

Final Year Project:

MULTIMEDIA AND DIGITAL SYSTEMS

Cross-platform Media Centre:

Final Report

- Simon Metcalfe • Informatics • Tutorial Group: 9 •
- Project Supervisor: Dr. Natalia Beloff •

Count	
Summary	37
Main Body	11862
Pages	65

Summary

The report covers the design and implementation of a cross-platform media centre application for a third-year final year project. The application is designed to view multimedia content from a computer, using a television and remote control.

Statement of Originality

This report is submitted as part requirement for the degree of Multimedia & Digital Systems at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged.

Signed: _____ Date: _____

Contents

Page	
	Report Body
1	Introduction
3	1.0 Background Studies
	1.1 Reasons for project choice
	1.2 Project relevance to degree
4	1.3 State of the market
	1.4 Existing products
8	1.4.1 Summary
9	1.5 Existing remotes
	1.5.1 Summary
10	1.6 Specifications and requirements
11	1.7 Professional considerations and requirements
12	2.0 Design
	2.1 Choice of design paradigm
	2.2 The Model View Controller paradigm
13	2.3 MVC: View
	2.3.1 Graphical user interface
	2.3.2 Technical considerations
	2.3.3 Target environment considerations
14	2.3.4 A user-centered design approach
15	2.3.5 Generic display components
16	2.3.6 Main menu
17	2.3.7 Media library
18	2.3.8 File browser
	2.3.9 Audio player (now playing)
19	2.3.10 Video player
	2.3.11 Configuration settings
20	2.3.12 Remote control
22	2.4 MVC: Model
	2.4.1 MP3 ID3 tags
23	2.4.2 Existing databases
24	2.4.3 Chosen database design
25	2.4.4 File system
26	2.4.5 M3U playlists
27	2.5 MVC: Controller
	2.5.1 General navigation
	2.5.2 Typical tasks
28	2.5.3 Use cases
29	2.5.4 Library browsing
30	2.5.5 Sequence diagram: playing music/viewing pictures

31	3.0 Implementation
	3.1 Choice of technology
	3.2 Resources required
32	3.3 Overview of code structure
33	3.4 MVC: View
34	3.4.1 Customisable appearance
	3.4.2 Remote interface
35	3.4.3 Implementing keyboard control
36	3.4.4 View screenshots
39	3.5 MVC: Model
	3.5.1 Reading audio tag data
40	3.5.2 Reading the file system
	3.5.3 Network drives
41	3.5.4 Database Xtra
	3.5.5 Implementing the database
44	3.5.6 Configuration
	3.5.7 M3U reader/writer
	3.5.8 Additional info methods
45	3.6 MVC: Controller
	3.6.1 General navigation
46	3.6.2 Media playback
47	3.6.3 OSD implementation and overlay issues
48	3.6.4 Media control
	3.6.5 DVD playback
49	3.6.6 Predictive text system
50	3.6.7 Dictionary system
51	3.7 Implementation overview
52	4.0 Testing
	4.1 White box
	4.1.1 ID3 reader
	4.1.2 File/folder reading
	4.1.3 AlbumArt reading
	4.1.4 Media playback
53	4.1.5 Image viewing
	4.1.6 Media database
	4.2 Black box
	4.2.1 Cross-platform testing
	4.2.2 User acceptance testing
55	5.0 Maintenance
56	6.0 Conclusion
	6.1 Future additions
57	Bibliography
	References
	Acknowledgements

Report Body

Introduction

With the increasing speed of internet connections, CPU speeds and hard disk capacity, computers are becoming a way in which many enjoy audio and video content. However, often this can only be achieved by using the computer monitor and speakers, sitting at a desk in, for example, the study.

It is most desirable to view this content from the comfort of an arm chair in a living room, such as you would with a traditional TV, VCR and HiFi system. As computers are now faster, smaller and quieter, they are finding their way into living rooms. However, for computers to fully integrate themselves as part of a home entertainment system, a user should not need to leave their comfy chair to select another video using the mouse and keyboard. To achieve this, the traditional monitor, keyboard and mouse must be replaced with a standard television set and infrared remote control.

Figure 01: *Microsoft's vision of everyday media centre use in the home.*



Since the typical media collection on a computer is growing larger every day, being able to navigate a vast selection of information is very important. Special attention must be paid towards access when a remote control, with a limited number of keys, replaces a computer keyboard. Already the traditional method typing characters can only be implemented through an on-screen grid or numeric pad. In addition the loss of the mouse leads navigation around items on screen can only be performed by the directional keys.

Also, most home-entertainment devices, such as digital receivers, DVD players and HiFi systems operate in an analogous way that is familiar to all, and thus very acceptable. For a computer to take the place of one or many of these items, it too must provide similar features in a recognisable fashion. An ideal media centre would allow anyone to pick up the remote and begin use within moments.

The intention of this project is to develop a Media Centre type application for viewing multimedia content on a computer, through a standard television set. The implementation will be designed particularly towards manipulating a high volume of media efficiently. The author intends to implement methods of searching/selecting media via remote control that are not found on existing products.

The report is divided into 5 main sections, **Background studies**, **Design**, **Implementation**, **Testing** and the **Conclusion**. Background studies introduce research performed prior to project design, such as reviewing the current market state of similar products, specifications and requirements the software may have. The design stage takes the specifications & requirements and produces a design based upon the Model View Controller paradigm. Usability requirements will be taken into account, as well as restraints that occur with the technology involved. Implementation describes the process of undertaking the project on the chosen platform. During this section, white-box testing will be performed on parts of the project as they are completed. Finally, the conclusion reflects upon the entire project, such as the overall success, further improvements and changes, and considers how the project would be approached differently if it were repeated.

1.0 Background Studies

1.1 *Reasons for project choice*

The changing way in which many people obtain, manipulate and view their media is of particular interest to the author. Never before have so many different products been available, allowing users to access their media in different ways and at different locations. For instance, portable media centre devices have reached a point where handheld devices can store days of digital video for instant access. However, the philosophy behind home media centres is the sharing of media around house, so multiple copies of CD's and DVD's are not required.

Although there has been a recent influx of media-centre based products by a variety of manufacturers, the technology is still in its infancy. While the principal is essentially the same, the way in which the systems are operated and the media is displayed varies considerably. This is especially noticeable when compared to established devices, such as mobile phones, DVD players, where most manufacturers adopt the most common, tried and tested interfaces. Many of the current media centres rely too heavily on the keyboard and mouse, or do not facilitate efficient navigation and control given low TV resolution of a and limited keys on a remote.

It is the author's intention to produce a media-centre type application that builds upon the features found in existing media centres, but with adding additional functionality to allow large collections of media to be managed efficiently.

1.2 *Project relevance to degree*

Implementing a project such as this relates to many aspects of the author's degree: **Multimedia & Digital Systems**. The basis of many modules involved object-orientated programming, electronic and visual communication, human-computer interaction and design, all of which are part of the majority of multimedia applications. The following modules were particularly relevant towards the project:

Introduction to Programming, Further Programming, Data Structures, Introduction to OS, and Multimedia Systems: The main skill studied and practiced in these modules was OO programming, which is vital for implementing a project such as this. Programming skills related to GUI development, databases, threads, exception handling and the JavaSound API were also learned.

Human-Computer Interaction: A user-centred design approach is important when planning the interface and navigation of any software, but especially that which is to be used by the general public. User-experience goals, design/usability principles and evaluation heuristics will be taken into account. This course introduced designing an interface from a user-centred design process.

Multimedia Systems 2: This course featured graphical design and animation programming in Macromedia Flash, which may be used to implement a GUI components.

Video production techniques: Although a project such as this is based around the playback of audio/video content produced by others, the course taught invaluable skills regarding television picture standards and producing graphics for display on TV.

Electronics Design Project and Computer Networks 1 & 2: A project of this kind will require remote control hardware to accompany the software. An interface is needed to interpret commands from an infrared remote control and pass to the software program accordingly. Alternatively there are commercial devices which may be purchased.

Databases: The project will require a database to store media information, and an appropriate database design is required. An entity-relationship diagram will be produced as part of the model component of the MVC paradigm.

1.3 State of the market

There are in fact many existing Media Centre applications currently on the market; the most predominant being Microsoft's Windows XP Media Centre Edition 2005. This is not as a software application, but in fact an operating system. However, Microsoft only ship this software with new 'Media Centre' PC's; purchasing the OS off the shelf as a full-version or Windows XP upgrade is not possible. Granted specific hardware is required to watch and record TV, but for users who wish to use an existing PC as a media centre, it is not a solution. The remote hardware must also be "Media Centre Compatible", although this standard is gaining popularity.

There are some media centre applications designed to run as an application within Windows. These include J.River's Media Centre 10, InterVideo's Home Theatre Platinum and 8 Dimensions TVedia. Not only do all these require XP, J'River's is essentially desktop media software with a Media Centre mode, and TVedia is dependant on an expensive PCI card and remote control. When considering these solutions, using a Mac or free operating system such as Linux is impossible.


There is very limited media centre software for use with the Mac and OS X. EyeTV is a Mac-compatible PVR solution, which also enables viewing/recording of TV. Not only is specific hardware required, but most operations are performed using the keyboard and mouse. The built-in TV tuner also adds unnecessary expense if it is only used for playing media already on the computer.

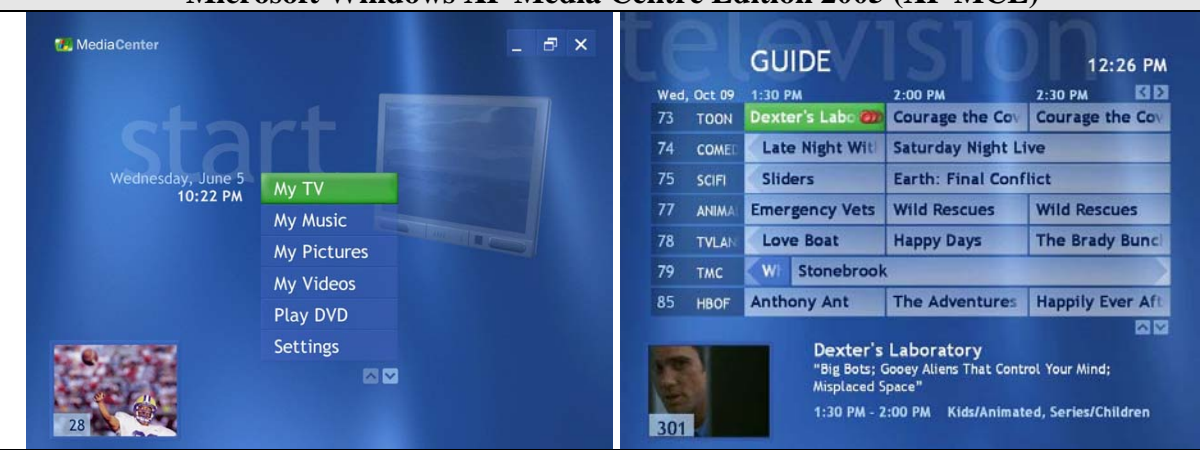
1.4 Existing products

Some of the most popular existing media centre products have been evaluated to find their features and draw-backs. The research will be taken into consideration when deciding on specifications for the project. Unfortunately only mouse and keyboard navigation can be tested on each application, so no evaluation on remote control operation can be performed.

Evaluation: Test Media	
Audio	17,000 tracks (1300 albums) with fully organised ID3 information.
Video	AVI (DIVX/XVID), MPG
Images	JPEG, BMP, GIF

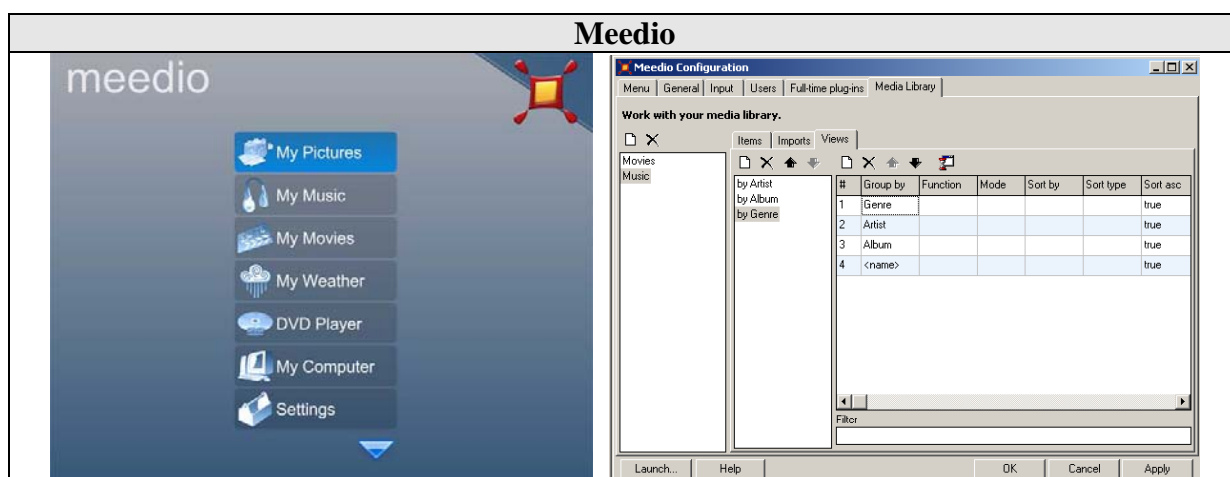
The evaluation table shows the test media made available to each application. Although many collections may not be such a vast amount of files, there are desktop applications capable of handling this amount without any trouble. It is often found that performance issues occur with the database when a large collection is used.

InterVideo Home Theatre Platinum	
	
Type: Software Application	Platform: PC, Windows XP
Support: TV tuner, DVD/VCD, Video, Images, Audio (with database)	
<p>InterVideo's home theatre platinum is installed as a software application on Windows XP. Installation was straightforward, but the application associated itself with all its supported media types, even after being instructed not to associate. Users who will be using the PC for desktop use will most likely have their preferred media software set to open media files. Incidentally, the software is slow to load, which is undesirable if the user wishes to play a single file only.</p> <p>A separate configuration program is used to select locations for video, audio, pictures and playlists. However, after adding media, accessing the Music menu took over 30 seconds. Tracks can be organised by songs, albums, artists and genres. However, if an artist is selected, all tracks by the artist are displayed in a single list, and cannot be separated by album.</p> <p>Single clicking any entry in the database 'ticks' it, and double clicks open the selection. Since double-clicking is not implemented on most remotes, differentiating between select and tick when using the remote was not clear.</p>	

Microsoft Windows XP Media Centre Edition 2005 (XP MCE)	
	
Type: Operating System	Platform: PC
Support: Multiple TV tuners, DVD/VCD, Video, Radio, Images, Audio (with database), Media centre applications/games.	
<p>Setting-up a test environment for XP MCE was difficult as it is only obtainable by purchasing a media centre, and requires an OS re-install. Installing/starting the system for the first time is no different to any other version of windows, with the exception of the “Media Centre” shortcut in the start menu. Once started, the centre takes you through a wizard that covers the licence/privacy, internet check/set-up, and display/speakers configuration. The default menu scrolls in a circular motion with base items appearing at the top once passed. However, not all menu items are displayed on the screen at once, and there is no indication that the menu contains more items off-screen. The menu makes discreet noises when navigated. This is useful for when using the remote as it gives the user feedback the command was received, but can be turned off. The centre is focused around media already contained within it; there is no link to CD/DVD or memory card devices. However, as the image above suggests, a ‘Play DVD’ menu item should appear on a computer equipped with a DVD drive and DVD disc.</p> <p>Media can be added by using the main interface (not an external program). The system also does not show you the overall progress of adding media, does allow you to continue using the software at the expense of DB creation speed.</p> <p>The music database offered more flexibility over the view/selection of music. When a genre is selected, the results can only be grouped by album and track, but not artist. If artist is selected, the software gives you radio buttons to select by songs or albums, but selecting by albums only shows ‘Please select view songs’. Selecting a track opens a further screen showing information and art (if available), but further input is required to play or queue the item. There was no way to queue the track or album without entering this additional screen. The current song playing is always shown in the corner of the screen, which also allows playlist access. The playlist has an edit mode which allows the order of songs to be changed, but only one song can be moved at a time. Searches can only be performed on the entire database (i.e. not by artist), but the searches do return albums and songs that match separately. A bug in Windows Media Player causing ID3 tags to be misread affects XP MCE too, as both share the same database.</p> <p>Video playback uses a folder view only, and folders/network shares containing video can be added. A major drawback with network shares is only the root share can be selected, folders inside cannot. Adding videos is performed by right-clicking and selecting an option with the</p>	

context menu. It is unclear how this is performed on the remote. Previews are made of each video, and they can be queued by selection or by folder.

