3
Where, the former part of the path is the path to your collection and the latter (those marked in angle brackets) vary depending on the track information.
If you are unsure about this please ask us.

Our Responsibility

As you will be testing this software in a 'real-world' environment (i.e., not in a laboratory) you will not be supervised by us. If, at any time during testing the hardware or software fails in any manner, please contact us and we will attempt to fix the problem immediately. See contact details below. Although the software has been tested and should not interfere with any data on your computer, we hold no responsibility for any damage, destruction or loss to data as a result of using this software.

I hereby agree to the terms above

Signed: .....

Print name: .....

Date: .....

Any issues, please contact 293828@swan.ac.uk or telephone 07976 475872