Music can be played during a slide show, but must be started first as the slide show is lost once it has been left. The on-screen media button (next, previous etc.) control the slide show and there is no obvious way to control music. .



Type: Software Application

Platform: PC, Windows XP

Support: TV tuner, DVD/VCD, Video, Images, Audio (with database)

Medio also uses a separate application for configuration and adding media; hence there are very few options in the Settings menu. The main menu does not show all available options at once, but does indicate the menu can be scrolled down.

The system checks for new media at start-up, which may be slow and not preferable for all users. A file browser allows media to be selected that is not in the database, such as from discs or memory cards. However, when media is launched from the file browser it is not opened with Meedio, but the associated Windows application. Meedio will generate thumbnails, but only for folders that have been opened.

The configuration program (shown above) is very comprehensive and allows full customisation of the music database, but the default settings only allow browsing by Artist. A jukebox mode is included that picks albums from random, but large grey spaces are left in the list for albums that do not contain art. Meedio will read album art which is stored within ID3 tags. Meedio's playlist does have an edit more, but it is unclear how songs are shifted and it cannot be performed with the mouse. As with XP MCE, there is no obvious way to manipulate the playlist while viewing a slide show.

Meedio's search allows categories to be selected, but a grid of alpha characters appears which suggests text is entered by using the arrow keys, and not the numeric pad. Only tracks are shown in the results, even if an album name matches.

1.4.1 Summary

Overall, many of the existing solutions share similar problems when handling large quantities of media. Beside the database performance issues, most have inadequate library browsing, restricted searches or poor organisation of results. In addition to these problems, most solutions exemplified no easy way to navigate long lists or of items and input alpha characters from the remote control.

The following design considerations have been produced as a result of the evaluations:

- Network shares should be treated like local drives, allowing folders and subfolders to be selected.
- Windows file associations should not be affected.
- Progress bars should be displayed for lengthy processes.
- All items of the main menu should be displayed so a new user will know what functions are available.
- Menu access and media library should be made instantaneous, even if this causes an increase in start-up time to initialise the database.
- A check for new media should be instantiated from the media centre menu to allow the system to accommodate new media without having to exit and use a separate application. If a start-up check is included, it should be optional.
- Audio tracks in the media library should always have the option to be viewed by artist and album.
- An enqueue feature should be available for the current selection, whether it be a genre, artist, album or track. This will allow all tracks of a certain genre, or by a certain artist, to be played.
- The media library should facilitate searching of all categories, and text input should be performed using the numeric pad.
- Text input using the numeric pad should be extended to allow items in long lists to be reached quickly.
- Employ an advanced text input method such as predictive-style input used on most mobile phones.
- A method to control audio playback at all times, such as during a slide-show.

1.5 Existing remotes

For an application such as this, the remote control design is just as important as the software itself. Unlike a keyboard and mouse, most remote controls are application-specific and must be designed carefully to maintain efficiency with a reduced number of keys. The following table contains an evaluation of remote controls from existing media centres.

Figure 02: Existing media centre remotes.



XP Media Centre (OEM/HP/Elonex)

All XP Media Centre remotes are similar in specification. Overall, the design is well planned, but many of the keys have fixed functions. For example, the dedicated DVD keys are only useful when a DVD is played.

As with all remotes reviewed, these have volume control keys.

Snapstream Firefly

This remote is designed for TVedia's Beyond TV software, but is popular with other applications such as Meedio.

The layout of this remote is acceptable, but the buttons are not particularly well marked. The remote does have 4 multifunction keys, labelled A, B, C and D, but these would benefit from colour labels instead.

MSI Media Centre PC

This remote does contain coloured keys, but has two sets of the same colours. Not only is this confusing to the eye, it would be unclear as to which set of coloured keys would relate to a colour on-screen display.

Most of the other keys on the remote are of the same size and colour, and cannot be identified easily. E.g. the play control and volume keys are no more prominent than the number pad.

1.5.1 Summary

The following points should be considered when designing the remote.

- Dedicated play control keys (play, next, previous etc.) that are clearly marked so media playback can be controlled at all times.
- Coloured multifunction keys with a multitude of functions.
- Avoid a high number of keys dedicated to functions that are seldom used.

1.6 Specifications and requirements

Although the existing solutions do meet some of the basic requirements for a media centre solution, each solution reviewed had several major drawbacks, especially when handling a large volume of media. Furthermore, some are reliant on specific hardware and operating systems, which may require considerable modification to a user's software/hardware set-up.

The requirements take into account the features a target user would expect from media centre. Possible target users are anyone who enjoys watching or listening to content on their computer, but would prefer to enjoy their media from somewhere more comfortable than at a workstation. While the interface will be designed for operation by users with little computer knowledge, it is generally intermediate to advanced users that utilise their computers for mass-storage of media.

The following specifications have been devised:

- Simplistic GUI designed for output to television, with customisable appearance: colour schemes and wallpapers.
- GUI dedicated to operation via remote control. The keyboard and mouse will not be required during general use.
- Quick and effortless set-up on a multitude of computer systems.
- IR remote control operation, using keys familiar to all remote controls, such as: directional arrows, select/back buttons, numeric and coloured fast text keys.
- Low cost non-proprietary remote control hardware or compatibility with existing hardware if already present.
- Support for the most widespread media formats: MP3/WAV/WMA audio, MPEG 1/2/4, DIVX/XVID/OGM & DVD video, JPEG/GIF/BMP images.
- Playback interface showing progress bar of time-based media, elapsed/remaining time, repeat/shuffle functions, plus standard functions such as pause, next, previous.
- Efficient media database, allowing browsing/searching by artist, album genre, track, and listing tracks by album order or alphabetically.
- Intuitive file browser for when accessing media from memory, cards, CD devices and other media not in the database.
- Predictive and non-predictive text input for searches using numerical keys on the remote. Predictive mode will use the media database in place of a T9 dictionary on a mobile phone.
- Playlist functions, including creation, editing, saving, and auto-recalling of playlists from previous days. Slideshow function for images.
- Cross-platform compatibility: the solution will aim to be compatible on Mac computers. This will be a benefit for Mac users as there is very few media centre solutions for use with OS X.

1.7 Professional considerations and requirements

The BCS 'Code of Conduct' is a set of ethical standards defined by the British Computer Society governing the computing profession in Britain. Even though a project of this nature is affected very little by the issues raised; the code of conduct was studied throughout project development to ensure compliance.

The nature of any media centre-based software is to exhibit media already present on a user's computer or network drive. The software will respect the user's privacy by only listing and displaying media that it has been instructed to by the user. This will be attained by requiring the user to select folder(s) containing media, as opposed to searching local drives for all content. Allowing the user to select specific folders gives them the opportunity to omit media they wish to keep private, such as media that does not meet a certain classification. A feature such as this is important, as media centre software is often designed for use by a number of persons, such as a family.

Implementing a password system to restrict access to particular folders, and the folders configuration itself, would prevent unauthorised users from using the media centre to access private content. Incidentally, the media centre is designed for private use, so does not require the security and rigidity of an application that is designed for use in public.

2.0 Design

2.1 Choice of design paradigm

The model-view-controller has been chosen as the design paradigm as a media centre type application can be easily divided into the three components. The main **model** for the system is the media database, which must be designed carefully. However, media files, their ID tags and the file system are considered the model too, and are defined already.

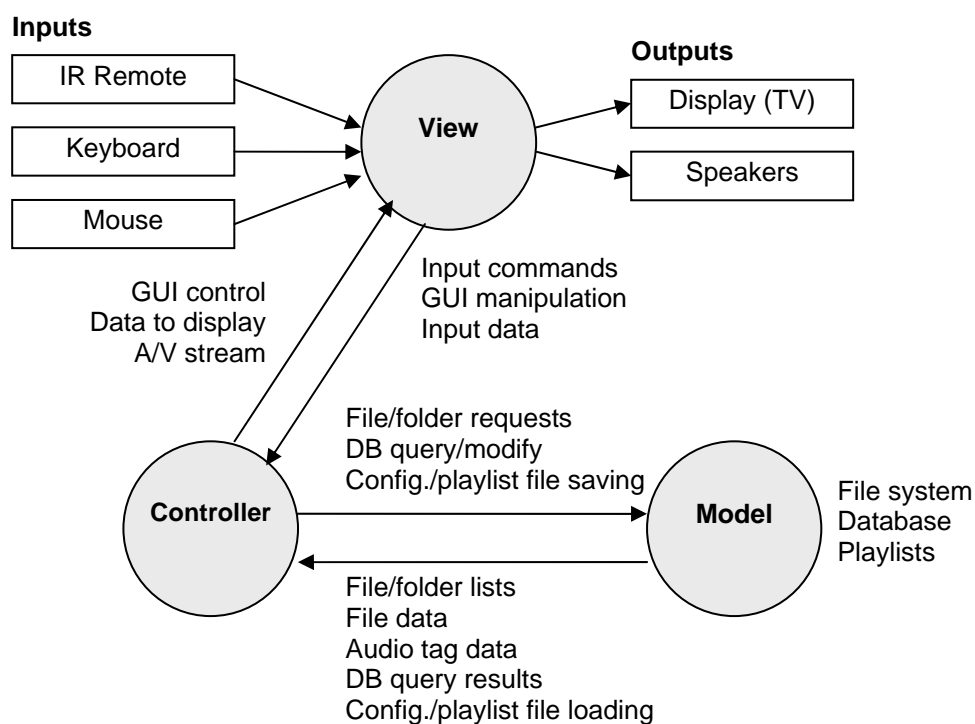
The **view** is essentially the GUI components and a/v output, but is closely related to the **controller**. For example, when keyboard input is received, the controller must update the view appropriately as the GUI is navigated. Similarly, if the GUI receives a mouse click, it must inform the controller.

2.2 Model View Controller paradigm

Both the design and implementation stage have been split into the sections defined in the MVC architecture. **View** is considered first, which details GUI design. Secondly, the **Model** is considered, which investigates the data models that are required or may be encountered. Finally, the **Controller** is investigated, which defines how the system should operate to ensure it meets the specifications set out in Background Studies.

The diagram below shows an overview of the system inputs/outputs, the required/expected data models, and the flow of information between each.

Figure 03: *Model-view-controller overview for a media centre application.*



2.3 MVC: View

2.3.1 Graphical user interface

When considering the GUI, it is important to realise some important considerations:

- What hardware will be used to display the GUI?
- Who is likely to view/interact with the GUI?
- Where is the GUI likely to be viewed from?

2.3.2 Technical considerations

Resolution: A media centre application is designed to be displayed on a standard television set. There are fundamental differences between computer displays and TV sets, which puts constraints on GUI design. The PAL I (Phase Alternating Line) broadcast standard used by all UK televisions uses a principle of lines and fields (625 lines). A resolution of 720x576 most desirable for PAL conversion as it contains the same physical parameters. However, the ratio is not 4:3, so is seldom found on computers. The nearest common 4:3 resolution is 800x600, which is supported by most TV-out devices too. The fixed resolution has the advantage that the software not need to adapt to different resolutions.

Safe Area: Unlike computer monitors, domestic television sets overscan by an arbitrary amount, approximately 5%. Consequently, there is no guarantee that objects placed near the screen edge will be visible on all sets. Meaningful information, such as text, should be placed at least 10% from the screen edges. Full-screen video and images should run to the edge to prevent gaps.

Serif fonts: Unlike progressive-scan PC monitors, PAL uses interlacing, which can cause thin lines to shimmer. To prevent display problems, serif fonts (such as Times New Roman) and thin borders should be avoided. A font such as Arial is ideal.

2.3.3 Target environment considerations

GUI size: As the media centre is to be operated at a distance from the TV a user would normally sit, the text/display items must be of an appropriate size. An approximate size was obtained from observing the GUI of a digital broadcast receiver.

Colour: The colour scheme must be chosen carefully to ensure clarity. A customisable system is best, but the defaults should satisfy most users. Unlike RGB monitors, most TV sets are calibrated by the end-user, thus colour may vary significantly.

A GUI is required that meets the considerations and the specifications set out in Background Studies. GUI design has been broken down into the following sections:

- **Main menu** for navigation.
- **Media library** to browse/select media from the database.
- **File browser** for removable devices or media not in the database.
- **Audio player** for viewing current track information and the current playlist.
- **Video player** and **video queue** for playing a list of videos.
- **Image browser** and **slide show** for viewing images.
- **Configuration settings** to change the look/behaviour.

2.3.4 A user-centered design approach

The main usability/ user-experience goals that are important for the software include:

- Efficiency
- Learnability/Memorability
- Enjoyable
- Utility
- Safety

The product is designed for everyday use and must be efficient. If the solution is too complicated or time-consuming to use, users will be put off. *Enjoyability* is very important: the product will be enjoyable to use if it can perform tasks quickly, easily. Additionally, ensuring the product is *learnable* will allow users to recall how they previously performed tasks. The solution will meet *Utility* goals if it performs the functions set-out in the specifications. User *safety* must be considered when potentially destructive situations arise. For example, a warning and option to cancel should be displayed if users are about to clear the current playlist. This is a problem with many PC-based audio players.

Similarly, design and usability principles are important to software of this nature.

- Visibility of system status
- Consistency and standards
- Constraints and error prevention

Constraints and error prevention is related to safety: constraints should prevent errors occurring in the first place. Good consistency and standards aids learnability, as users are more likely to remember how to use the software if the design and operation is consistent throughout. Finally, visibility of system status should be achieved by ensuring it is always obvious what the system's current activity is, and if user input is required.

During remote control design, *mapping* is important. Mapping involves simple checks, such as ensuring the play control keys are not in a confusing order, or the coloured control keys match the display on-screen. Cultural constraints, such as a red cross (X) symbolising a problem has occurred, will be taken into consideration.

2.3.5 Generic display components

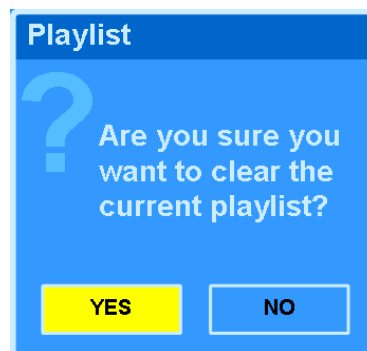
Generic display components will be re-used throughout the software. Besides general navigation using directional keys and enter/back keys, a method is needed to present the user with the different choices that may be available. Instead of using many dedicated remote keys, a set of recognisable generic keys is far tidier, but their current functions must be clearly displayed. The chosen method is to implement 4 coloured menus at the base of the screen that are activated using the coloured ‘fast text’ keys on a remote. Only currently enabled menus will be shown. This is a viable alternative to vertical lists that other solutions employ, as these waste screen space. The design below presents an example generic menu system.

Figure 04: 4-colour generic menu.



Dialog boxes are important generic components. They are used to notify the user of an event, or if a decision must be made. The dialog box should provide space for several lines of text, and 1-3 buttons. The dialog box should steal focus, preventing the user from continuing without acknowledging the message or selecting an option. Cultural constraint: If the box is presenting an error, a red X should be displayed to alert the user there is a problem.

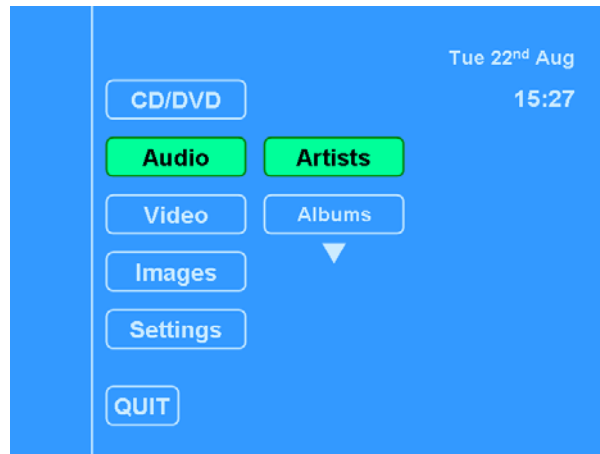
Figure 05: Generic dialog box



2.3.6 Main menu

The menu interface is what most users will see upon first use. It must be inviting, intuitive and present the system's capabilities at-a-glance by displaying all main menu items without needing to scroll. Additionally, useful information such as time/date is recommended, but no more to prevent clutter. Customisable wallpapers will allow the user to personalise the centre, similar to hanging a picture in their living room. A submenu will be required for most menu items, but a third level should be avoided to prevent confusion.

Figure 06: Main menu



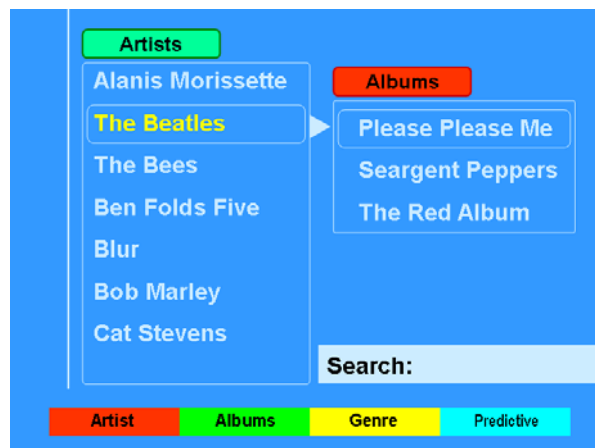
2.3.7 Media library

The media library is perhaps the most advanced interface; however it should be made easy to use. The media library should allow multiple ways of viewing and searching for media. The specifications are important here, and the library should provide the following:

- Listing by artists, albums, genres and songs.
- Predictive and non-predictive seeking or lists, or searching by artist, album, genre or song.
- Songs listed alphabetically or by track number.
- Allow entire albums, artists or genres to played/queued easily.

Most existing solutions contain the option to search *or* browse by category. In order to meet the specifications, browsing and searching should be integrated so a search can be performed, and the results can be refined by further browsing. A provisional interface is provided below, but the library's capabilities will be discussed in the **controller**.

Figure 07: Media library



2.3.8 File browser

Most of the file browsers in existing solutions lacked fundamental features. The specifications for the file browser should include:

- Display supported media files only.
- Files should be played/queued by the media centre only.
- Refining by a certain media type (e.g. videos only).
- A folder tree should be displayed.
- File properties should be available on request.

2.3.9 Audio player (Now Playing)

The audio interface is designed to play audio, show media information and to edit playlists. The interface should include:

- Artist name, album name, track name, track number display
- Elapsed time or total time
- Album art or visualization
- Current playlist

Elapsed time can be expressed over total time with a progress bar. It should be made easy to advance to any song in the playlist, or to modify the order by moving or deleting tracks.

Figure 08: *Now playing/playlist editor.*

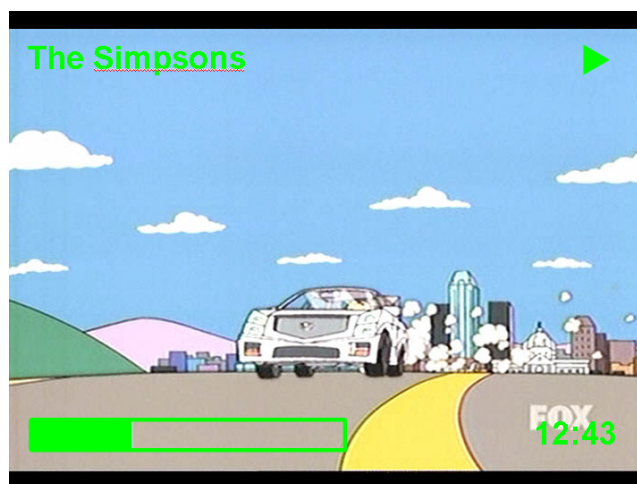


2.3.10 Video player

Video playback is similar to audio, but with the inclusion of a video display. Videos of all sizes should be enlarged to fill the screen, but while maintaining their aspect ratio. Title information and elapsed time should be available on request, but hidden during playback.

The system should facilitate the queuing of videos to play sequentially. This would be most practical by using a dedicated interface instead of an overlay.

Figure 09: Video playback and OSD



2.3.11 Configuration settings

Configuration settings must be kept straightforward and easy to navigate using the remote. The sample components designed in Flash were based on the 4 common types found on most operating systems:

- Tick boxes
- Radio buttons
- Drop-down boxes
- Text boxes

Figure 10: Configuration components

☒ Check for new files at start-up.

☐ Show AlbumArt if available.

☐ Always show visualisation.

My Photos

Main Menu Colour:

Blue

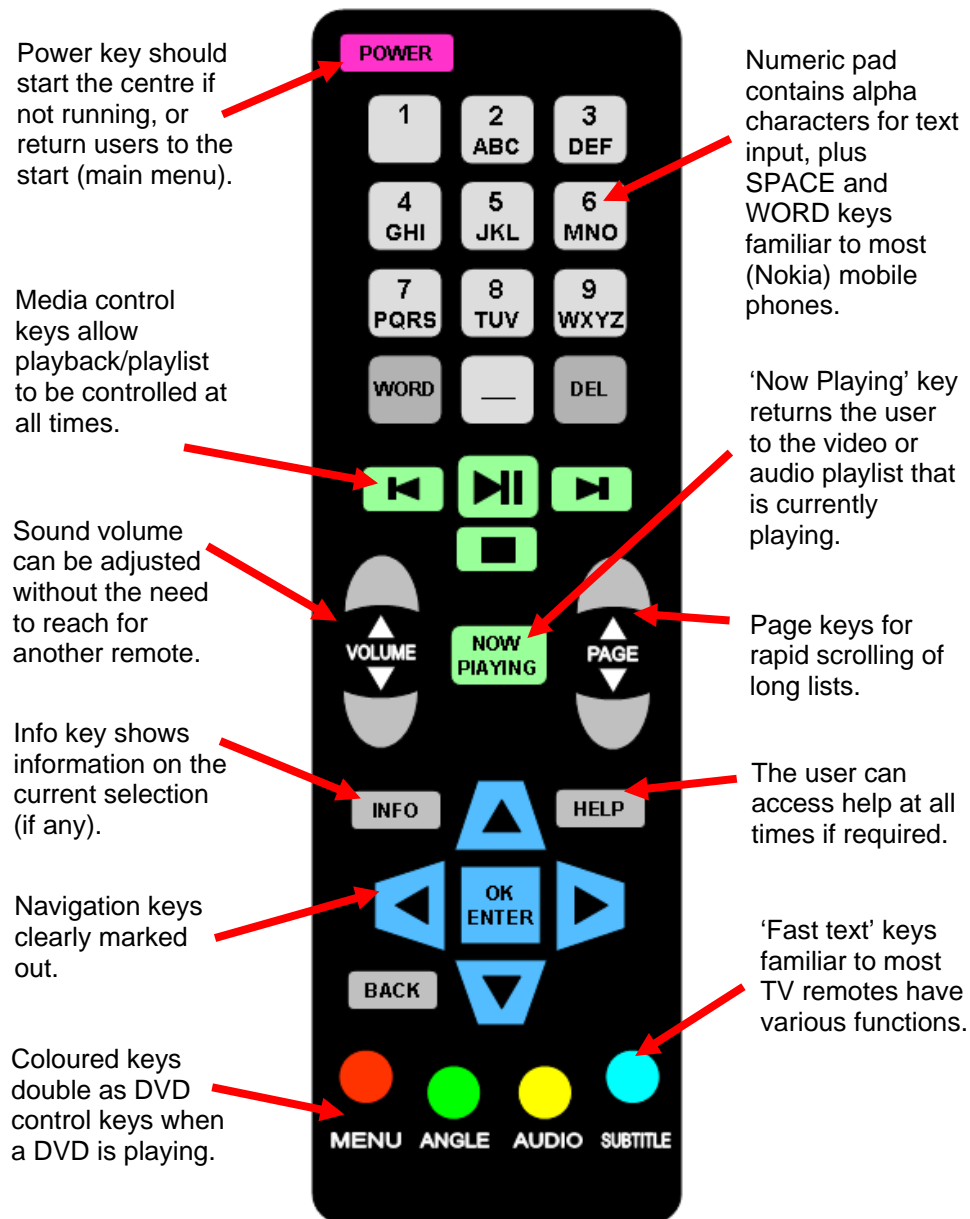
2.3.12 Remote control

The remote is the main input device for the system. The design is equally as important as the GUI. The design should be similar to most other household remotes, and meet the specifications and requirements.

- **Standard navigation:** Directional keys are required for navigating/selecting lists and menus. A 'back' button is needed to back out of a selection, and page up/down keys will aid navigation of long lists
- **Alphanumeric pad:** Most a/v remotes feature a numeric pad, but this solution allows text input from the remote, so requires each button to be marked with alpha characters. For predictive text, it makes sense to place the 'word' key in the same location as most mobile phones (primarily Nokia).
- **Media control:** Dedicated keys to control media and volume should be clearly grouped so playback can be controlled at all times.
- **General function keys:** Coloured multifunction 'fast text' keys that meet the need of the GUI are required, and will be familiar to most users.
- **Help/information:** A dedicated context-sensitive help key to assist the user if something is not understood. Additionally, an info key will be a valuable method for obtaining extended media information.
- **DVD playback:** DVD discs have several instant-access functions for use during playback. A permanent on-screen display of functions would be irritating, so dedicated keys are a necessity. The coloured 'fast text' keys are ideal for dedicated keys in this situation.

The number of keys on a remote is critical. Too many specific keys will result in confusion and many keys may only be seldom used. Too few and each key may have to control several functions, which may be hard to label.

The following remote design is a guide based upon the considerations above. An attempt will be made to find a physical remote of similar specification.

Figure 11: *Proposed remote layout*

2.4 MVC: Model

To discuss the model, it must be realised what types of data model the media centre requires, or is to encounter.

- Audio tag data
- Media database
- Disk and network file-systems
- M3U Files

2.4.1 MP3 ID3 tags

A media centre that only employs a tree-structure of folders and files would be no more than a file browser.

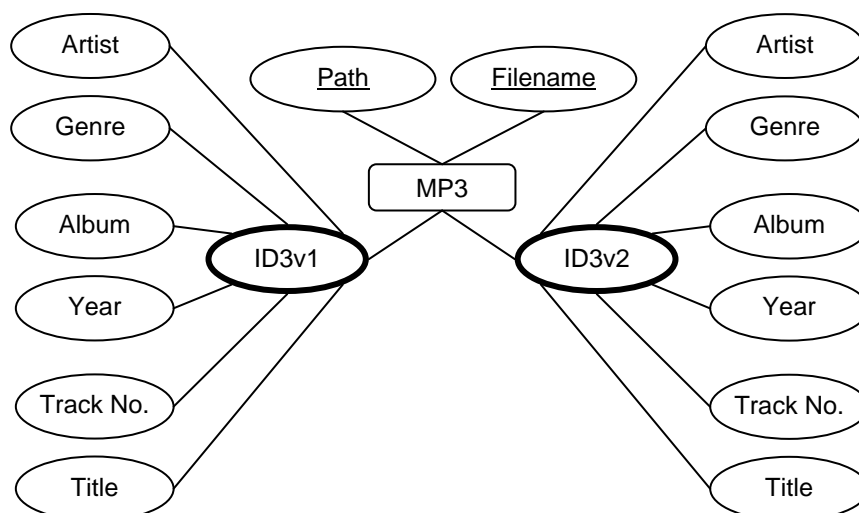
Many JPEG image and MPEG audio files contain tags that hold information about the file. The EXIF tags in JPEG files are mostly exposure information recorded by digital cameras. MP3 ID3 tags, however, are widely used to store the artist, album, track name and other details. ID3 tags supersede filenames as the information is categorised, and can be used to create a media database that can be queried in numerous ways. As most MP3's originate from audio CD's, MP3 converters use CDDB servers to complete the ID3 information accurately. There are two versions of ID3 tag: V1 tags have a fixed number of fields and field lengths. V2 tags allow any number of fields with larger capacities, and later versions even allow album-art to be stored.

There are some concerns with ID3 tags when they are used to create a database:

- An MP3 may not contain any ID3 tags, or the tag information may be incomplete.
- The v1 and v2 tags may contain contradictory information.
- A group of MP3 files that make up an album may have contradicting ID3 tags.

Consider the following E-R diagram that represents the data a single MP3 on a disk may contain.

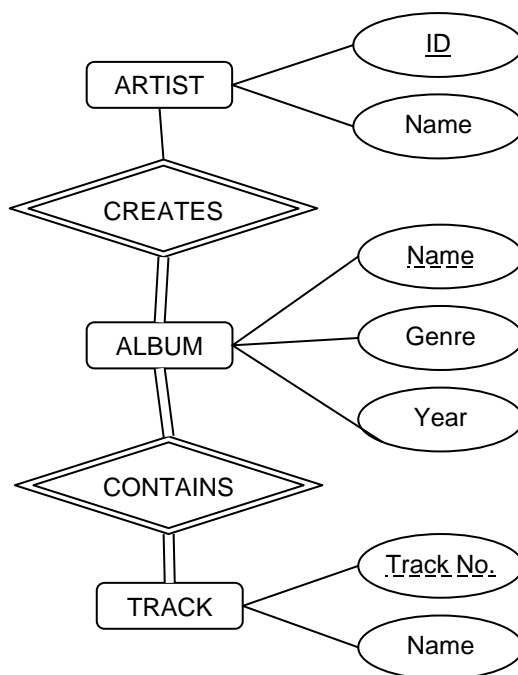
Figure 12: E-R representation of MP3 tag data in a file system.



There are no relationships. An album comprised of 10 MP3 tracks effectively contains 20 individual artist fields, excluding the filename.

Now consider a database allowing users to catalogue their CD collection. The relationship includes weak entity types, to ensure all tracks belong to an album, and all albums belong to an artist.

Figure 13: *E-R model for a CD collection.*



Although this model would ensure all tracks are part of an album, it would not permit the storage of individual tracks that do not belong to an album.

A decision must be made into an appropriate database design.

**Should a database with relationships be produced,
and bad tag data manipulated to fit the constraints?**

OR

**Should a basic database be implemented that may contain poorly organised data,
but is faithful to the original tags?**

2.4.2 Existing databases

Before deciding upon a database design, two existing media centre databases were investigated. Both **Meedio** and **XBMC** (Xbox Media Centre) use SQLite databases, which can be viewed easily. Both systems were populated with the same music for comparison. It was discovered the two databases are very different in design.

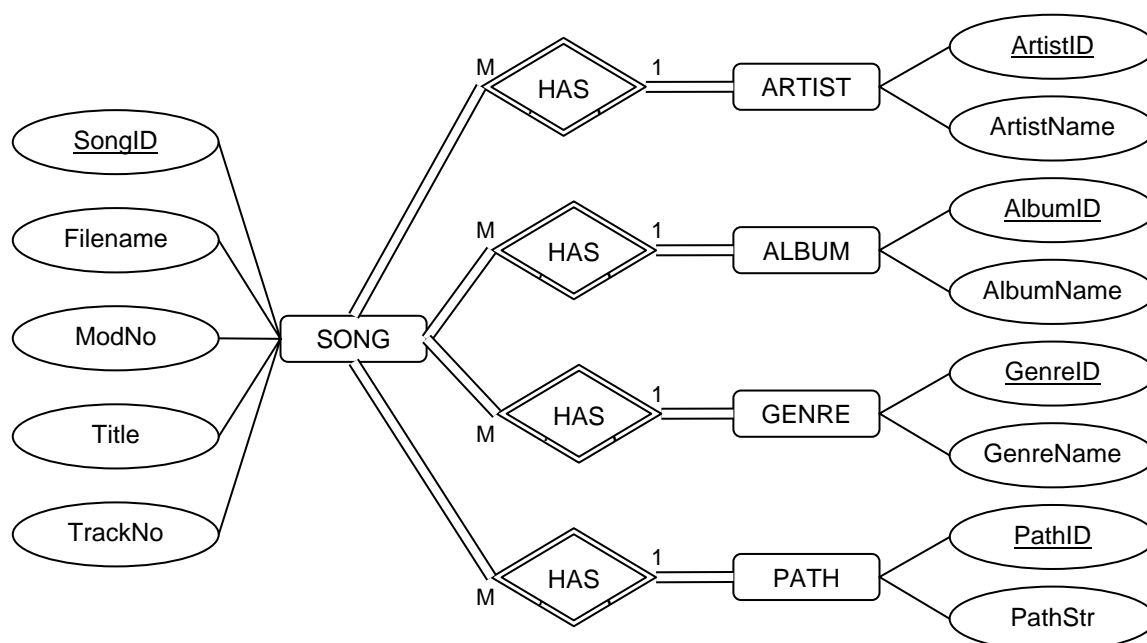
Meedio: The Meedio database is simple in structure, but very large at 44MB. A single table 'Items' is used to store a record for each MP3, and all its details. Each entry contains an ID, path with filename, modified number, image location, and 48 tag columns, most of which are unused. The tag columns are named tag_1 to tag_48, and a separate table 'tags' defines their actual names. The Meedio database contains a great deal of repeated data and empty fields that constitutes to the large file size. However, program code to populate this database would be simple.

XBMC: The XBMC database is more complex, and only 2.7MB in size. The design is somewhat similar to the CD collection ER model. Several tables are used to hold data: 'Artist', 'Album', 'Genre', 'Path' and 'Song'. The first four tables contain an ID and string for each unique artist, album, path etc. The 'Song' table contains an entry for each MP3 file, but uses ID's for the artist, album, genre and path. Each record also contains a name, title, date, and some basic tag data. This design is far superior as there is much less repeated data and very few empty fields. Although program code and for this database will be complex and computationally expensive, it should be more efficient during use.

2.4.3 Chosen database design

The benefits of the XBMC database were obvious, and the chosen design is based upon multiple tables. The design requires every MP3 added to have an Artist, Album, Genre, and naturally a path. If any of the tag information is missing, bogus information will be used in its place, for example "Unknown Artist".

Figure 14: Chosen database design E-R diagram.

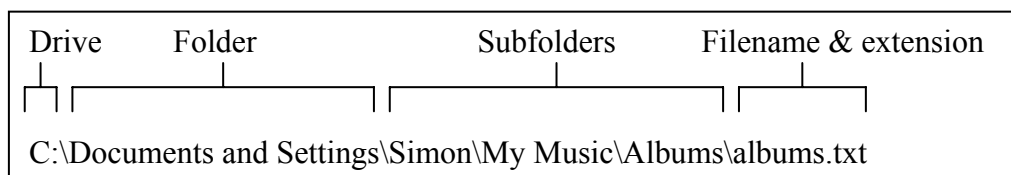


In addition to the information fields, each song (MP3) has a ModNo field to store the file's hash. When the database is updated, the hash is checked to see if the file has changed. If a change is detected, the database is updated. Incremental updates are important as generating the database from will be time-consuming.

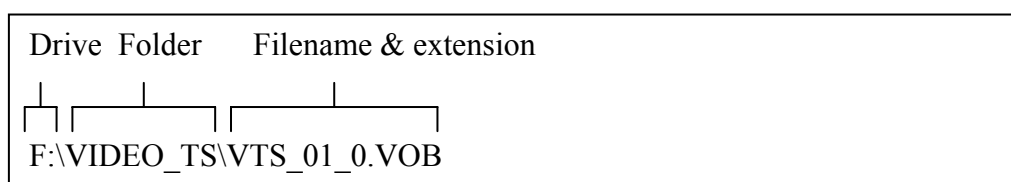
2.4.4 File system

The media centre will access media from hard disks, removable media and network locations. The file system models are established and handled by the operating system, and most implementation technologies provide simple methods for reading and writing files. However, system will need to interpret file and UNC paths. Some examples are shown below.

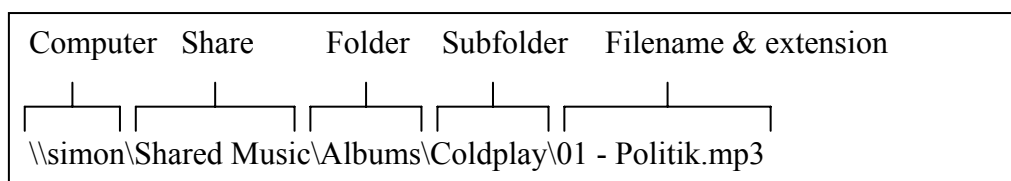
A typical file path for a local disk, with several levels of subfolders:



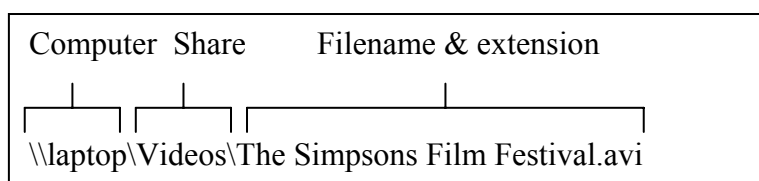
A typical file path to an optical drive containing a DVD video disc:



A typical UNC network path to a computer named 'simon' and share named 'Shared Music'.



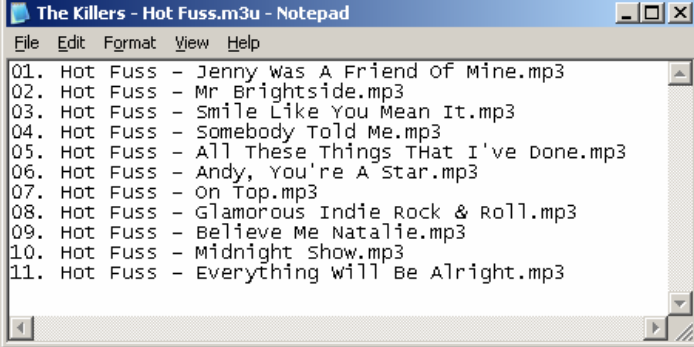
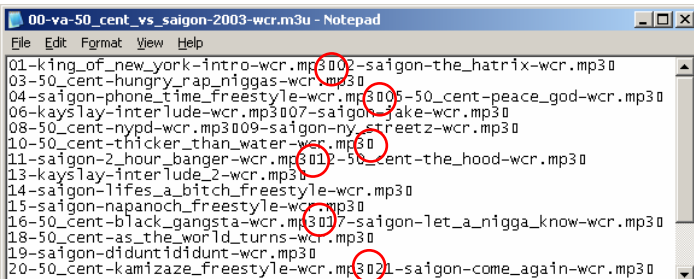
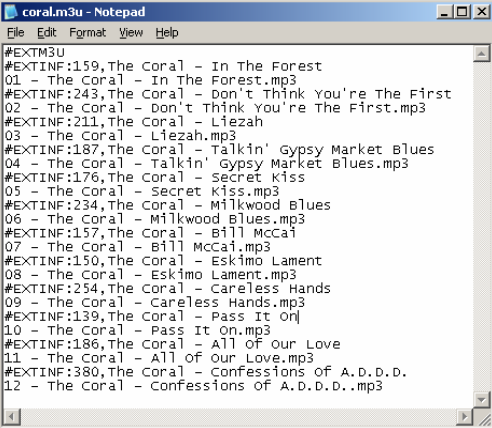
A typical UNC network path to a file in the root of a share.



Mapped network drives, and mass-storage devices (e.g. Flash Pen Drives) appear as local drives and can be treated in the same way. However, the drive letters may not always be available.

2.4.5 M3U playlists

The M3U universal playlist is an ASCII list of media files. There are two types of M3U files: basic and extended. After examining sample playlists, it was noticed that some files contained line feeds only, and not carriage returns. Line feeds appear in Windows Notepad as small squares (□) where the string should begin on a new line. There is a possibility the chosen development platform may not detect new lines properly.

Basic Format	Extended Format
 	
<p>The basic M3U format is a list of filenames separated by carriage returns. Note in the second example, the file only contains line feeds represented by square characters. The entire M3U will span a single line when word wrap is disabled.</p>	<p>Extended M3U files contain an extra line for each file, containing the total time in seconds and a display name.</p> <p>The extended information is generally acquired from the ID3 tag, and allows the file's title and play time to be displayed without reading the tag.</p>

2.5 MVC: Controller

The controller discusses how the user is to navigate the system to perform typical tasks.

2.5.1 General navigation

General navigation will be performed by using the remote's directional keys to move between items. The OK/Enter key will select currently highlighted option. The user should be able to return back to the previous menu, or the last screen they were viewing where appropriate. E.g. if the user navigates to the now playing screen from the file browser, the back key should return to the file browser.

2.5.2 Typical tasks

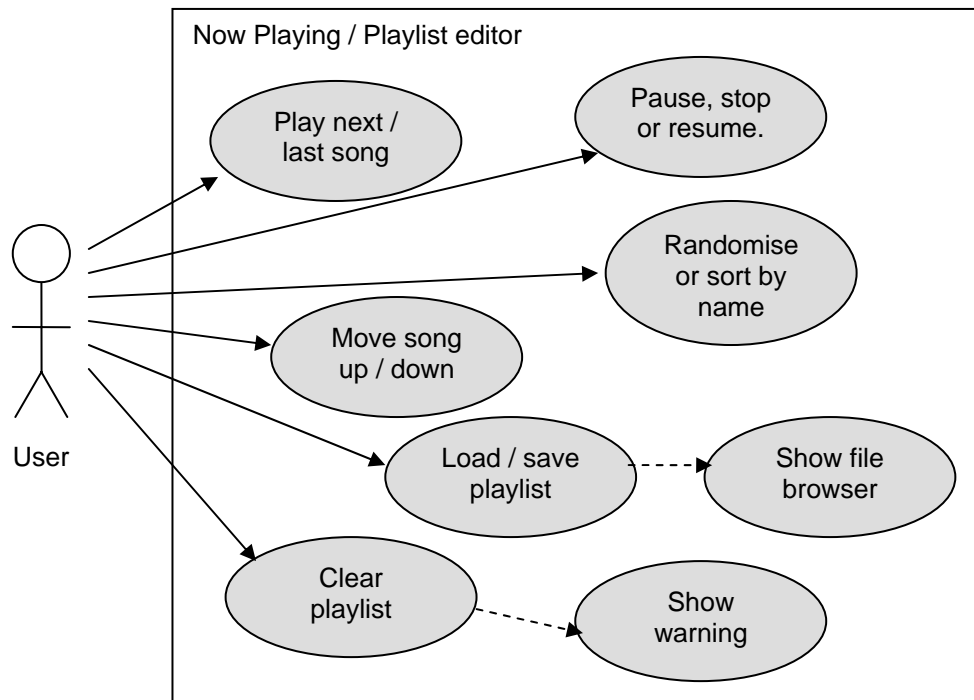
The typical tasks that are likely to be performed in a media centre include:

- Playing an album, or selection of albums.
- Searching for a song where the artist is not known.
- Several users queuing up songs for a party.
- Watching a selection of videos or a DVD.
- Viewing sound, videos or pictures from a PC or memory card.
- Playing an album and listening to it while viewing photographs.
- Adding more media from a new network location.
- Changing the way the media centre operates.

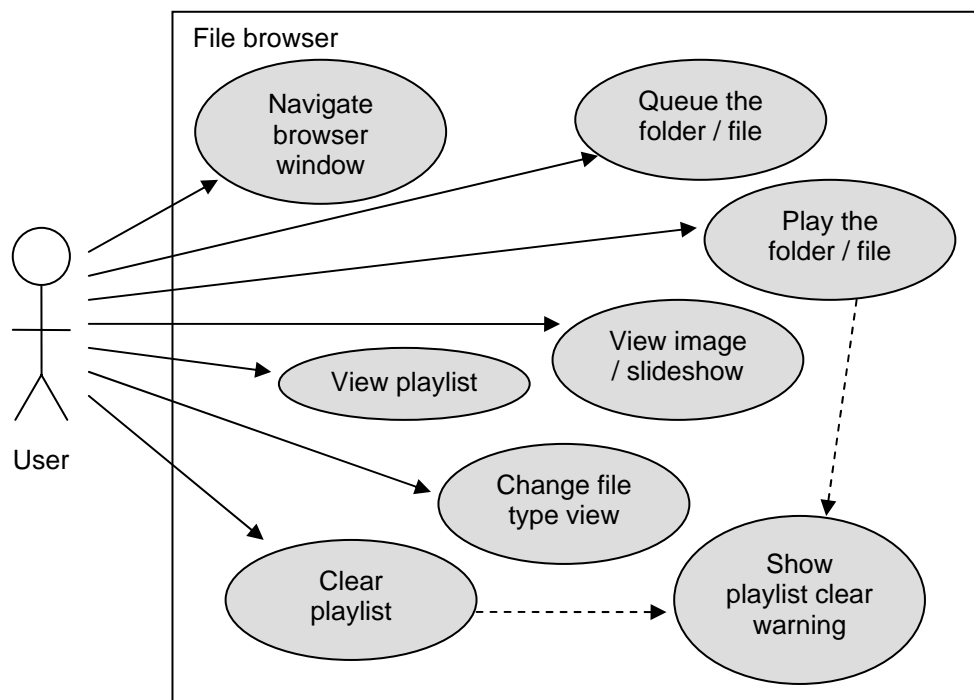
All the typical tasks would be initiated by selecting the desired menu item from the main menu. If the user is not at the menu, the Media Guide key should return them to it. If the user wishes to perform two functions, such as playing an album and viewing photographs, the first functions should be completed in the same order.

2.5.3 Use cases

Use cases are a good way of showing the user-system interaction for a given task or scenario. There is only one type of user considered to use the media centre. The now playing screen is ideal to hold the playlist editor, as it will also show the upcoming songs.

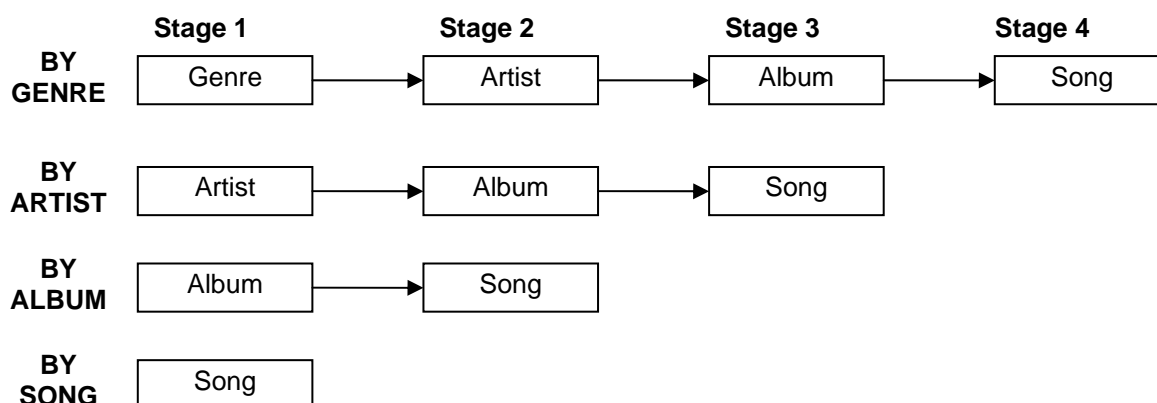


Besides list navigation, the file browser will require actions for the media.

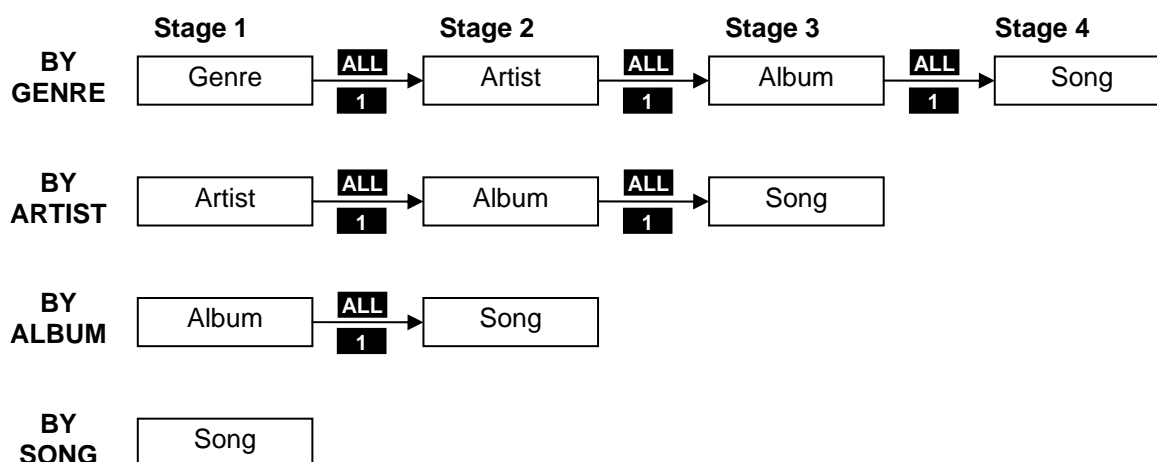


2.5.4 Library browsing

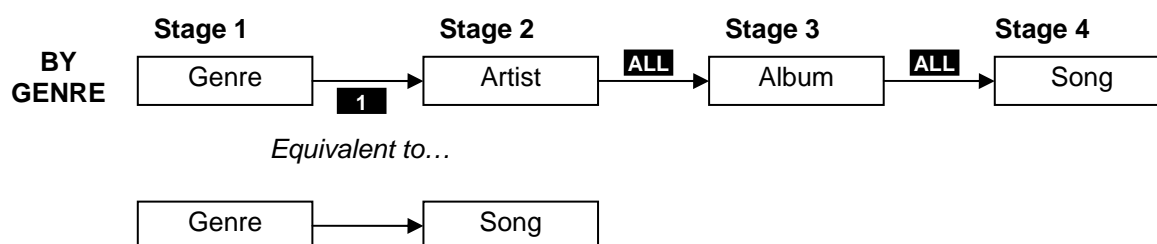
Many of the existing media libraries do not have flexible browsing options. Although many feature browsing by artist, album, genre or song, the next stage often shows all the matching tracks. There is no option to group the results by another category. The proposed design allows browsing to be refined by up to 4 stages. The 4 browsing options reflect the most practical browsing options.



However, browsing flexibility can be greatly increased by adding a 'Select All' item to the top of each list. Now, the user is not forced to refine at every stage, which allows far greater browsing flexibility.

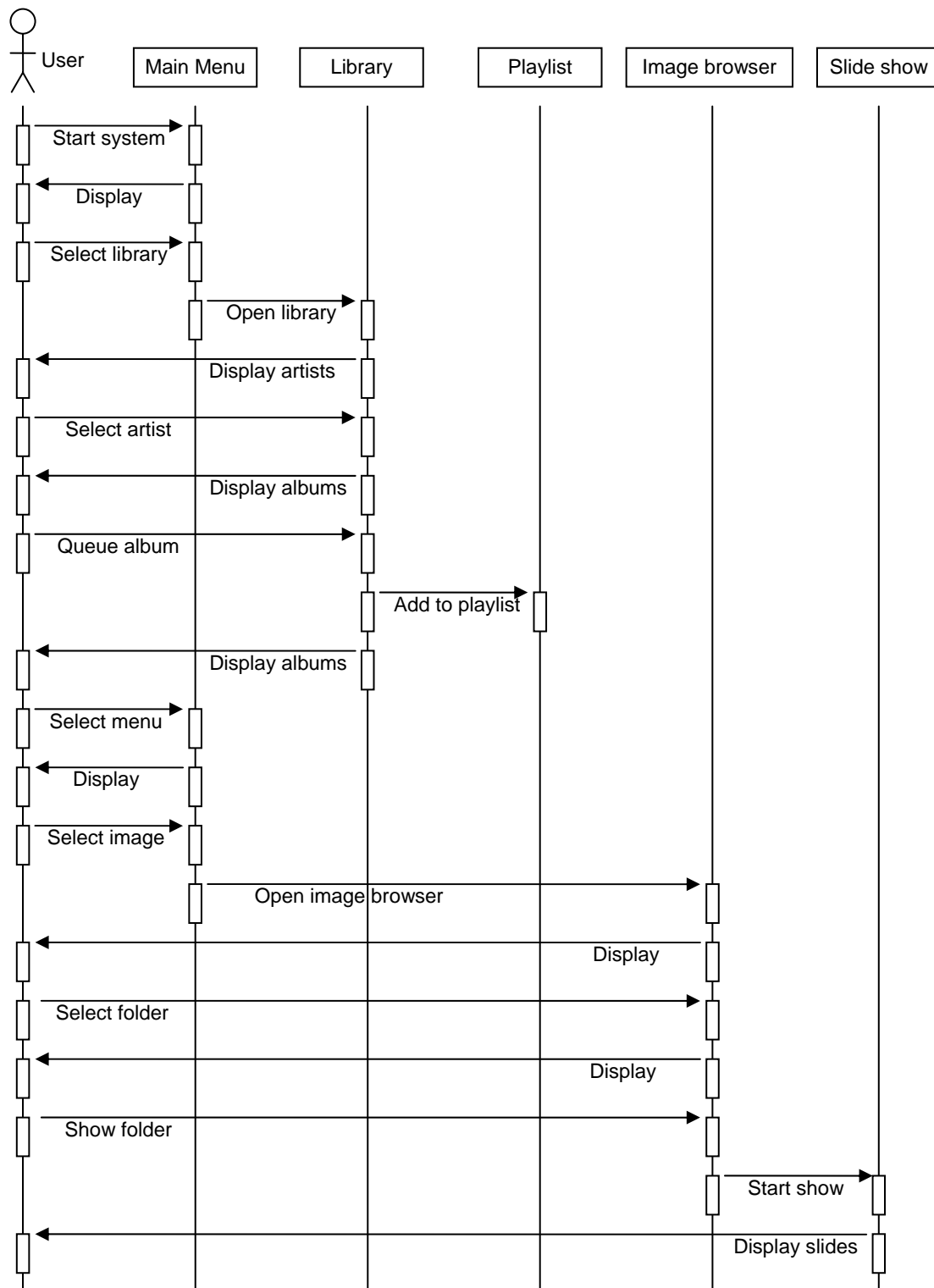


For example, if the user wishes to view a list of all songs that match a genre, they can use the ALL function at stages 2 and 3 so not to refine by artist and album.



2.5.5 Sequence diagram: *Playing music and viewing pictures*

As discussed in the specification, a user may want to play a selection of audio files, then watch a slide show while listening to the music. The following sequence diagram shows how the user would perform this by switching between the appropriate sections of the media library. For the purpose of the exercise, screens take the place of classes in the diagram.



3.0 Implementation

3.1 Choice of technology

When considering the choice of technology, the functions requiring implementation were considered:

- Bespoke GUI
- Audio/video playback
- Image file display
- Database
- Remote control support
- Cross platform compatibility

On first sight, Java was considered to be the most suitable platform for development. The Java Media Framework, or JMF, is an optional package enabling Java applications to play time-based media. JMF contains built-in support for media, which allows for cross-platform playback without relying on system codecs. However, playback is limited to basic media types (MP3, MPEG1/2, QuickTime), and to add support for additional audio/video formats would be beyond the scope of the project. JMF 1.0 was introduced in 1997, and has undergone significant development in recent years to attain version 2.1. Consequently, there is limited up-to-date online material and books available to aid development.

Due to the shortcomings of Java, Macromedia Director MX 2004 has been chosen to develop the solution. Director is most suitable for embedding multimedia content, rather than sourcing it from the user's PC at run-time. However, with the addition of plug-ins known as Xtras, Director can support an SQL database, ID3 reading, all popular media formats, and remote control support. Director is also fully compatible with Macromedia Flash, which allows vector-based content to be imported into Director. Unlike Java, there are no compatibility limitations with recent codecs, as Director utilises OS-installed codecs. Also, Director applications are Mac compatible which will aid cross-platform compatibility.

The latest version of Director (MX 2004) has been updated to support JavaScript, in addition to Director's legacy programming language **Lingo**. Although the author has greater knowledge has experience with Java and Flash ActionScript, Lingo was the favoured as it is well-established with Director and most code examples on the Internet are written in Lingo.

3.2 Resources required

A PC containing a selection of common media types, connected to a TV using composite or S-Video. Infrared receiving hardware/software and a remote control is required for interfacing the software with a remote, so the software can be tested in its target environment.

A Mac to the same specification is needed to test Mac compatibility. Alternative infrared hardware may be required.

Both computer set-ups may require additional software to play all formats, but this will be discussed during implementation and testing.

3.3 Overview of code structure

The media centre comprises of **movie**, **behaviour** and **parent** scripts. These all function differently:

- Variables/methods defined in movie scripts are available.
- Behaviour scripts are assigned to frames.
- Parent scripts are equivalent to classes, and instances known as child objects are created where necessary.

Movie Scripts	
main	Initialises instances of parent scripts, creates the backHistory property list; clears/initialises input variables.
sessionManager	Manages data appropriate to the session, such as global variables.
playManger	Initiates and manages audio/video playback and slideshows.
inputManager	Maps keyboard input to remote functions; handles Flash GUI clicks.
Parent Scripts	
CONFIG_MANAGER	Handles saving, loading and accessing of configuration.
FILE_MANAGER	Handles folder, ID3 and text file reading, album art detection, saving/loading M3U files and filename/path string methods.
PL_MANAGER	Contains a playlist and handles playlist manipulation. An instance for audio and video is created.
DB_MANAGER	Handles all DB routines, including predictive queries and library DB routines.
FL_MANAGER	Handles methods for initialising and controlling flash GUI objects.
Behaviour Scripts	
pre-start	Ensures Flash objects & A/V players are fully initialised at start.
start	
mainmenu	Main menu initialisation and navigation.
library	Media library.
sound	Now playing items and playlist viewer/editor GUI.
pre-video	Initialises video sprites before video.
video	Handles video playback and OSD for all video players.
postVid	Ensures Flash objects are re-initialised after video playback.
postVid2	
picture	Thumbnail image file browser.
browser	Multi-purpose file browser.
slideshow	Image slide show.
dvd	Interaction with DVD playback Xtra.

Additional scripts not written by the author include: MP3_PARSER, FILE_BINARYIO, FILE_FILEIO, DATA used by the MP3 parser.

3.4 MVC: View

Director's built-in graphics tools are very basic. Unlike Flash, graphical objects are not object-orientated, and do not contain their own timeline or code. For these reasons, Flash was used to implement most GUI components. The Flash objects contain ActionScript for initialisation and animation where necessary. However, Director will manage all navigation, such as storing the current position in a menu. Instructions will be passed to Flash to update the GUI component when necessary.

Mouse input will be received by Flash and not Director. Although the solution is designed for control via the remote, mouse support may be useful when the user is within reach. To facilitate this, Flash passes the object names of items clicked to Director.

Flash Mouse Event Method	Mouse Event Received by Director
<pre>onRelease = function() { getURL("lingo: flashClick (\\" + _parent._name + "\\" , \\" + this._name + "\\")"); }</pre>	<pre>on flashClick(buttonParent, buttonName) lastFlashClick = [buttonParent, buttonName] end</pre>

An additional piece of frame behaviour code in Director instructs the frame to call `getMouesClick()` to discover which object in the Flash movie was last clicked.

```
on getMouseClick
  lastFlClick = lastFlashClick
  lastFlashClick = [ "", "" ]
  if not voidP(lastFlClick) then
    return lastFlClick
  else
    return [ "", "" ]
  end if
end
```

Upon first looking at Flash to build the GUI components, Flash does contain its own GUI components such as buttons, text boxes with scroll bars etc. However these are of a predefined size and difficult to change (they cannot be scaled). For a media centre type application where all display items must be of a relatively large size, custom components are required.

Flash objects must be drawn once on the stage so their internal methods are initialised. To ensure this, two movie frames 'pre-start' and 'start' force the flash objects to be drawn. This is performed behind a black rectangle that prevents the user from seeing the flash objects momentarily.

To pass an array to Flash, a new Flash array must be created inside the desired Flash movie using `newObject(Array,"item1", "item2","item3")`. Unfortunately the only way to convert director lists Flash is to use a loop to produce a string, and run the string using DO as if it were code. This is hacky and the statement is not checked at compile-time.

The following method converts a Director list to a string. It also ensures quote marks are successfully handled.

```
on arrayToString(theArray)
  --converts array into string(used for sending Flash Arrays)
  if not voidP(theArray) then
    theString = ""
    repeat with i=1 to count(theArray)
      theString = theString & ", " &QUOTE& searchAndReplace(theArray[i],QUOTE,"'")&QUOTE
    end repeat
    return theString
  end if
end
```

For reusable GUI items, it was decided that it is better to create most components dynamically using attachMovie(). It is important to enable “Export for ActionScript” on objects that are to be manipulated by code.

3.4.1 Customisable appearance

The colour of Flash GUI components is set by applying a transformation filter which can be changed or removed. All objects are set to black in colour (with a gentle gradient). The colour and transparency can be adjusted on the fly.

3.4.2 Remote interface

After researching, the following systems for interfacing IR hardware were discovered:

Simsoft Systems IR Xtra and IRMan (£140)

This uses an Xtra to communicate with Director. However, the Xtra costs over £100, and only operates with a certain serial IR products, such as IrMan (£40). It is not suitable for Mac systems.

<http://www.simsoftsystems.co.uk/products.htm>



Keyspan Digital Media Remote (£30)

This is a USB-based PC & Mac compatible infrared receiver and remote. The supplied remote is not suitable, as it contains too few buttons. The IR receiver is supposed to be compatible with other remotes, but no documentation is provided and the company did not reply to any emails. <http://www.keyspan.com/>

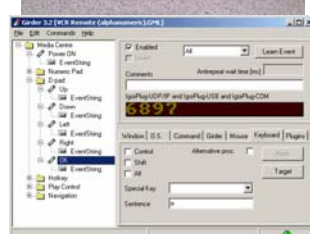


IgorPlug USB & Girder Keyboard Emulation (£16)

This simple USB receiver will work with most remote controls. The device receives the code sent by each button, and a free application, **Girder**, allows functions to be assigned for each key, such as replicating keystrokes.

This is the cheapest and easiest to implement. The device can be interfaced with Director through the keyboard.

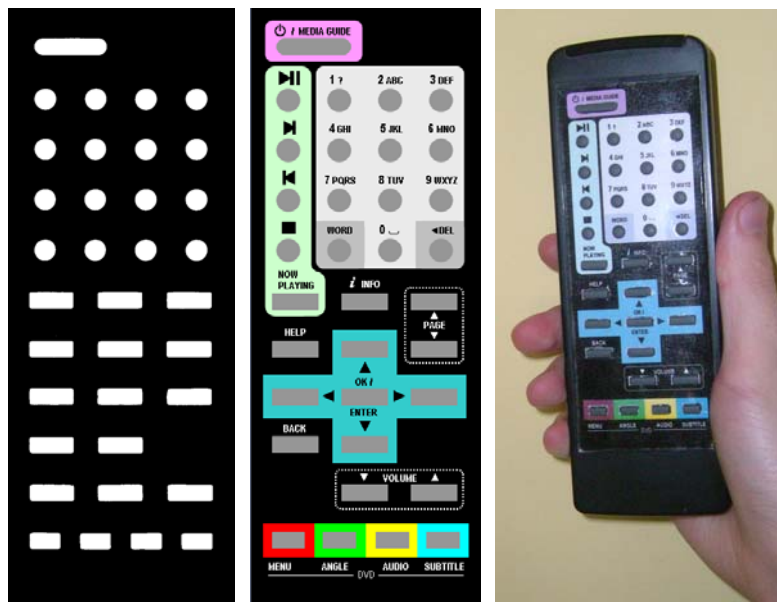
<http://www.gibbsdesign.co.uk/>



The IgorPlug USB device was selected due to the price and simplistic implementation. The receiver is not supplied with a remote, so an obsolete handset with a suitable number of keys and removable face plate was sourced.

The original face plate was removed and scanned to produce a blueprint. The required buttons, as set out in design, were positioned in the best possible arrangement.

Figure 15: *Fabricating the remote*



During use it was found several keystrokes were received for a single keystroke. To prevent this, the anti-repeat delay in Girder was increased.

3.4.3 Implementing keyboard control

Since the selected IR hardware emulates keystrokes, Director must listen for these keystrokes. The following methods were tested:

```
keyDown --does not work unless movie is playing
keyUp   --works, but conflicting key codes are returned when
used with the keyCode and the keyPressed
```

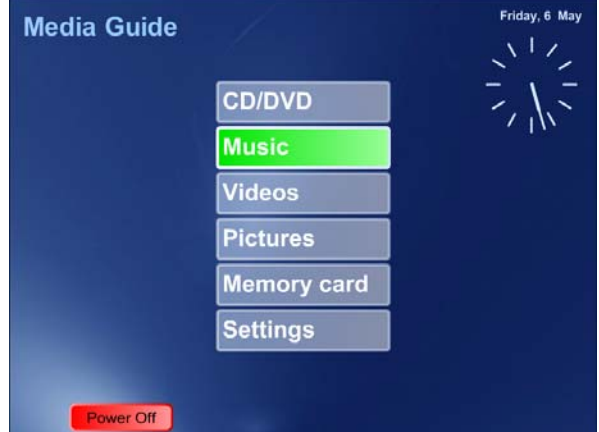


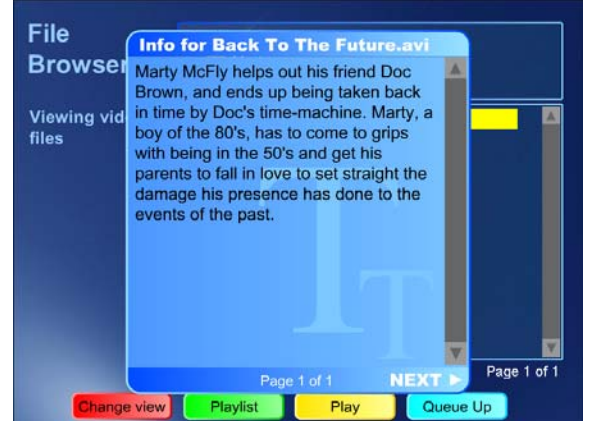
```
the keyPressed --returns conflicting codes
the keyCode    --returns conflicting codes
the key        --returns actual character (most reliable)
```



It was found that “the Key” was the most reliable method, but this only works with alphanumeric keys.



All key mapping is contained within a script (input_Manager) so the remote interface can be easily adjusted. Each key on the remote is translated to a standardised name, such as “playKey”.

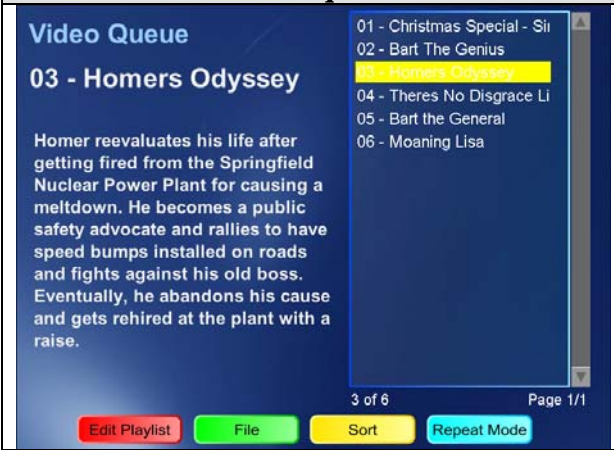
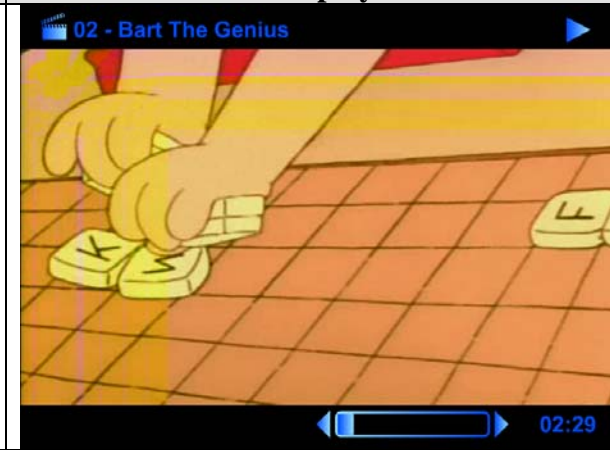
3.4.4 View screenshots


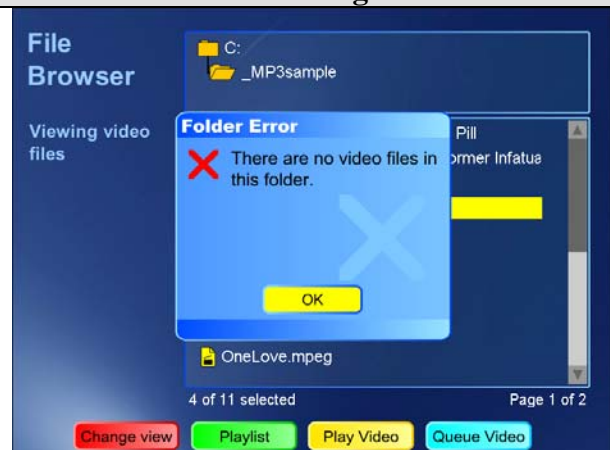
The following screenshots show an example of the completed GUI at different stages.

Main menu	Sub menu
 <p>The Main menu is titled 'Media Guide' and shows a list of options: CD/DVD, Music (highlighted in green), Videos, Pictures, Memory card, and Settings. A clock in the top right corner shows 'Friday, 6 May'. A 'Power Off' button is at the bottom left.</p>	 <p>The Sub menu is titled 'Media Guide' and shows a list of options: CD/DVD, Music (highlighted in green), Artists, Albums (highlighted in green), Genres, Songs, and Playlists. A clock in the top right corner shows 'Friday, 6 May'. A 'Power Off' button is at the bottom left.</p>
<p>Each item on the main menu is a different colour to aid recognition.</p>	<p>When a sub menu is opened the main menu is shrunk and moved into the background.</p>
File browser	Information window
 <p>The File browser is titled 'File Browser' and shows a list of files: Back To The Future.avi (highlighted in yellow), Bruce Almighty.avi, Finding Nemo.avi, Kill Bill 2.avi, and Spiderman 2.avi. A clock in the top right corner shows 'Friday, 6 May'. A 'Power Off' button is at the bottom left.</p>	 <p>The Information window is titled 'Info for Back To The Future.avi' and shows the file's ID3 tag information: 'Marty McFly helps out his friend Doc Brown, and ends up being taken back in time by Doc's time-machine. Marty, a boy of the 80's, has to come to grips with being in the 50's and get his parents to fall in love to set straight the damage his presence has done to the events of the past.' A clock in the top right corner shows 'Friday, 6 May'. A 'Power Off' button is at the bottom left.</p>
<p>The file browser can be set to view all files, or files of a specific type.</p>	<p>Pressing INFO shows file properties, the ID3 tag or a text file that matches the file name.</p>

Picture browser	Slideshow/picture viewer
	
<p>The picture browser shows a folder of images as thumbnails.</p>	<p>The slideshow allows the interval and repeat to be adjusted, and the title can be hidden.</p>

Media library	Now playing/playlist editor
	
<p>The media library allows media from the database to be queued. The predictive window appears when a search is performed.</p>	<p>The now playing window shows information about the current song, and will display AlbumArt in place of a visualisation if available.</p>

Video queue	Video player
 <p>Video Queue</p> <p>03 - Homers Odyssey</p> <p>Homer reevaluates his life after getting fired from the Springfield Nuclear Power Plant for causing a meltdown. He becomes a public safety advocate and rallies to have speed bumps installed on roads and fights against his old boss. Eventually, he abandons his cause and gets rehired at the plant with a raise.</p> <p>01 - Christmas Special - Si 02 - Bart The Genius 03 - Homers Odyssey 04 - Theres No Disgrace Li 05 - Bart the General 06 - Moaning Lisa</p> <p>3 of 6 Page 1/1</p> <p>Edit Playlist File Sort Repeat Mode</p>	 <p>02 - Bart The Genius</p> <p>02:29</p>
<p>The video queue shows information about the selected video.</p>	<p>Pressing INFO while playing a video will show the OSD for 5 seconds. This shows the video name, elapsed time and progress bar.</p>

Question dialog box	Error dialog box
 <p>Now Playing</p> <p>Playlist Editor</p> <p>Save Changes</p> <p>? Save the changes made to the playlist?</p> <p>Yes No Keep Editing</p> <p>Press BACK to done editing.</p> <p>00:00</p> <p>9 (3 / 11 selected) Page 1/1</p> <p>Remove Select None Move Down Move Up</p>	 <p>File Browser</p> <p>Viewing video files</p> <p>Folder Error</p> <p>✗ There are no video files in this folder.</p> <p>OK</p> <p>OneLove.mpeg</p> <p>4 of 11 selected Page 1 of 2</p> <p>Change view Playlist Play Video Queue Video</p>
<p>After modifying a playlist, the user is prompted to save or discard the changes.</p>	<p>If a folder is queued that contains no media files, an error is displayed.</p>

3.5 *MVC: Model*

Director's only method of writing to disk is using the **FileIO** Xtra to read and write text files. There is no method for creating a database structure. However, the first problem that must be solved is the reading of tag data in audio files.

3.5.1 *Reading audio tag data*

Before implementing a database, a method for reading ID3 tag data is required. Director's SWA cast member will return ID3 information if the file is loaded and played, but the procedure is slow and unreliable.

Instead, a script was discovered that uses the **BinaryIO** Xtra to read ID3v1/v2 tag information. This script was obtained from http://staff.dasdeck.de/valentin/lingo/mp3_swa/. The script was found to be very reliable.

ID3v2 tags are preferable as they are not limited to 30 characters per field (the ID3v1 limit). However, some MP3 files may not contain both tags. As the database design requires Artist, Album, Genre and Title information for each track, the following practice is used to obtain information:

- Use v2 tag fields for all information where available
- Attempt to use v1 tags for missing or blank v2 fields.
- If artist, album or genre tags are missing, tag reader will return these set to "Unknown".
- If no tags are available, the filename minus the extension is used for track name, and containing folder name for the artist.

Additionally, track numbers in xx/xx format are converted to a single integer, and artists that begin with 'The' are changed to 'Artist, The' to aid searches.

Unfortunately no techniques were found to enable Director to access tag data from WMA, MP4, OGG & AAC audio files and JPEG image files. The imbedded WMP, Real and QuickTime elements do not perform such a function.

3.5.2 Reading the file system

A method is needed to read the contents of a folders and subfolders, so a group of audio files can be added to the database. Director does not have the capability to do this, but a free Xtra called **FileXtra** has the capability.

This features many file-system functions, but of particular importance is the function to return the contents of a folder to an array. However, the Xtra is not recursive and does not return the contents of subfolders. This is much needed, so a recursive method, `folderRead` was written that facilitates reading of subfolders. This was achieved by the method calling itself whenever a folder contains a subfolder. Subfolders are distinguished by their trailing backslash. In addition to subfolder reading, `folderRead` will sort lists, or return files of specific types only if an array of extensions is passed.

In addition to the filename, a hash is needed so the database can tell if a file has changed or not. Initially the hash was provided by an additional hashing Xtra, `CaluMD5`. This was too computationally expensive, so was replaced with `FileXtras getModNo()`, which returns the date as a float.

Several useful folder and filename functions were written to compliment `FileXtra`.

Method	Function
<code>getExtension(fn)</code>	Returns characters following the period
<code>getNameNoExt(fn)</code>	Return the filename without the path, period and extension.
<code>getFileName(fn)</code>	Returns the filename & extension without the path.
<code>getPath(fn)</code>	Returns the path without the filename.
<code>getUpOneLevel(fn)</code>	Returns the path for the folder one level up. Detects UNC paths (\\) and prevents the computer/share name for being treated as a folder.
<code>getContainingFolder(fn)</code>	Returns the name of the deepest folder in the path

3.5.3 Network drives

While `FileXtra` can return the contents of a specified UNC share, it cannot return workgroups, node names, or share names for a given node from a SMB network. An Xtra that facilitates this was not found.

3.5.4 Database Xtra

Several 3rd party database Xtras were discovered for Director. Two examples are:

- MagicSoft XMySQL

This allows direct connection to a mySQL server without ODBC. The server can be local or remote, and started/shut down from within Director. Binary data can be inserted into the database too.

- Tabuleiro Arca Database Xtra

This is a customised version of SQLite; a widely used embedded database engine (e.g. PHP5). The Xtra is cross-platform, and the Projector itself acts as the client and the server, so there is no risk of being disconnected. The database can be viewed outside of Director using the free utility.

After some research, the Arca database was chosen because:

- No separate database server
- SQLite format can be examined with utilities.
- Integrated escape character handling between Lingo and SQL.
- Incorporates a createSelection() method to return a large group in fragments.

3.5.5 Implementing the database

The database design is defined inside Director, so if the DB file is damaged or lost, a new one can be created. After an initial test with a single-table database, the DB defined in the design section was set-up. The following code defines the database. The drop table/index statements ensure if a previous database is found, it is destroyed.

Defining the database using SQL

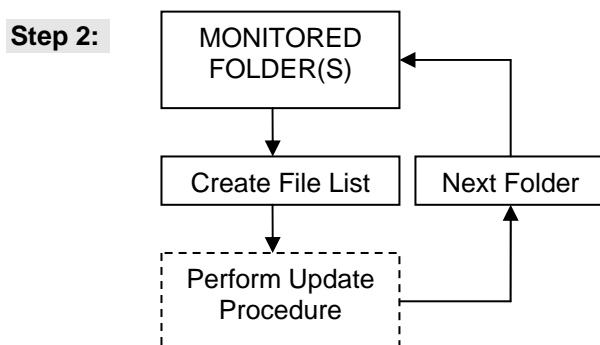
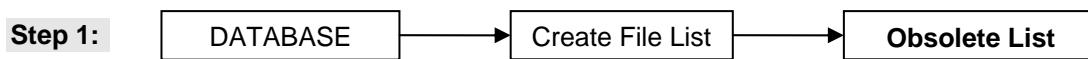
```
on resetDB(me)
  --delete tables/indexes
  gDB.executeSQL("DROP TABLE Artist")
  gDB.executeSQL("DROP TABLE Album")
  gDB.executeSQL("DROP TABLE Genre")
  gDB.executeSQL("DROP TABLE Path")
  gDB.executeSQL("DROP TABLE Song")
  gDB.executeSQL("DROP INDEX indexArtist")
  gDB.executeSQL("DROP INDEX indexAlbum")
  gDB.executeSQL("DROP INDEX indexGenre")
  --make them again
  gDB.executeSQL("CREATE TABLE Artist(ArtistID integer primary key, ArtistName text)")
  gDB.executeSQL("CREATE TABLE Album(AlbumID integer primary key, AlbumName text)")
  gDB.executeSQL("CREATE TABLE Genre(GenreID integer primary key, GenreName text)")
  gDB.executeSQL("CREATE TABLE Path(PathID integer primary key, PathStr text)")
  gDB.executeSQL("CREATE TABLE Song(SongID integer primary key, ArtistID integer, AlbumID integer, \
  GenreID integer, PathID integer, Filename text, ModNo numeric, Title text, TrackNo integer)")
  gDB.executeSQL("CREATE INDEX indexArtist ON Artist(ArtistName ASC)")
  gDB.executeSQL("CREATE INDEX indexAlbum ON Album(AlbumName ASC)")
  gDB.executeSQL("CREATE INDEX indexGenre ON Genre(GenreName ASC)")
  gDB.executeSQL("COMMIT")
end
```

SQLite automatically increments the 'integer primary key' field of each table so AUTOINCREMENT is not required. Three indexes have been created for the tables which are frequently accessed in alphabetical order.

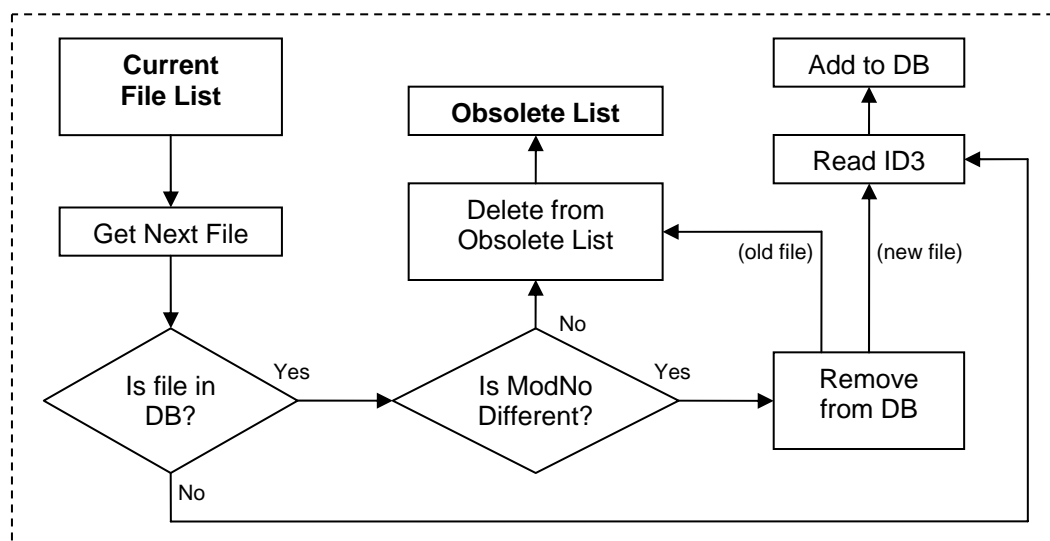
Records are added to the database using the method `addTrack()`. Before the record is written to the song table, the method calls `getID()` to obtain IDs for the artist, album, genre and path. For example, `getID()` checks the Artist table for the specified artist. If the artist is found, the ID is returned. If not, the artist is created and the ID is returned.

After initialising and testing the DB, the updateDB method was written to add media to the database. Creating a large media database from scratch is very time consuming, so the update function performs these steps:

- Files in the database are checked to ensure they exist.
- Any new files in the monitored are added to the database.
- Files in the database that have changed on the disk/network are updated.



Step 3: Update Procedure



* If the update process is cancelled, step 4 should not be performed as the obsolete list will contain files that should not be removed from the DB.

3.5.6 Configuration

Configuration information is written to disk using an Xtra called PropSave. This allows Lingo property lists to be saved and loaded from disk. PropSave is used in place of FileIO as there are several problems associated with FileIO, including the handling of null, void and quotes.

Configuration is set-up to check the configuration file. If corrupt or missing, a new file is created with the default settings as defined inside the media centre.

3.5.7 M3U reader/writer

The M3U reader/writer utilises the FileIO Xtra to read and write to text M3U files. The reader is more complex, as it must determine the type of M3U file, and detect the return characters used. The carriage return character is defined as RETURN, and the line feed used by UNIX systems is defined as numToChar(10). The writer creates the extended type M3U files with absolute paths.

3.5.8 Additional info methods

To help gather extended information about a media files, two separate methods are set up.

txtReader: Checks for text files in the same folder as the media file, and returns the contents of each file in an array. If a text file matches the media file name (excluding extension) it is returned as the first array element.

artReader: artReader checks for album art in the same folder as the media file, and uses special keywords to classify each image found as either Front, Back, CD, Inlay or Other. The method successfully detects the largest image created by Windows Media Player.

3.6 MVC: Controller

The controller discusses general navigation, plus the discussion of two interesting problems during controller implementation.

3.6.1 General navigation

Each frame is given a label, which is used to advance to the frame when desired. Variables cannot be passed to frames easily, so methods are used to set global variables before the movie advances to a frame.

An array 'backHistory' is generated on start-up that lists all the frame labels. This is used to store the previous section for a given frame, to facilitate moving back to the previous section of the media centre.

gotoNowPlaying() is passed the frame that called it.

```
on gotoNowPlaying(history)
  backHistory["sound"] = history
  _movie.go(_movie.label("sound"))
end
```

This method is assigned to the back key to backtrack though the media centre if possible.

```
if backHistory[the frameLabel] <> "" then _movie.go(_movie.label(backHistory[the frameLabel]))
```

3.6.2 *Media playback*

Director contains a Sound object that plays **WAV** (not ADPCM however) and **MP3** files. Three additional media elements are included to handle video, and additional audio formats: **Windows Media**, **Real Networks** and **QuickTime**. All elements require the player to be installed on the host OS. For cross-platform compatibility, all players bar Windows Media are available on PC & Mac platforms.

A separate Director project was set-up to test the players. Media files are set dynamically by changing the filename property of each player's cast member:

```
member("castname").filename = "C:\film.avi"
```

This is equivalent to importing the media element as linked, and the last played video file remains in the cast after the movie is stopped. The filename property cannot be cleared, so as a workaround each player's cast member is set to a blank video file when the movie starts and stops. This occurs with audio and image files too, so the same method is used to clear these cast members, as several high-resolution images will increase the cast size considerably.

Two copies of each media element (Windows, Real, QuickTime) were used; one for audio and video. Since the software is designed for playing music while browsing the centre, the audio sprites were placed off-screen, with the video option disabled, and set to be present on every frame. Each video element was placed on its own frame.

Each media element was tested with different media types to fathom its capability. The audio and video play methods were set-up to check media file's extension, and to choose the most appropriate player. When using a Mac, the Windows Media must be avoided. Although this means WMA audio cannot be played on the Mac platform, the Real media element can be used to play AVI files.

When initialising video files with different resolutions, Director continually changed size and position of the player's video window on the stage. A method was introduced to detect the aspect ratio of each video, and resize/reposition the window to occupy the whole stage for each video. A similar approach was used to fit the images to the screen in the slide show.

Automatically scaling video files to split the screen.

```

on setVideo()
    currentVideoPlayMode = 2
    if currentVideoPlayer <> "real" then
        member(currentVideoPlayer&"Video").fileName = currentVideoFile
    end if
    vidWidth = member(currentVideoPlayer&"Video").width + 0.0 -- required so not integers
    vidHeight = member(currentVideoPlayer&"Video").height + 0.0
    vidRatio = (vidHeight / vidWidth)
    if (vidRatio <= (600.0 / 800.0)) then
        _movie.sprite["videoInst"].width = 800
        _movie.sprite["videoInst"].height = 800 * vidRatio
    else
        vidRatio = (vidWidth/vidHeight)
        _movie.sprite["videoInst"].height = 600
        _movie.sprite["videoInst"].width = 600 * vidRatio
    end if
    --osd set-up
    osd = false
    osdVideoChange = false
    showOSD()
    if currentVideoPlayer = "real" then
        _movie.sprite["videoInst"].play() --real videos dont automatically play
    end if
end

```

3.6.3 OSD implementation and overlay issues

As with most PC media players, video playback is rendered as an overlay in Director. Consequently it is not possible for objects to appear on-top of the video. To enable an OSD, the video window is resized disproportionately to produce horizontal spaces above and below the picture. However, if a widescreen video is playing, the video will not be resized if there is already sufficient space for the OSD. The OSD will display automatically if a button is pressed, but will always disappear after 5 seconds of video playback.

A bug causes QuickTime videos to flicker if placed above or below a flash sprite on the stage. To correct this, all flash sprites besides the OSD were removed from the video stage. This causes some Flash sprites to not be initialised when the movie moves to a frame that references a Flash sprite immediately (on enterFrame). A further fix uses two frames with scripts to redraw the Flash sprites before a script attempts to call their methods. This is a good example of how Director makes a poor platform for a media centre.

3.6.4 Media control

To further complicate media playback, some media elements have different methods to control playback. The table illustrates the methods/properties for each element:

Media Element	Function	Method
Director Sound	Play: Seek: Pause: Stop: Duration: Current Time:	playFile(filename) currentTime = ms pause() stop() endTime currentTime (returns ms)
Windows Media	Play: Seek: Pause: Stop: Duration: Current Time:	play(filename) currentTime = ms pause() stop() duration currentTime (returns ms)
Real Networks	Play: Seek: Pause: Stop: Duration: Current Time:	play(filename) currentTime = ms pause() stop() duration currentTime (returns ms)
QuickTime	Play: Seek: Pause: Stop: Duration: Current Time:	filename = filename movieRate = 1 currentTime = ms movieRate = 0 movieRate = 0 currentTime = 0 duration currentTime *(returns ticks)

* QuickTime returns the current position in ticks, but setting the current position is done with milliseconds. A method to convert ticks (60th/sec) to ms is used when retrieving the current time for QuickTime.

The video and audio scripts use variables to store the current media player and file, to ensure the correct functions are called for each player. playManager is also responsible for moving to frame contains the appropriate video element.

3.6.5 DVD playback

DVD playback is implemented by adding a DVD media element to the stage, and by calling its functions. Initially, basic DVD playback was achieved from using a test file. However, an error occurs when the DVD module was added to the media centre cast. Other Director users have encountered this error, but a solution is not known.

3.6.6 Predictive text system

The predictive text system is the other specific feature that makes this software different to existing products. The system works by converting a string of numbers into an SQL statement. A look-up table is used to convert each number into a statement representing the letters associated. For example, the numbers '23' would convert into:

```
SELECT ArtistName FROM Artist WHERE
    (ArtistName LIKE 'a%' OR
     ArtistName LIKE 'b%' OR
     ArtistName LIKE 'c%'
     ArtistName LIKE '2%')
AND
    (ArtistName LIKE '_d%' OR
     ArtistName LIKE '_e%' OR
     ArtistName LIKE '_f%' OR
     ArtistName LIKE '_3%')
```

The underscores are important as they represent how many characters have been pressed. The following considerations were important during implementation:

- A button for punctuation is required to type artists such as R.E.M. A wildcard key was found the most practical solution.
- Special characters should be included. For example, the 3 key represents D, E F, 3 and è, é, ê, ë.

3.6.7 Dictionary system

The current predictive system does use a dictionary, it simply uses an SQL statement to match results the begin with certain letters.

Consider the following. A search for 'CUT CHEMIST' under artists will yield no results no artists start with these letters. However 'Cut Chemist' can be found in artists:

'DJ SHADOW AND CUT CHEMIST' can only be returned using predictive if
'?????????????CUT CHEMIST' is entered. The number of wildcards is important.

A dictionary could be built from all fields in the database. Words should be divided up, such as OCEAN and COLOUR and SCENE would be inserted separately for the band OCEAN COLOUR SCENE.

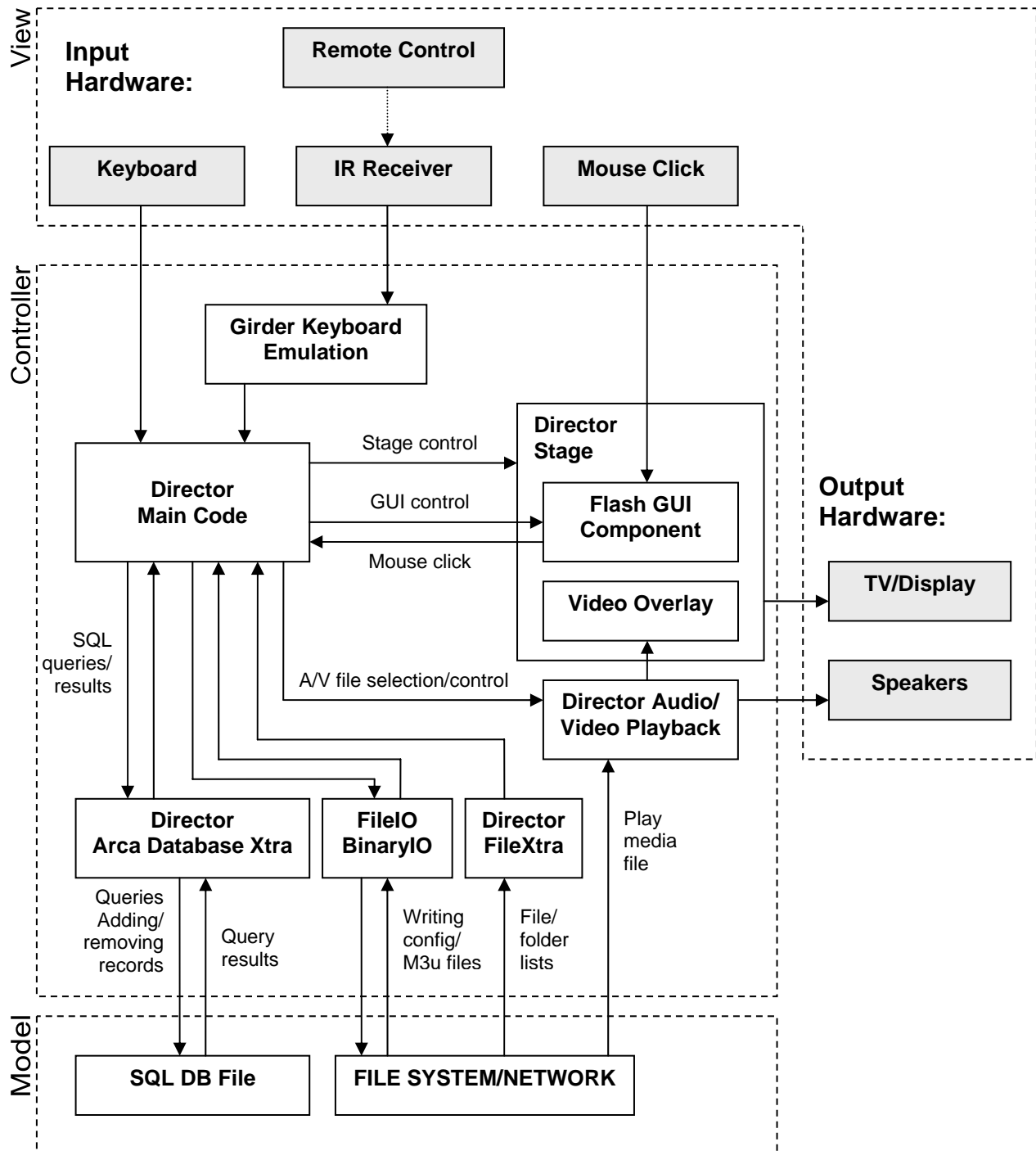
The predictive would use the dictionary to determine the word(s) typed, and perform a string SQL search on the database. This not only facilitate searching of all fields, but results would be returned where there is a match in the middle.

The only downside with the extended dictionary is matches may be less precise, increased DB creation time and slower search performance. The following example dictionary table could be used to facilitate searches by any combination of artist, album, genre and track:

WordID	inArtist	inAlbum	inGenre	inTrack	WordString
1	1	0	0	0	Adema
2	1	1	1	0	Soundtrack
3	1	1	0	1	Music
4	0	0	1	0	Electronica

3.7 Implementation overview

The following diagram shows an overview of the implementation.



4.0 Testing

4.1 White box

White box testing was performed throughout implementation, to ensure each component is fully operational before continuing with development. During testing, example media files were set-up to test different parts of the system.

4.1.1 ID3 reader

The ID3 reader was tested by creating several MP3 files with specific tags:

- double_trackno.mp3 (track number in xx/xx format)
- filename_bothtags.mp3
- filename_notag.mp3
- filename_nov1tag.mp3
- filename_nov2tag.mp3
- maxinfo.mp3 (all extended ID3 fields completed)
- speech_marks_in_fields.mp3
- filename.with.dots.in.it.mp3 (for folder read testing)

4.1.2 File/folder reading

The folder reader was tested with 13,000 files and a property list is returned in a reasonable amount of time. A file named “filename.with.dots.in.it.mp3” checked the method for returning the filename without the extension.

4.1.3 AlbumArt reading

A set of album art images with typical filenames were taken to test the album art reader.

4.1.4 Media playback

One of the most important features of the media centre is its ability to play the most popular types of media. A set of sample audio formats were made using Easy CD-DA Extractor:

- WMA, WAV, AAC, AIFF, M4A, MP3, MP4, OGG, RAM, WAV

These were used initially to check each media elements capability. Once the capabilities were understood, the media files were used again to check the playAudioFile() method was selecting the correct player for each file

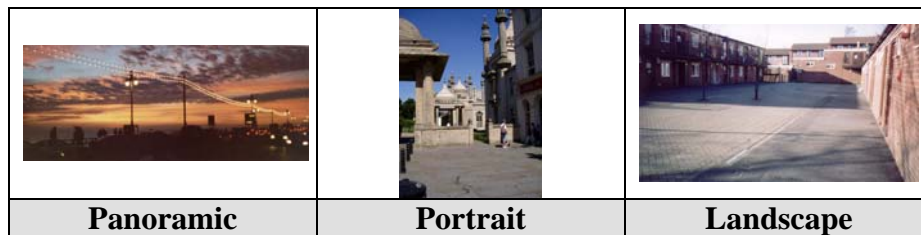
The same method was used for video files:

- AVI, MOV, MPEG, RAM, WMV

RealAudio and RealVideo formats often have the same file extension (RAM or RM), so the media centre is unable to tell the difference between Real audio and video files.

4.1.5 Image viewing

A panoramic, portrait and landscape image were used to test the slide show to ensure the images are resized correctly and maintain their aspect ratio.



4.1.6 Media database

The media database was tested with a real-world MP3 collection of 17,000 tracks (1300 albums) with fully organised ID3 information.

4.2 Black box

Black box testing was performed when the media centre was near completion. The two types of black box testing were:

- Cross-platform testing
- User acceptance testing

4.2.1 Cross-platform testing

Unfortunately I Macintosh PC was not available for full Mac testing. It should be realised however that methods for working with folders and filenames will need to be reworked as a Macintosh computers use a colon to separate folders, whereas PC's use a backslash.

4.2.2 User acceptance testing

To test user acceptance, the system was set-up in its target environment, in a living room connected to a TV. Users were asked to perform tasks using the remote, and their comments were recorded.

In addition to each users comments, the software was monitored for new bugs that appear when other users try the software.

User:	Nick Atkins, student	Age / sex:	24 / male
Positive Feedback		Improvements/Suggestions	
<ul style="list-style-type: none"> The remote is well designed and easy to use. Media guide key is useful to return to the main menu from anywhere. Predictive text function is fantastic for finding songs quickly. 		<ul style="list-style-type: none"> If the song playing is deleted from the playlist, it continues to play which is misleading. Feedback of items being successfully queued should be displayed, such as a small thumbs-up in the screen corner. Number pad should allow searching in long file browser lists The OK button should select files in the file browser or songs in the media library 	

User:	Gale Hubbard, student	Age / sex:	21 / female
Positive Feedback		Improvements/ Suggestions	
<ul style="list-style-type: none"> The clock on the main menu is useful The information box and display in the video browser is very useful 		<ul style="list-style-type: none"> The red 'Edit Playlist' button changes to delete abruptly when pressed so the selected song may be deleted. Cannot back-up to main menu from the file browser or media guide 	

User:	Robert Morrissey, student	Age / sex:	23 / male
Positive Feedback		Improvements/ Suggestions	
<ul style="list-style-type: none"> A warning dialog prevents accidental clearing of the current playlist, which is a major problem with players such as Winamp The mini folder tree prevents you from getting lost in the file browser The GUI is very crisp and pleasing on the eye. 		<ul style="list-style-type: none"> The items highlighted in yellow cannot be read easily Long filenames should scroll instead of being truncated The predictive window should be wider when space is available. Information about the current song should be displayed in the screen corner in the browsers and main menu 	

The overall acceptance by users was very pleasing. Users liked the GUI design, and operating technique. Most were able to perform the tasks with little help, however on occasion the media centre did not behave as they expected. Although more improvements and suggestions were received than positive comments, most were small changes that could easily be implemented. However, even the small abnormalities in software can prevent it from being enjoyable to use.

5.0 Maintenance

When developing a project of this scale, care must be taken to ensure the development is well documented to aid the software's life cycle. The end user will expect the software to be maintained by releasing bug fixes and new features. Documentation should be sufficient to allow another programmer to continue maintaining and developing the software without requiring the author's assistance. Otherwise, the lack of understanding may cause the software to be retired.

The following practices have been in-place to aid the software life cycle:

- Code is commented, plus notes and descriptions of scripts and methods are recorded in the commenting.
- A dated activity log contains notes on all research and experimentation performed, plus any changes made or features implemented to the project. The log is provided in the appendix.
- A dated log book contains rough notes made during development.
- A copy of the project is saved when objectives are reached or a potential problem is discovered. The changes or reason for the save is documented in a Project Changes text file.
- Testing and experimental projects are retained for future reference.

6.0 Conclusion

The overall success of the project is best evaluated by reviewing the specifications and requirements set out in Background Studies. Although a majority of the specifications were implemented successfully, some ambitious specifications were omitted due to time limitations.

GUI implementation was very successful. The components designed in Flash were perfectly suited for display on a television, and provided the necessary functionality. The fast-text menus are practical for controlling multiple aspects of the media centre. However, although the colour of all Flash items is set dynamically, no interface was provided for changing the appearance.

Full remote control support was entirely successful, as the keyboard and mouse are not required for any part of the media centre. The selected IR device was low-cost, and provides compatibility with most existing remotes.

The media centre is capable of playing all popular media formats, including DivX and XVID. The Now Playing interface provides a detailed display of the current track, including the detection of AlbumArt. Auto-saving and recalling of playlists from previous days was not implemented. While fully-working M3U import and export methods were completed, sadly no interface was provided to allow the user read and write playlist files.

The SQLite database, media library and predictive search function provide a very powerful but easy to use interface for locating MP3 files. The library combined browsing and searching functions seamlessly, allowing the user to search, and further refine the results. Incidentally, non-predictive text was not implemented as the predictive mode was so successful.

Director's suitability for implementing media-centre type applications is questionable. Although media playback of many formats was achieved easily, many workarounds are required to ensure Director loads each file successfully. Incidentally, the software uses 100% CPU usage at all times, which challenges the low-noise cooling systems of media centre PC's. The performance of background applications will also suffer. If a similar project were to be undertaken again, an alternative development platform may be considered.

MAC Compatibility

6.1 Future additions

- **TV tuner support:** Director supports capture devices, which would facilitate TV viewing inside the media centre.
- **Internet streaming audio and video:** The media elements built into Director support streaming media, such as Internet TV and Radio stations.
- **3D visualisations using Shockwave 3D:** A replacement for 2D Flash visualisations.
- **Online news and weather reports:** Many other media centre applications support weather reports, which can be obtained from servers on the Internet.
- **Email composition using predictive text:** The existing predictive SQL could be coupled with a dictionary for writing emails using the remote.

Bibliography/References

Sources of information used in the report.

Books:

Macromedia Director MX 2004 Bible – Riley
 Macromedia Director MX and Lingo. - Phil Gross
 Macromedia Flash MX 2004 Bible - Riley
 Java Database Best Practices – O'Reilly
 Software Engineering – 6th Edition – Ian Sommerville
 Interaction Design (Beyond HCI) – John Wiley & Sons – ISBN 0-471-49278-7

Java Media APIs: Cross-platform Imaging, Media and Visualization - Alex Terrazas
 Java Database Programming – Brian Jepson
 The Complete Guide to Java Database Programming – Matthew Siple

Other resources:

Video Production Techniques – Editing and Post Production Handout – Dr. Phil Watten
 Technical Communication Course Notes – Dr. H. Prance

Internet:

Author:	Macromedia	Title:	Macromedia Home Page
URL:	http://www.macromedia.com		

Author:	Tabuleiro	Title:	ARCA Database Xtra
URL:	http://xtras.tabuleiro.com/products/arca/index.tdb		

Author:	MagicSoft	Title:	MagicSoft Xtras
URL:	http://www.xtra-ucd.com/		

Author:	Pimz	Title:	PropSave Xtra
URL:	http://www.pimz.com/		

Author:	MediaMacros	Title:	Useful Lingo String Functions
URL:	http://www.mediamacros.com/		

Author:	Steve Grosvenor	Title:	Integrate Flash/Director MX 2004
URL:	http://www.sitepoint.com/print/flash-director-mx-2004		

Author:	SQLite	Title:	SQLite Home Page
URL:	http://www.sqlite.org/		

Author:	Xtrasy	Title:	Xtrasy Home Page
URL:	http://www.xtrasy.com/		

Author:	(unknown)	Title:	MP3 ID3 Tag Reader Script
URL:	http://staff.dasdeck.de/valentin/lingo/mp3_swa/		

Author:	Kent Kersten	Title:	FileXtra Home Page
URL:	http://www.kblab.net/xtras/index.html		

Author:	Calu	Title:	Macromedia Director MD5 Hashing Xtra
URL:	http://xtras.calu.us/xtrasMD5.php		

Author:	Igor Cesko	Title:	Infrared Devices
URL:	http://www.cesko.host.sk/hardware.php		

Author:	Proximis	Title:	Girder Home Page
URL:	http://www.promixis.com/		

Author:	Keyspan	Title:	Keyspan Peripherals Home Page
URL:	http://keyspan.com		

Author:	BCS	Title:	British Computer Society Code of Conduct
URL:	http://www.bcs.org		

Author:	Creative Technology	Title:	Creative Zen Portable Media Centre
URL:	http://www.creative.com/products/		

Author:	Microsoft	Title:	Windows XP MCE 2005
URL:	http://www.microsoft.com/windowsxp/mediacenter		

Author:	CNET	Title:	Windows XP MCE 2005 Review
URL:	http://reviews.cnet.com/4520-3672_7-5536454.html		

Author:	Meedio	Title:	Media Centre Software
URL:	http://www.meedio.com/		

Author:	Intervideo	Title:	Home Theatre Home Page
URL:	http://www.intervideo.com/jsp/HomeTheater_Profile.jsp		

Author:	I. Baker	Title:	Safe Areas for Widescreen Transmission
URL:	http://www.ebu.ch/trev_280-baker.pdf		

Author:	(unknown)	Title:	PowerPoint on TV
URL:	http://www.soniacoleman.com/FAQs/FAQ00143.htm		

Author:	(unknown)	Title:	The M3U (.m3u) Playlist File Format
URL:	http://hanna.pyxidis.org/tech/m3u.html		

Author:	EETD	Title:	Standby image
URL:	http://eetd.lbl.gov/Controls/pics2/standby.gif		

Author:	C Board	Title:	How to test whether integer is even or odd
URL:	http://cboard.cprogramming.com/		

Author:	Unknown	Title:	The Software Life Cycle
URL:	http://www.cs.wm.edu/~coppit/csci435-spring2004/lectures/4-lifecycle.pdf		

Author:	Peter Joel	Title:	Flash Dynamic Gradient Fill
URL:	http://www.peterjoel.com/Samples/		

Author:	Glenn E. Krasner/ Stephen T. Pope	Title:	A Description of the MVC User Interface Paradigm in the Smalltalk-80 System
URL:	http://www.ccmrc.ucsb.edu/~stp/PostScript/mvc.pdf		

Java-based:

Author:	Sun Microsystems	Title:	Custom Player GUI
URL:	http://java.sun.com/products/java-media/jmf/2.1.1/solutions/CustomGUI.html		

Author:	JavaWorld	Title:	Java Media Framework Player API
URL:	http://www.javaworld.com/javaworld/jw-04-1997/jw-04-jmf.html		

Author:	JavaWorld	Title:	Program multimedia with JMF, Part 1
URL:	http://www.javaworld.com/javaworld/jw-04-2001/jw-0406-jmf1.html		

Acknowledgements: