Who Needs a TV and a VCR?

SoftVCR for Windows

Providing a Software Implementation for Consumer Video Recording

Myles MacRae
mgm1@doc.ic.ac.uk

Supervisor: Ian Harries
2nd Marker: Iain Phillips

21st June 2000

Abstract

Digital Video Recording is a relatively new, but growing application on home computers. The last couple of years has seen the level of technology on an average desktop grow to the point where video can now be recorded and viewed by anyone using low priced hardware. Traditional video applications have been aimed at the amateur video editor rather than those who simply want to record programmes for later viewing. As people get used to using their computers for viewing video and television, the need is there for providing applications that can mimic, and build on, the functionality of their home VCR.

This project presents a software implementation of a video recording application designed to replace a home VCR. It provides a means of recording programmes directly or via timers with facilities for managing recordings. With many home computers also gaining permanent network connections, this project also examines how networked information sources, such as TV Listings Websites, can be used to further increase the ease of recording and managing programmes. As a result, this report also presents an integrated Electronic Programme Guide which is able to extract data from listings websites. The parser can be configured for different websites without recompiling the application via a parser description language created for the project.

Preface

The text of this report, together with the source code of SoftVCR (as a Visual C++ 6.0 project) can be found at:

`http://www.doc.ic.ac.uk/~mgm1/`

Acknowledgements

I would like to express my thanks to my supervisor, Ian Harries, for his guidance and advice during the project.

Thanks must also go to Annmarie Fernando, Phil Rowlands, Mark Rossiter, Jamie Galloway, Simon Buckle and Marcus Pickering for their help with evaluating this project.

Finally, thanks to the numerous members of Computing/ISE 4th year for their invaluable advice and support.

Contents

1 Introduction **1**

1.1 The Move To Digital 1

1.2 Project Objectives 2

1.3 Report Structure 3

2 Background **5**

2.1 Video Capture Cards 5

2.1.1 Hauppauge WinTV PCI 5

2.1.2 ATI All-In-Wonder [4] 6

2.1.3 Matrox Marvel G400 TV [5] 7

2.1.4 Pinnacle Systems Studio Family [6] 7

2.2 Current Software VCR Applications 8

2.2.1 Cinax WinVCR [7] 8

2.2.2 MGI Pure DIVA [9] 9

2.3 TV Listings Sources 10

2.3.1 DigiGuide [10] 10

2.3.2 Website Listings 11

2.4 Compression Technologies 13

2.4.1 Intel Indeo Software [13] 13

2.4.2 MPEG [15] 13

2.4.3 Streaming Media 17

2.5 Programming Technologies 18

2.5.1 Quicktime for Windows 18
2.5.2 Video For Windows 19
2.5.3 DirectX and DirectShow 19

3 Design Overview **21**

3.1 Suitability of VFW and DirectShow 22
3.1.1 Video For Windows Evaluation 22
3.1.2 DirectShow 22
3.1.3 Conclusion 24
3.2 Results of Compression Algorithm Testing 24
3.2.1 Intel Indeo 25
3.2.2 PICVideo Motion JPEG 25
3.2.3 Windows Media & Real System 26
3.2.4 Conclusion 26
3.3 Software Functionality Overview 27
3.3.1 Timer 27
3.3.2 Electronic Programme Guide 28
3.3.3 Video Library 32

4 A Software Implementation of SoftVCR **35**

4.1 WebParser -
A Prototype Application for Parsing Listing Websites 35
4.2 Video Recording - Extending VidCap 38
4.2.1 Architecture 39
4.2.2 Adding Support for Hauppauge 40
4.2.3 Interface Changes 41
4.3 Video Playback 42
4.4 Timer 43
4.4.1 Object Persistence 44
4.4.2 Serializing the timer 44

4.4.3 The Timer Dialog 45
4.4.4 Adding a Timer 45
4.4.5 Running a Timer 47
4.5 Video Library - Cataloguing and Managing Video Clips 47
4.5.1 The Library List 48
4.5.2 Showing Statistics 48
4.5.3 Recompressing Videos 49
4.6 Electronic Programme Guide 50
4.6.1 Getting Listings 51
4.6.2 Showing a Programme Description 53
4.7 Parsing HTML Television Listings 53
4.7.1 Listing Parser Format 54
4.7.2 Describing a URL 55
4.7.3 Parsing the Page 55
4.8 Case Study: Cable Guide 57
4.9 User Evaluation 58

5 Conclusion and Future Work 63
5.1 Commercial Potential 64
5.2 Extensions and Future Work 65

A User Guide 67

B Listing Parser Format (LPF) 73

List of Figures

2.1 Screenshot from Cinax WinVCR 8

2.2 Programming the scheduler in MGI Pure DIVA 9

2.3 Planner View in DigiGuide 10

2.4 General Reference Model of MPEG-1 [19] 15

2.5 MPEG-2 Video Profiles [19] 16

3.1 An Example DirectShow Capture Filter Graph - shown in the DirectShow utility, GraphEdit 23

3.2 Screenshots from TV Listing Websites 31

4.1 Screenshot of WebParser 36

4.2 Screenshot from VidCap 38

4.3 SoftVCR interface 41

4.4 Video Player Screenshot 43

4.5 The Timer Dialog 45

4.6 Adding a new timer in SoftVCR 46

4.7 The Video Library 48

4.8 Recompression Dialog in SoftVCR 49

4.9 The Electronic Programme Guide 51

4.10 Get Listing Dialog 52

4.11 Parser Setup Dialog 52

4.12 Showing a Program Description 53

4.13 State Diagram for Cable Guide 58

Chapter 1

Introduction

Home computing is currently undergoing a paradigm shift. Driven by the rapid introduction of powerful hardware and low prices, it is now possible to perform tasks on home computers that until now have been reserved for the professional domain. This project deals with one such domain: Digital Video Recording. In this market led industry, the technologies used by traditional broadcast networks and Computing networks has started to converge. This has lead to a blurring of the distinction between traditional media appliances such as television or radio and IT appliances such as computers.

1.1 The Move To Digital

The next few years will see the majority of the worlds broadcast networks migrate to an increasingly digital platform. The savings on bandwidth and the promised increase in quality of service are driving broadcasters towards a system that is free from the vertical integration previously enjoyed. This system is known in Europe as the Multimedia Home Platform[18] and incorporates the management and viewing of multimedia from any source, using as much of the same decoding equipment as possible.

For instance, it won't matter which delivery method, such as satellite or cable, you choose to subscribe to, the content and quality will be identical. Also, subscribers will not need a separate computer and television, as the technologies converge. This is already becoming the case with many new generation set top boxes allowing subscribers to access the Internet and send email, while computers allow subscribers to watch television and video discs.

This project concentrates on one of these areas of convergence, consumer video recording. This area of technology has traditionally been dominated by analogue VHS video tape for delivering both pre-recorded media and home recording. Digital Versatile Disc (DVD) is now establishing itself as the best delivery method for pre-recorded media, but VHS still dominates the home recording market. Many see the problem as being one of copy protection, with broadcasters and content providers fearing that digital recording systems would mean unlimited, flawless copying of their products. However, with the prevalence of large hard drives and fast computers, there are still very few computer implementations of video recorders that can provide the same type of quality and functionality as would be found on a home VHS recorder.

1.2 Project Objectives

This project addresses this issue of video recording on a suitably equipped Windows PC. There are two objectives with regards to this project:

- To explore the area of desktop and domestic digital video. In particular, to discover the technologies that would be available to developers to create video applications and the technologies that will be important in the next few years. Also, to explore the possibility of incorporating other sources of information such as television listings websites, to help improve the usability and management of video.
- To produce a software implementation of a video recording application designed to replace the home VCR.

The features of the software implementation are not to be restricted to those found on a VCR. As mentioned in the first aim, there is a wealth of information to be found on the Internet or in the program streams themselves that can be used to set timers, provide descriptions and generally make the experience more interactive. The functionality of the video recorder can be broken down into the following areas:

- *Software Video Recorder* - The basic functionality of recording and playing back video. This will need to include consideration on issues of compression, as well as the transport controls users will be used.
- *Recording Timer* - As with all good VCRs, a facility for setting recording timers for unattended recordings should be provided. This would need to

be able to be set either directly, or from some external information such as television listings. A timer should also check that there will be enough available storage to record the full clip, and provide detection of conflicts with other programmes.

- *Video Clip Library* - The library would provide a high level overview of the clips recorded by the recorder; including such information as programme name, original start time, programme duration and channel. The library must also provide a number of management functions so that clips can be deleted, renamed or even archived using another compressor.
- *Electronic Programme Guide* - An electronic programme guide (EPG) should provide an interface for gathering and reviewing television listings. These listings will be used to set timers for programmes.
- *Listing Parser* - Although an integral part of the EPG, the listing parser should be capable of extracting programme information from a named website. Additionally, this parser should be reconfigurable to handle different websites without the need to recompile the application.

TARGET ENVIRONMENT: The Department of Computing possesses a number of Hauppauge WinTV PCI TV tuner and video capture cards [1]. As a result, the implementation will be based around this card and its functionality. The platform will be Windows NT 4.0. The user will be considered to be a computer literate individual with knowledge of standard Windows user interface features.

1.3 Report Structure

The remainder of this report will divided as follows:

Chapter 2 - provides background into the area of Digital Video Recording and PC Multimedia Programming.

Chapter 3 - gives an overview of the design procedure and decisions made towards producing a final implementation.

Chapter 4 - describes the implementation of SoftVCR for Windows, with an evaluation of its use by external users.

Chapter 5 - gives a conclusion to the project, outlining the main achievements and discussing the further work that could be done.

Chapter 2

Background

This chapter discusses many of the technologies in use today with regards to consumer digital video. It begins by giving an overview of the Hauppauge WinTV capture card used in the project, along with some of the competitors.

2.1 Video Capture Cards

The implementation is based around the popular family of Hauppauge WinTV video capture cards. These boards incorporate on-board TV Tuners and video capture hardware into an affordable package. The reason for their selection is due to their availability within the Department of Computing, however it should be noted that other cards are becoming available as the demand for integrated video solutions increases. We begin by looking in detail at the Hauppauge card.

2.1.1 Hauppauge WinTV PCI

Hauppauge manufacture a range of multimedia cards based around the Brooktree BT848 PCI Video Capture device[1][3]. Their cards, in general, integrate a region specific TV tuner, with a high quality video capture device. This allows the user to view live television and video on a standard desktop computer. The cards contain a number of useful features:

- Teletext: All the boards feature the ability to decode Teletext information for use in any application.
- PCI Bus Master: The board operates as a PCI Bus Master, therefore providing the ability to perform basic capture and video display without

the need for CPU resources. The data is simply transferred directly across the PCI bus to the relevant device.

- FULL FRAME CAPTURE: The device can capture video at up to full PAL resolution (758x576) at 25 frames per second. Full performance is dependent on the speed of the host computer but frame size can be scaled in hardware.
- COLOUR FORMAT: A number of colour formats can be provided directly by the hardware including a extensive selection of RGB and YUV formats.
- DIRECTX AND VFW: Drivers operate through fully compliant DirectX and Video For Windows (VFW) interfaces, allowing integration into any external application. Control of the tuner can be achieved through the Hauppauge SDK.
- AUDIO: Audio is available via dedicated outputs on the board. These can be connected to the host computers sound card for inclusion with the captured video.

Hauppauge estimate that they have an installed user base of 1 million users [1], making it the most popular retail TV card range currently available.

The software provided with the cards allows provides video capture capabilities, but does not support any type of compression. The result is that captured files can be extremely large and unsuitable for a consumer video application.

For external developers, Hauppauge provide a Software Development Kit (SDK) to access features not available through Video For Windows or DirectShow. In particular, features such as Tuner Control and Image Capture can be added any applications made in C++, Visual Basic or even HTML.

2.1.2 ATI All-In-Wonder [4]

The ATI All-In-Wonder video card is one of the first generation of a fully integrated desktop video solution. It combines the traditional functions of a VGA card with advanced TV tuner and video capture functions. Of particular interest is the ability to capture and record real-time MPEG-2 video, thus making a substantial saving on capture file size whilst maintaining a decent level of quality. Here is a summary of its capabilities:

- 128 bit 2D/3D graphics
- Video input
- Video output
- TV-Tuner
- Hardware DVD Playback
- Video Editing
- Real-time MPEG-2 Video Encoding

2.1.3 Matrox Marvel G400 TV [5]

This is another card that has established itself into the fully integrated multimedia card market. As with the ATI card, it combines a VGA card with a TV Tuner and video capture hardware. It also provides hardware video compression in the form of Motion-JPEG. This provides a high quality source of video, but at a higher bitrate than MPEG-2. The package is supplied with a software only MPEG-2 transcoder for converting its own files to MPEG-2, but this is provided at the expense of speed.

2.1.4 Pinnacle Systems Studio Family [6]

Pinnacle Systems manufacture a range of dedicated video capture cards aimed at differing segments of the market:

- Studio DC10 Plus: PCI M-JPEG capture device with composite and S-video inputs. Maximum resolution of 640x480.
- Studio 400: Parallel port low resolution Intel Indeo capture (160x120).
- Studio MP10: Parallel port MPEG-1 codec and video capture. Maximum resolution of 352x240.

These devices are primarily aimed at capturing short clips for creating presentations and sending emails.

Figure 2.1: Screenshot from Cinax WinVCR

2.2 Current Software VCR Applications

VCR applications are not a new idea. Many of the above hardware devices are provided with their own video capture application. These range from complete video editing programs to VCR programs that utilise the specific hardware available to them. There are, however, an increasing number of stand alone applications becoming available which offer digital VCR facilities.

2.2.1 Cinax WinVCR [7]

Released in August 1999, this product is an MPEG based software VCR. The MPEG codec is licensed from Ligos Technologies [8] and implements the MPEG-1 CPB standard allowing 30 frames per second recordings at 352x288. It has the following features:

- Utilises latest MPEG technology from Ligos.
- Works in Windows 95 OSR2 and Windows 98 (not NT compatible).
- Constructed on DirectX Media 6.0.
- Supports a number of TV Tuner boards.
- Real Time MPEG encoding requires a 300MHz Pentium or faster.

This product does not include any library functions for organising recorded clips. Nor does it utilise any Electronic Programme Guide for setting the timer.

Cinax have recently licensed WinVCR to Hauppauge for bundling with their new range of TV Tuner boards. By not using the more processor intensive MPEG-2

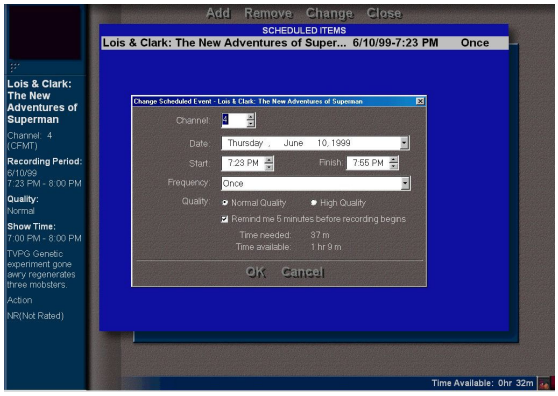


Figure 2.2: Programming the scheduler in MGI Pure DIVA

technology, WinVCR keeps the hardware requirements at a reasonable level so that a larger percentage of end users can utilise the product.

Personal testing of WinVCR on a Windows 98, Hauppauge WinTV machine revealed problems with the installation of the product. Installation corrupted the WinTV drivers resulting in the failure of Hauppauge software. The only remedy was a complete reinstallation of the WinTV drivers and removal of WinVCR. This problem was reported to Cinax technical support. As mentioned above, this program does not support Windows NT, so its use on the target lab platform is not possible.

2.2.2 MGI Pure DIVA [9]

This product utilises Intel Pentium III SIMD technology to deliver an MPEG-2 based video recorder. It was debuted at the E3 show in Los Angeles in May 1999 as a complete entertainment product featuring:

- TV Viewing
  - Time Shift technology allows a live program to be paused, rewound and replayed.
  - Pan and Zoom functions while watching a live program
- Video Recording
  - Video can be programmed via Internet updateable Electronic Program Guide
  - Bookmarks can be placed anywhere in video

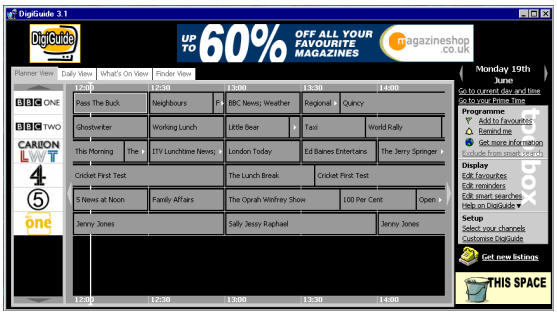


Figure 2.3: Planner View in DigiGuide

  - Library functions for organising and deleting recorded clips

- DVD Mode
  - Allows playback of DVD videos
- CD Mode
  - Allows playback of normal audio CDs

Such features require a very powerful desktop computer and the minimum specification is a 500MHz Pentium III. This product is only available through OEM licensing so the only way to obtain a copy is to buy a new computer that is bundled with the software. At the time of writing, only Dixons are distributing Pure DIVA with their own range of PCs in the UK. There are no demonstration versions available for evaluation.

The Electronic Program Guide(EPG) is updated via the Pure DIVA website using a proprietary format. The default recording resolution is 352x480 at 6 Mbps with 176kbps audio. This is used for both time shifting and recording. If the user wishes to record only, then a resolution of 720x480 can be achieved.

For future versions, MGI are looking into providing access to different EPGs. They also plan to add editing functions and support for other multimedia formats such as MP3.

2.3 TV Listings Sources

2.3.1 DigiGuide [10]

DigiGuide is a TV listings application that provides access to most UK terrestrial and satellite channels. At the time of writing, it includes details of 151

channels. It works by downloading proprietary data at regular intervals (once a week) on the channels you select. The data is displayed as a list for each channel showing the time, name and description, or as a daily planner comparing all your channels across a horizontal timeline (see Fig. 2.3).

DigiGuide is free to download and use; their financial revenue is generated via adverts displayed in the application and merchandising links on the website and application.

The application includes a number of features for making listing navigation easier and more useful:

- Reminders can be set to provide visual warnings on when programmes are about to start.
- The user can elect favourites so that users can see at a glance when a programme is on.
- Smart searches allow complex queries to be entered based on channel, category of programme and a search string.
- Clicking on "Get More Information" spawns a web browser to open a page in their www.tvlibrary.co.uk website providing more information about the programme as well as site and merchandising links.

2.3.2 Website Listings

A popular source of TV listings information is the World Wide Web itself. Many websites exist that provide comprehensive listings and information on a multitude of terrestrial, satellite and cable channels with all their regional variations. Many of these sites are run by traditional listings sources such as:

- Radio Times: http://www.radiotimes.beeb.com
- Cable Guide: http://www.cableguide.co.uk
- Teletext: http://www.teletext.co.uk

Structure

The common factor between these listings sites is the HTML language that they are written in. HTML is a tag based hierarchical language and describes how a page should be displayed in terms of headings, bodies, tables, frames,

etc. Listings in all of the above websites are displayed using tables. As tables are split into cells, this makes it relatively easy to isolate different types of information depending on the cell being looked at. For example, the following is an edited example of a line from the radio Times website:

```
<TR>
   <TD><IMG SRC="file.gif"></TD>
   <TD><FONT><SPAN>23:45</SPAN></FONT></TD>
   <TD><FONT><SPAN><B><A HREF="programme details">
   <IMG SRC="arrow.gif"> Every Home Should Have One</A></B>
   </SPAN></FONT></TD>
   <TD></TD>
   <TD></TD>
</TR>
```

Here, the second cell contains the start time of the programme. The third cell contains the name of the programme and also a link to display more information on the programme. The other mentioned websites are simple variations on this structure and also supply the descriptions for the programs in the listings instead of having to get another page.

A more detailed description will be given in the section on the web parser.

Getting The Right Information

These websites use server side programming to extract information from a database. A side product of this method is that the URL for each page contains the exact parameters to construct the necessary query. For example:

```
http://www.cableguide.co.uk/bread/go?
date=17/01/2000&time=03%A00&theme=&alias=8&type=list
```

This is the Cable Guide URL to get the listing for the next six hours from 3AM on BBC2 (alias 8) on 17th January 2000. However, the variation between site queries can be dramatic. For example, the listings at ananova.com are accessible via the following form.

```
http://www.ananova.com/tv/listings/carlton/day1/bbc1.html
```

Here, the 'carlton' indicates the television region, 'day1' describes the desired day relative to today (ie. today is day1, tomorrow is day2, etc.) and 'bbc1.html' gives the desired channel by name.

2.4 Compression Technologies

Many applications of digital video, from broadcast to desktop computing, have only become possible due to the emergence of compression techniques that reduce the size of the required bitrate to more domestic levels. To place this in perspective, a full rate PAL video signal would need in excess of 253 Mbps (assuming a 24bit colour resolution) not including sound. Broadcast systems, such as Digital Video Broadcasting[18], can compress this signal down to around 6 Mbps without any discernible loss in quality, with component quality at only 9 Mbps, using MPEG-2. Technologies designed for video delivery over the Internet can reduce the bitrate down to as little as 20 kbps and still provide a recognisable, moving image with sound. Described below are some of the more popular technologies for compressing video.

2.4.1 Intel Indeo Software [13]

Designed for the Video For Windows architecture. Indeo was originally developed to provide constant bitrate video compression for delivery via CD-ROM. It is optimised to take advantage of features only found in Intel's range of processors and is only found on Windows computers.

The latest version of Indeo (Version 5.11), is specifically designed to take advantage of MMX and Pentium II technology. Indeo 5.11 features a wavelet compression algorithm, which is intended to improve video quality over previous versions. It also support Progressive Download, a feature of DirectShow which is intended to provide scalable video at real time over networks.

Indeo provides two versions of its codec; the first is called a Quick Compressor. As the name suggests, it provides a means to compress a video stream quickly, mainly for prototyping and capturing. In tests, a Pentium II 350MHz can capture SIF (352x240) at 25fps in real time without problems using the Quick Compressor. The second is the standard codec. It produces higher quality, lower bitrate video but at the expense of processing speed.

The advantage of the Indeo compressor is that it is free to download from the Intel website and free to use in applications.

2.4.2 MPEG [15]

Unlike other compressors, MPEG does not only define the way that a video stream is compressed, it describes the entire framework for coding audio and

video using a toolkit. The Moving Pictures Expert Group (MPEG) was established in 1988 and is a working group formed by the joint efforts of the International Electrotechnical Commission (IEC) and the International Organisation for Standardisation (ISO). The joint committee formed (JTC1) was given the general title of "Information Technology" and was further split into a number of subcommittees to deal with many aspects of IT. Each subcommittee then divided into a number of working groups who deal with individual technologies. MPEG is the name given to working group 11 (WG11) of subcommittee 29 (SC29) with the title "Coding of Moving Pictures and Audio" [19].

MPEG development is split into ten subgroups utilising the wide expertise of the members including Systems, Video and Audio. There are over 300 members of MPEG who attend meetings, plus many more hundreds who participate in the work externally. The method of standardisation that MPEG uses, described in detail in [19], has many unique features that has helped the success of the MPEG specifications. Of most interest perhaps, are the notion of providing tools instead of systems, and specifying the minimum needed for interoperability. Defining tools allows MPEG to address the needs of many industries at once. As will be shown later, MPEG does have recommendations for combinations of toolsets depending on the desired application. The development of the MPEG standards has followed a bottom up approach with MPEG-1 and MPEG-2 describing the lower level encoding and multiplexing of video and audio, while MPEG-4 and upwards describe higher level semantics dealing with multimedia frameworks.

MPEG-1

MPEG-1 was the first standard to be published by the committee, in 1993. It is described in ISO/IEC 11172 and is split into 5 sections:

1. Systems
2. Video
3. Audio
4. Conformance Testing
5. Software Simulation

It is designed to provide audio and video at a bitrate of around 1.5 Mbps. The basic application for MPEG-1 was for VHS quality video on a CD. MPEG-1 has a number of firsts associated with it. It was the first standard to define only the

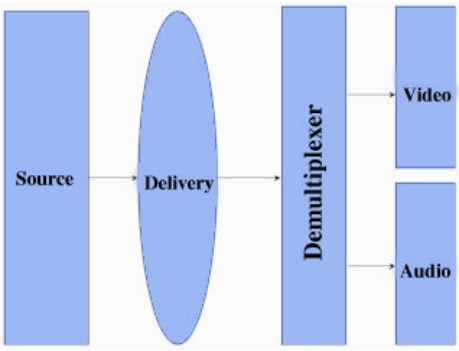


Figure 2.4: General Reference Model of MPEG-1 [19]

decoder. This follows from the minimum specification philosophy mentioned above. It is the first video standard to be independent of the video format. Also it is the first standard to be developed entirely in software and to include a software implementation (section 5). Figure 2.4 shows the model of MPEG-1. The delivery mechanism is optional in the case of a local disc source and normally describes a mechanism such as the telephone network or a data network. Audio and Video streams are constrained to having a common time base so that they can be recombined at the receiver system layer.

The flexibility of MPEG-1 has led to its success in many markets. For example, the Video-CD format is extremely popular in the far-east for pre-recorded media. MPEG-1 audio is being used for Digital Audio Broadcasting in Europe. Further more, MPEG-1 Audio Layer 3 is now synonymous with web users for exchanging music over the Internet (legally or otherwise).

MPEG-2

MPEG-2 is defined in the ISO/IEC 13818 series of documents. The motivation for MPEG-2 was to move digital video into the realms of broadcasting. In particular, certain parties found the progressive scanning of MPEG-1 to be an unnecessary constraint on compression. By using the extra redundancy to be found in interlaced video, a saving of around 20%was predicted [19]. To enable the quality demanded by broadcasters, MPEG-2 was to provide composite quality video at 6 Mbps and component quality at 9 Mbps.

To achieve this range of functionality, MPEG-2 more closely follows the toolkit approach mentioned above. These tools can be used in different combinations to allow the desired quality and features. To prevent anarchy, MPEG-2 standardises certain combinations of tools into 'profiles'. Each profile has certain

	Simple Profile	Main Profile	SNR Scalable Profile	Spatially Scalable Profile	High Profile	4:2:2 Profile	MVP Profile
High Level		X			X		
High-1440 Level		X		X	X		
Main Level	X	X	X		X	X	X
Low Level		X	X				

Figure 2.5: MPEG-2 Video Profiles [19]

parameters which are standardised into levels. These are shown in figure 2.5 , where only the combinations marked with an X are specified. For example, the main profile is currently being used for the majority of Digital Television services in Europe. The 4:2:2 profile increases the quality to make it suitable for professional video editing systems. Audio has been extended in MPEG-2 to allow full multi-channel capabilities. As well as a backwards compatible audio specification, there is also a separate specification for audio to make use of newer, more efficient algorithms such as the recent Advanced Audio Coding (AAC) standard.

MPEG-2 also specifies a number of features to assist in different aspects of content delivery. For example, the transport stream is defined so that it can carry a large number of programmes at any time. Tables are sent with the stream describing the nature and location of programs within the stream. Copyright protection is provided in MPEG-2 through the provision of streams that carry information necessary for decrypting any data in the transport stream that has been encrypted. The actual methods of encryption are not defined by MPEG.

To enable interaction between the client viewer and the source, part 6 of the MPEG-2 specification defines a communication protocol called Digital Storage Media Command and Control (DSM-CC). This allows content to be delivered to a user in a network environment where some control over the delivery is required.

The success of MPEG-2 is evident in the rapid growth of Digital Television and DVD (Digital Versatile Disc) Video. Each technology has taken MPEG-2 as a core technology for delivering the high quality demanded, while adding the necessary service customisations at higher levels, such as service information and copyright protection.

MPEG decoders are freely available with Windows for use in DVD playback applications and network video applications. MPEG coders are available, but normally incur higher processing costs due to their complexity. Until recently, MPEG encoding was restricted to dedicated hardware (such as the Pinnacle Studio MV10, above), but processor speeds and features are now allowing software coders to emerge such as the Ligos GoMotion codec [8] used in WinVCR

[7] and Pure DIVA [9]. However the problem with these encoders is that they are proprietary and incur high licensing costs to use.

MPEG-4

The latest finalised work of MPEG has been the introduction of MPEG-4[19][23]. It is important to note that a new MPEG standard does not mean it is better than previous standards, the difference is in the industries and applications they are aimed at.

MPEG-4 provides a higher level specification of multimedia applications, namely, the coded representation of natural and synthetics audio and visual elements and their composition into a scene. There are a number of motivations for providing this level of abstraction. Content creators can obtain a certain degree of material re-use, as well as easily integrate synthetic and natural elements. For the user, as composition is left until the receiver, the ability to tailor the final scene can be beneficial in scenarios such as news and information broadcasts. As well as adding animation and text as possible elements, all elements are treated as objects or AVOs (Audio-Visual Objects) instead of simple bitstreams. For example, a traditional news broadcast may be transmitted with the background as one object, the presenter as another, with the captions, picture insets and audio also sent separately. Any element can be natural or synthetic (computer generated). In the news example, it may be possible that everything except the presenters image and audio is generated synthetically.

MPEG-4 also has the goal of providing very low bitrate for use over constrained networks such as mobile phones and Internet. By using the above object oriented approach, together with more efficient video compression over previous MPEGs, the delivery of video can be scaled from broadcast applications, to low bitrate mobile applications.

2.4.3 Streaming Media

The Internet is playing an increasingly important role in audio and video delivery. Real Networks and Microsoft both produce technologies for providing real time streaming media over a network.

RealMedia[16]: The first RealPlayer was released in 1995 and was only able to play audio. RealPlayer is now at version 8.0 and is able to support a range of audio, video and animation codes. Version 8.0 vastly improves the video

quality on previous versions with near VHS quality at 500 kbps (kilobits) and full motion video at as little as 20kbps. Their encoder is available as an Software Development Kit and as an ActiveX control for simpler inclusion in applications.

Windows Media[17]: Microsoft responded to Real Networks by extending their Media Player product to support their own streaming media technology, now called Windows Media. The video codec of version 4.1 uses a subset of the early MPEG-4 standard. Only the video compression part of the standard are used. The current version is 4.1 with version 7 now in beta testing. Version 7 is no longer labelled as an MPEG-4 codec, and is now a totally proprietary technology. It also provides extremely high quality at broadband rates, boasting a near DVD quality video stream at only 750 kbps. Version 7.0 also redefines how the media is delivered and stored, with support for a new container capable of storing 17 million terabytes of video, and intrinsic support of Digital Rights Management. However, at the time of writing, there is no publicly available encoder or SDK.

2.5 Programming Technologies

The specification states that this project will be based on a Microsoft Windows 98/NT4 platform. Display and Audio technology has come a long way from the monochrome CGA graphics and PC Speaker of early home PCs. To begin with, application developers would write code themselves to support different makes of multimedia devices. As the number of devices grew, it became clear that it was infeasible for application developers to support them all and maintain their code. To that end Windows removed that hardware dependence by creating a uniform programming interface to hardware. For graphics programming, this abstraction was known as the Graphics Device Interface (GDI). Hardware manufacturers would then write device drivers to provide the hardware layer programming.

Different layers of abstraction now exist for the different services required by modern computers. Many of these are captured by the DirectX family of APIs [11]including the one of most relevance to this project, DirectShow.

2.5.1 Quicktime for Windows

Before discussing DirectShow and its predecessor, Video For Windows, it is relevant to mention an alternative architecture, Quicktime for Window (QTW). Quicktime is the video environment developed by Apple for their Macintosh

platform. It was initially ported over to Windows to enable playback only of QTW videos. As of version 3.0, the ability to edit and capture Quicktime videos under Windows had been added. Quicktime is favoured by many over Video for Windows (discussed below) as it has better overall performance and better synchronisation methods than VFW.

A notable feature of early QTW is that it provides its own device level drivers, bypassing the GDI and therefore improving performance. Newer versions now use DirectDraw (discussed below) to achieve similar performance.

2.5.2 Video For Windows

Video For Windows (VFW) is a multimedia framework designed for the Windows 3.1 platform by Microsoft. It was originally implemented as a collection of 16-bit utilities and drivers but has since migrated to 32-bit with the introduction of Windows 95.

The core of VFW is the AVI file format. This is a specialisation of the RIFF (Resource Interchange File Format), defined by Microsoft as a container for Audio and Video streams. It has since become a de-facto standard due to its wide support on Windows PCs and other platforms. However, AVI suffers from a number of limitations. The original AVI release was restricted to a maximum size of only 1 Gigabyte. This was increased to 2 Gigabytes with the release of the 32-bit version, and a professional extension created by the OpenDML group accommodates for files much larger than 1 Gigabyte [25], although this was rarely used. Microsoft have since moved on to other formats such as Windows Media for accommodating larger files.

The structure of the AVI file allows any type of codec to be used, as long as the codec was written as a Video For Windows driver. The most popular codecs in use are Intel Indeo and Cinepak. An M-JPEG driver also exists, made by a company called PICVideo[26].

2.5.3 DirectX and DirectShow

DirectX was developed by Microsoft to provide device independent access to a number of services. These are split into two layers, known as DirectX Foundation and DirectX Media. DirectX Foundation provides the low level access to devices:

- DirectDraw: API for accessing graphics devices
- Direct3D: API for accessing 3D graphics devices
- DirectInput: API for accessing input devices such as Joysticks
- DirectSound: API for accessing audio devices
- DirectPlay: API for providing network gaming functionality
- DirectMusic: API for providing Musical services such as MIDI

DirectX Media Adds the following services:

- DirectAnimation: API for supporting different media types such as vector graphics, sprites and images.
- DirectX Transform: API for creating transforms to applying space and time effects to 3D objects.
- DirectShow: API for capturing and playing back multimedia streams (such as Video and Audio) from local files and the Internet

At the time of writing, Microsoft have announced that DirectX Media will no longer be developed as a separate product to DirectX Foundation and will be integrated into the upcoming DirectX 8.0. The last version of DirectX Media is version 6.0.

DirectShow has its roots in Video For Windows (VFW). The initial release was flawed by inadequate audio and video synchronisation and poor overall graphics performance. Next came ActiveMovie, which addressed the synchronisation problems and added support for the MPEG standard. Finally, ActiveMovie was superseded by DirectShow, which is integrated with the DirectX SDK and supports new media sources such as the Internet.

DirectShow is implemented as a collection of standard Component Object Model (COM) interfaces which can be implemented or called by any compatible application. This is expanded in the next section.

Chapter 3

Design Overview

As the background shows, there are numerous technologies and applications in use today for capturing an playing digital video on a desktop computer. However, each has its weaknesses, either due to its age or due to the market it was originally designed to catered for. Only MGI's Pure DIVA, comes close to the functionality that would be ideally required in a VCR product. This chapter will now provide an evaluation of the technologies most relevant to achieving a final solution, beginning with a discussion of Video For Windows and DirectShow.

With regards to the choice of implementation languages, it was decided from an early stage to use Visual C++ 6.0 as installed in the Department of Computing teaching labs. This was mainly due to the authors experience with C++ and Visual C++, but also due to the availability of documentation and Software Development Kits (SDKs) for the various programming and compression technologies for Visual C++ and Windows Programming. Visual C++ also provides specific support for the Microsoft Foundation Classes (MFC), a collection of libraries that provide high level, object-oriented access to the many features of Windows Programming. It also provides libraries of abstract data types such as Lists, Arrays and Maps which remove the need for much of the memory and pointer management normally associated with C++ programming.

3.1 Suitability of VFW and DirectShow

3.1.1 Video For Windows Evaluation

Included as an integral part of Windows 95 and NT, Video For Windows has the widest support of the PC multimedia frameworks. However, it's relative age is also a primary concern when using it for modern multimedia applications. As mentioned above, under Win32 VFW has a file size limit of 2 Gigabytes. In addition, there is also a limit on the number of video frames (324'000). In reality, this would provide over 3 hours worth of video at 25 frames per second. Using the Intel Indeo compressor limits this to just over 2 hours before the file becomes corrupt. However, the VFW architecture will not warn if the file has exceeded these limits and will continue to capture to a corrupt file.

The Video For Windows sample application is known as VidCap and is included with the Windows Platform SDK. It is written in C, directly using the Windows API for creating a Windows application. It provides a simple interface for accessing the many features of VFW, allowing capturing of not only video, but also single frames and still sequences. It's configurability also illustrates one of the problems with the VFW architecture: there is no provision for hard coding the selection a specific codec for compression. this stems from the assumption that older computers could not perform real time compression while capturing. As a result, the only way to select a codec is from a dialog of all installed codecs.

Simplicity is also one of VFWs strengths. A user needs only to create a capture window, connect to a capture device, pass a few structures containing settings to the window and run a command to start capturing. The window created is of a class called AVICap which handles most of the underlying mechanics of connecting to and displaying the output from a video capture card. The AVICap class would also keep track of settings and statistics generated during capture, all accessible via system level procedures.

3.1.2 DirectShow

DirectShow, the video capture and playback component of the DirectX family of APIs, is Microsoft's successor to Video For Windows. Based around C++, there are a number of key coding differences introduced in DirectShow:

- Drivers and interfaces are now implemented as COM (Component Object Model) objects. This means that the interfaces have to be exposed via the COM mechanism before use.

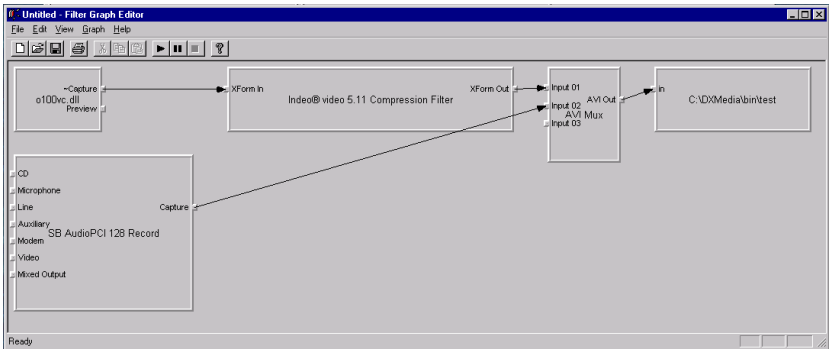


Figure 3.1: An Example DirectShow Capture Filter Graph - shown in the DirectShow utility, GraphEdit

- Capturing and rendering video is conducted through the notion of filter graphs. These graphs contain the input drivers, codec filters, and output writer/renderers necessary to create the desired output, along with the flow of control between them. Figure 3.1 shows an example capture filter graph. This is illustrated in GraphEdit, a utility for editing and testing filter graphs provided in the DirectShow SDK.

This structure allows an enormous amount of flexibility with the way multimedia is handled and processed. New filters can easily be written as long as they implement the standard interfaces that allow objects to communicate in the filter graph. Monitoring functions, such as dropped frames, are implemented as separate interfaces which must be instantiated and attached to the program when used. With VFW, statistics were always available from AVICap. The benefit of the DirectShow structure is that a multitude of graphs can be constructed for capturing and rendering not only video, but almost any type of streaming data.

However from an application programming perspective, it is now a lot harder to implement even the simplest of capture programs. The DirectShow SDK provides a sample capture application known as AMCap. It provides similar functionality to VidCap with one major exception: it does not include any compression. The graph it constructs simply connects the video capture driver directly to the AVI Multiplexer.

Adding compression is not a trivial task. Unlike VFW, DirectShow allows hard coded selection of compression drivers. However, there is no standard dialog as in VFW for choosing the codec. As a result, device enumeration and selection has to be coded manually.

Another major problem comes from trying to keep audio and video streams synchronised. During initial tests, the Intel Indeo Quick Compressor was inserted into the capture filter graph. However, when capturing, there was no synchronisation between audio and video with frequent dropped frames. There was no obvious method of keeping the streams synchronised in the DirectShow documentation.

3.1.3 Conclusion

The design decision to be made here is in which program, VidCap or AMCap, to use as a base for the VCR implementation. Using AMCap would mean having the best flexibility for handling media stream, but at the expense of some highly complex COM code. Using VidCap would mean that some basic functions, such as selecting the codec, would have to be left to the user at runtime. Also, with VidCap written in C using the Windows API, this would have implications when trying to add new features in Visual C++.

The deciding factor was to do with time. Staying with AMCap would entail trying to fix video synchronisation problems which do not exist in VidCap, thus taking time away from working on other aspects of the project. The result was to take VidCap as the platform of the implementation. However, this decision will create its own problems when trying to add new C++ features to the legacy C program.

3.2 Results of Compression Algorithm Testing

The choice of compression algorithm to use depends on a number of factors:

- Availability
- Cost
- Quality of output
- Processing overhead
- Size of encoded bitstream

As the implementation is required to capture real time video at VHS quality, any codec must be capable of real time capture. Unfortunately, the real time MPEG codecs from companies such as Ligos require a substantial licensing fee to use, and other MPEG codecs are unable to encode in real time.

Resolution	Quality	Average Bitrate	Notes
640 x 480	85%	6192 kbps	No preview, and high dropped frames
384 x 284	85%	3974 kbps	Good
320 x 240	85%	3452 kbps	Good

Table 3.1: Performance of the Indeo Quick Compressor

The encoders that were evaluated for this project were the Intel Indeo 5.11 codec, The PICVideo M-JPEG Codec[26], the Windows Media 4.1 system and RealSystem. The PICVideo codec is a proprietary real time version of the M-JPEG system used by numerous other manufacturers in both software and hardware. The Matrox G400 mentioned in the background incorporates a hardware M-JPEG encoder for capturing video. The first two systems are codecs implemented as VFW drivers. Windows Media and RealSystem are implemented using their own file format, designed for streaming media and were evaluated using their respective encoder applications. All results were gathered by capturing a 30 second clip from live television with 16-bit mono audio. Clips were captured 5 times to achieve an average. This is because many of these codecs operate as variable bitrate codecs.

3.2.1 Intel Indeo

As mentioned earlier, this codec is provided in two varieties: standard and quick compressor. The standard compressor achieves higher quality at lower bitrates than the quick compressor, but is unable to do so in real time making it unsuitable for this application. However, on a reasonably modern computer (the test platform is a Celeron 433) the quick compressor proves to provide suitable quality in real time. Table 3.1 shows the result of the Indeo Quick Compressor. It shows that the codec is unable to capture at anything above VHS quality. The quality setting of 85%is the default setting for the codec and provides a good quality picture at the given resolution. Either 320x240 or 384x284 would be suitable for use in this application. Full screen playback was excellent at these resolutions due to Indeo's interpolation features.

3.2.2 PICVideo Motion JPEG

The M-JPEG codec proved to be disappointing in tests as shown in table 3.2. At resolutions below 640x480, the picture appears blocky. The codec also requires registration to use, leaving a watermark on all encoded videos if registration is not done, or if there is a problem with installation. It also produced much

Resolution	Quality	Average Bitrate	Notes
640x480	4:1:1 Q18	10344 kbps	No preview, Dropped Frames
640x480	4:1:1 Q16	7648 kbps	No Preview, Dropped Frames
640x480	4:1:1 Q14	5613 kbps	No preview

Table 3.2: Performance of Motion JPEG codec

higher bitrates than the equivalent Indeo files. It was decided that this codec would not be used.

3.2.3 Windows Media & Real System

Both of these systems are designed for creating streaming media over a network connection. As a result, their emphasis is on low bandwidth connections. At the time of testing, neither systems were capable of producing VHS quality video in real time.

However, at the time of writing, both companies have announced major updates to their technology. Windows Media is now promoting version 7.0 as having television quality at 750 kbps, with VHS quality at 500 kbps. Unfortunately, no encoder has been released at the time of writing so no evaluation is possible. However, demonstrations are viewable at the Windows Media website [17].

RealSystem has upgraded to version 8.0, also delivering VHS quality at 500kbps. The encoder is available as RealProducer. However, the encoder is still unable to provide suitable quality in real time. The quality at 500kbps is suitably impressive to consider using for any offline compression. The encoder is heavily dependent on the speed of the host computer and normally takes about 50-70% longer than the length of a clip to compress on the lab computer (a Celeron 433), but the saving in bitrate along with the relatively low loss in quality meant that this could be successfully used as an archiving tool.

3.2.4 Conclusion

Based on these results, it was decided that the Indeo Quick Compressor would be the best codec to use for the real time capturing aspect of the implementation. It's speed and quality allows for VHS quality on a modest platform.

There is also the question of hardware video codecs. There are now an increasing number of cards that offer M-JPEG or MPEG compression on an expansion card so as to allow real time compression. However, due to their high cost an lack of availability during this project, they were not evaluated. It is also one of

the assumptions that a software VCR product should rely as little on extra hardware as possible, therefore allowing easier upgrading to take advantage of new encoding technologies. Something that would be difficult to do if the codec was built into hardware. It is also worth noting the lack of free, or low cost real time software MPEG compressors. Even though the Ligos GoMotion technology proves that it is possible, the use is still restricted to integrated applications.

3.3 Software Functionality Overview

The primary goal of the implementation is to replace the need for a separate VCR in the home. Therefore one of the most basic of functions to add is a timer for unattended recording of programmes. However, given the nature of recording on to a media that can't be removed, labelled and filed away, the facility to catalogue and manage recorded programmes must be added. Additionally, making use of information sources to provide a better user experience is one of the key advantages of creating a VCR application on network connected PC. Therefore an Electronic Programme Guide that can make recording programmes a matter of clicking on a programme list would be desirable.

This section will outline the decisions made in designing the functionality of these components. The timer is presented first.

3.3.1 Timer

There are features of a recording timer that must be provided by any VCR, whether hardware or software.

- The ability to select a start time, end time and channel to create a recording event.
- The ability to check whether a new entry will conflict with any existing timers.
- The ability to select some degree of timer repetition for programmes that are repeated at the same time daily, weekly, on weekdays or weekends.

In addition, the following properties will be desirable in a software recorder where the recording media cannot be removed.

- The ability to identify clips by a stored name for identification without playing the file.

The timer should also exhibit persistence between application sessions. It would be unreasonable to expect the user to keep the program open for the entire duration that timers are in existence for.

Any dialog should make the setting of a timer as simple as possible. It is widely regarded that the restrictions on a traditional VCR, for example the small remote control, LCD display or simplistic on-screen graphics, can make it very difficult to set a timer in some circumstances. Using the standard Windows controls allows the user to gain the maximum interface recognition and help maintain familiarity between Windows applications. It would be unreasonable to assume that a user of this implementation will have no prior knowledge of the Windows interface and could be considered a hindrance to impose any non-standard controls on the user. However, there is the argument that a user migrating from a traditional VCR will be most familiar with interface of the device they were using previously, therefore introducing the desire for this implementation to mimic the interface of that device. Doing so would restrict how advanced features could be used. For example, no current VCRs have the ability to store programme names with the clips. This has already been decided as a necessary feature for locating clips on a large hard drive, so would mean that the mimicked interface would still have to be customised for this feature.

A better way of providing customisations to the user would be with the introduction of 'skins'. Skins are customised dialogs created by third parties for use on an application. Popular media programs such as Winamp (http:\\www.winamp.com) and RealJukebox (http:\\www.real.com) already support them and have online libraries of hundreds of skins. However, this may be beyond the scope of this project given the time and resources available.

In conclusion, this implementation will use standard Windows dialog features for the timer, and subsequent modules.

3.3.2 Electronic Programme Guide

As shown with the DigiGuide and Pure DIVA applications, electronic programme guides are a powerful way of presenting information about channel schedules. However, both of these applications rely on proprietary information provided from their own network sites. This has the advantage that information can be tailored for the application. Also, providing the data themselves allows guarantees of availability.

However, there is a wealth of information already available by other means. Teletext has been a primary form of television listings for over 25 years but

is unsuitable for complex information such as programme descriptions. The emergence of the Internet as a source of information has also embraced TV listings. A number of websites have appeared which carry comprehensive listings for most UK terrestrial, satellite and cable channels.

Also, opening out the types of information source that are used will allow external developers to easily write support for new sources or maintain existing filters.

Early designs for this part of the project saw the sources of information being represented by programming APIs. Any developer wanting to write a new filter would have to implement a Dynamic Link Library (DLL) that exposes a set of known functions that return a known data structure with listing information. For example the following functions could be used for getting information:

```
BOOL getSourceName(LPSTR sourcename);
BOOL getListing(starttime, endtime, channel, resultstructure*);
```

Where getSourceName returns an identifier to the filter and 'resultstructure*' is the pointer to some publically known data structure holding a list of:

- Programme Name
- Programme Description
- Start Time
- End Time
- Channel

This would leave it to the developer to create the entire filter in a suitable manner. However, this would be an inefficient system if the majority of sources were network based. Developers would have to write their own network code and parsers to handle a website. A user with many of these filters installed will end up with a lot of libraries containing very similar code.

Another strategy is to create a generic parser that is able to read web pages and extract information based on some external configuration file. This has a number of advantages:

- A third party does not have to have specialist developers tools to create a new parser, a simple text editor would suffice.

- New parsers can be created without recompiling any code.
- The application is kept smaller by having a single parser and a collection of small configuration files.
- Even novice users can examine and alter the configuration files for their own sources.

Given the available time for the implementation, it was decided that a reconfigurable HTML listing parser would be created. No provision for other sources, such as Teletext, would be made at this time.

Full implementation details of the parser and the associated configuration language will be given in a later section. However, it is worth describing how the structure of the parser was formulated.

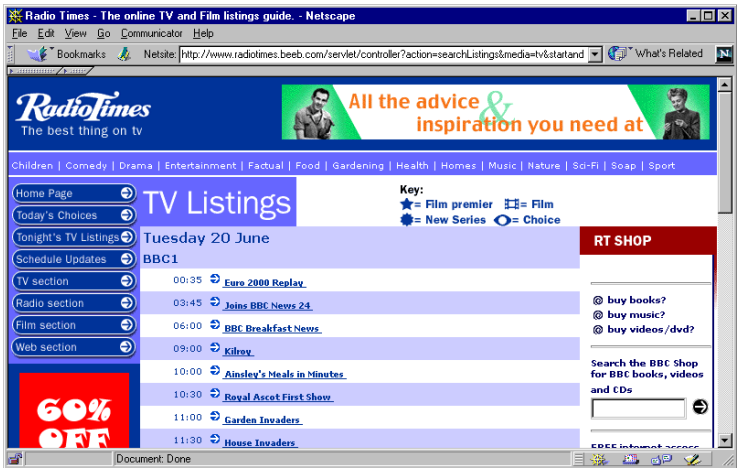
As described in the background, a HTML listing page consists of two elements necessary for gathering information: The URL (Uniform Resource Locator) containing the necessary query, and the HTML code containing the actual information. Therefore the parser problem can be decomposed into two problems:

- Constructing the correct URL query
- Extracting the correct information from the returned page.

Beginning with the latter, a web page can be thought of as a stream of HTML tags interspersed with user data. Here is the example of the BBC Radio Times web page seen in the background on page 12:

```
<TR>
   <TD><IMG SRC="file.gif"></TD>
   <TD><FONT><SPAN>23:45</SPAN></FONT></TD>
   <TD><FONT><SPAN><B><A HREF="programme details">
   <IMG SRC="arrow.gif"> Every Home Should Have One</A></B>
   </SPAN></FONT></TD>
   <TD></TD>
   <TD></TD>
</TR>
```

Figure 3.2 shows examples of the TV Listings web pages. Traditional parsers will work by constructing an internal representation of the entire page in a structure such as an abstract syntax tree. This representation will then be used



(a) BBC Radio Times Website - http://www.radiotimes.beeb.com

(b) Cable Guide Website - http://www.cableguide.co.uk

Figure 3.2: Screenshots from TV Listing Websites

for processing as necessary by the application. However looking at the above code, it can be seen that not everything needs to be read in to be understood by the parser. Only the second and third cells contain information of any use, the rest can be discarded. Also, there can be a large amount of pre- and post-amble on a web page before any useful information is found. In the example of the BBC Radio Times, a large proportion of the beginning of the page contains Javascript functions for handling the many custom interface features on the site.

Therefore, for the purposes of a parser which extracts known types of information, it is only necessary to track the position of the reader in the stream and trigger a recording when the correct tag is found. This idea of state-based triggering is most simply described by a Finite State Machine.

The configuration file can then be used to describe the combinations of states, triggers and actions necessary to navigate through the web page. Further detail shall be left for a later section.

Returning to the Programme Guide, the information retrieved by the parser should now be presented to the user in a clear and consistent manner, allowing the user to navigate the listings quickly. To make the guide useful to a VCR application, users should be able to use the listing to create timer events. The information gathered from website such as programme name can be used to catalogue the recorded programme in the video library.

3.3.3 Video Library

The Video Library should provide a means to access and manage the programmes recorded by the Timer and Recorder. The information stored by the library will be different to that needed by the timer and guide. In particular, the library needs to store information about the actual recorded file. The library needs to store:

- Programme Name
- Original Start Time
- Original Channel
- Duration
- File Path
- File Type

The first three are necessary for the user to identify the correct clip. The duration is used to indicate the resources being used by this clip.

One important notion throughout the application is that computer specific terminology such as disk space should be kept to a minimum. In a VCR application, a user will not be interested in how much disk space is being used by a programme or how much is left. They will much rather see the used and remaining resources in terms of absolute time. Thus an estimated remaining time is much more useful in judging what can be recorded than the remaining disk space. The library should display some amount of collective statistics. The estimated remaining recording time would be displayed here.

The File Type information is necessary for handling the possibility that a different file format is used for different needs. One example of this would be archiving of programmes. If a user is short on remaining disk space, but is reluctant to delete programmes from the library to make room, they may be more interested in being able to recompress a programme into a format that will occupy much less space, such as RealVideo or Windows Media. Obviously, this type of reduction in bitrate will inevitably lead to a reduction in quality, but for the purposes of archiving, a user will probably accept this given the gain in free disk space. To give some perspective on the saving, an Indeo QC video will generally be in the region of 3.5 Mbps. An equivalent RealVideo or Windows Media file will be 500 kbps, a saving of 85%

The functionality of the Library should therefore include a function to recompress a selected video. The original video can then be deleted to make space. Other management functions should include the ability to directly delete videos from the library and therefore the disk. Also, the ability to rename clips should be added, especially if the video has been added from hitting the 'Record Now' button in the recorder.

The question of whether recompression should be done while the application is unattended is as much a matter of implementation as well as design. It would be desirable for the library to recompress videos during idle time that the user has either explicitly designated as archive material, has reached a certain age or has been watched at least once by the user. One of the implementation problems with this is that the RealSystem codec can not be resumed if it is interrupted, therefore, any recompression should only be started if it is guaranteed to finish before any new timer events. This will be expanded on in the next chapter.

A final note regarding the recompression. It was decided that the RealSystem encoder, RealProducer, would be used for this application. This was because at the present time, Real Networks had introduced their RealSystem 8.0 codecs and

the RealProducer ActiveX control allowed access to these new codecs. The new Windows Media 7.0 gives better quality video, but there was no SDK available for the new system.

Chapter 4

A Software Implementation of SoftVCR

With the design decisions outlined above, the next area of focus is the implementation of the VCR. The program was created in Visual C++ 6.0.

Implementation was not linear. During early testing and evaluation of VFW and DirectShow applications, a prototype parser was created which demonstrates the viability of the Finite State Machine parser. From there, implementation began on the VCR with VidCap as the base application.

This chapter describes the structure and design of various stages of the VCR application, beginning with the prototype parser.

4.1 WebParser - A Prototype Application for Parsing Listing Websites

WebParser was created for the purpose of proving the viability of a Finite State Machine based parser. To keep the implementation simple, WebParser is specifically written to extract information from the BBC Radio Times website.

It is written as a single dialog MFC application in Visual C++. Figure 4.1, shows the main dialog. The application allows the user to select a date and time range, along with the desired channel so that a query may be constructed. The available channels are hard coded into the application. The main window will then display the returned results.

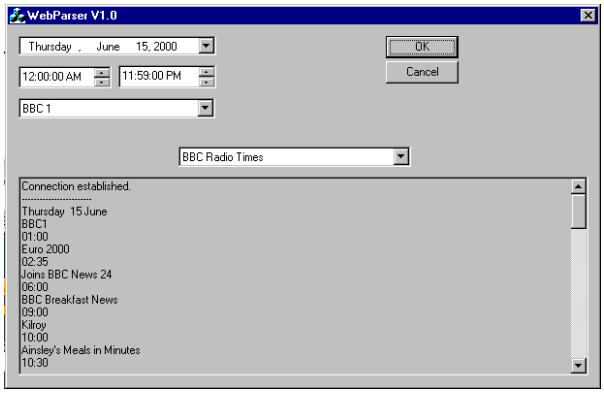


Figure 4.1: Screenshot of WebParser

The first task of the parser is to take the information from the dialog and create a URL to connect to the website. This is accomplished by constructing a string part by part, inserting the necessary information in the correct place. The following code segment shows the insertion of the date after the initial string:

```
CString theURL = "http://www.radiotimes.beeb.com/rt
/tc/listings_search/index.jsp?startandenddate";

//Date must be in format YYYYMMDD

tempstr.Format("%.4d", date.GetYear());
theURL += tempstr;
tempstr.Format("%.2d", date.GetMonth());
theURL += tempstr;
tempstr.Format("%.2d", date.GetDay());
theURL += tempstr;
```

The rest of the URL is constructed similarly. MFC provides a number of classes for dealing with network connections. The classes removes the majority of the network and session protocol handling from view, leaving the developer with a text or data stream almost identical to that found with memory or local files. The classes of interest here are CInternetSession and CInternetFile. The former handles the TCP and HTTP connection to the requested network address, while the latter provides the stream interface to the received file.

When the connection is successfully to the website, the stream is passed onto the next stage, the parser for processing. Without going into line by line detail, the process of parsing can be summarised as follows:

1. Search the stream until a known initialising string is found. ie. '<!– start of programme listings table –>'.
2. Set the current state to INITIAL and start reading the stream one tag at a time.
3. On each tag, check whether current state contains a trigger containing this tag. If not, read in next tag and continue.
4. If there is a trigger, perform any action associated with the trigger then change to next state. Read in next tag and continue.
5. Continue until end of file or TERMINATE state is reached.

The state is usually used to describe at what point in the page the parser is at. States may indicate that a certain cell has been found or that the parser is currently in a certain row. The use of states allows the parser to be unrestricted by the form of the page, so a page that does not have a table will still be parsible by some other combination of states.

Actions are optional entities. In many cases, the trigger is simply there to move the parser into a new state. An action will usually take the form of an instruction to extract a piece of text. For example a time, programme or description. The following code fragment shows how the programme name is extracted:

```
switch(currenttag){
case href:
   switch(parserstate){
   case programmefound:
      //get and print programme name
      extractText("VSPACE=\"3\">", currentline, &tempstr);

      parserstate = gotprogramme;
      break;
...
```

Here, the parser is instructed to extract the text following 'VSPACE="3">' and dump to screen as the programme name. The parser state is then changed and the loop continues.

Also illustrated here is the use of switch statements for the decision making. The switching takes place on two levels; in this prototype, it switches first on the current tag, then on the current state. As strings can not be used in switch

Figure 4.2: Screenshot from VidCap

statements, the tag is first converted into an enumerated type. Similarly, the state is represented by an enumerated type. The advantage of the use of switch statements are their speed of processing. However, it does imply that all possible tags and states have to be pre-programmed into the parser a compile time.

As the screenshot in figure 4.1 shows, the parser is successful at extracting channel information, programme times and programme names. However, this prototype only extracts the text and displays it to the user. It makes no attempt to interpret the information or construct any data structure. It does prove, however, that a Finite State Machine is a suitable model for extracting information from an HTML web page.

4.2 Video Recording - Extending VidCap

The decision to use Microsoft's VidCap sample capture application introduces a number of both technical and design issues. VidCap is a simple interface to the Video For Windows API with a host of options for customising the captured video. Figure 4.2 shows the standard capture window from VidCap.

In terms of functionality, it is clear that a number of the features in the VidCap interface are not needed. The toolbar, for example, can be purged of the following buttons:

- Set Capture File - This will only need to be set once, to set the path of the video library. So it should not be placed on a toolbar which is associated with frequently used functions.

- EDIT CAPTURED VIDEO - There is no provision for video editing in this application.

- TOGGLE PREVIEW/OVERLAY - It is generally the case that Overlay provides the best quality of preview with the lowest demands on the system. Therefore, overlay should be set by the system as a default. There is no need to burden the toolbar with these controls.

- CAPTURE SINGLE FRAME/SEQUENCE - As a VCR application, it is assumed that the function of the program is to capture video clips. Therefore, the function to capture frames and manual sequences is a burden on the functionality that is undesirable.

- CAPTURE PALETTE - This is an advanced feature, not needed by the VCR.

Similarly a number of the menu items, mainly representing equivalents to the above features, can be removed. The only exception is the 'Set Capture File' option, used to set the path of the video library.

4.2.1 Architecture

VidCap is written in C, directly accessing the Windows API for the creation of frames and windows. It implements its own modules for creating toolbars, status bars and video frames.

One of the key decisions to be made when modifying VidCap was how to add the necessary features to the application in an clean and efficient manner. It would be beneficial to be able to use the visual programming features of Visual C++ wherever possible, so it was decided that the other program modules would be written in C++ using the MFC classes. Therefore, some way of allowing VidCap to access these modules is necessary. The Windows concept of DLLs (Dynamic Link Libraries) provides a suitable mechanism for doing this.

A DLL is a collection of functions, written in a similar fashion to an executable, except instead of a single entry point, the library provides access to a number of publically declared functions to any program that links to it. In fact, the Windows kernel is implemented as a DLL for this reason. The public functions are declared in such a way any programming language that knows the name of the function, and its argument types, can call them correctly, including C.

4.2.2 Adding Support for Hauppauge

Before writing these libraries, a number of functions must first be added to VidCap. One of the first modifications to be made is to add native support for the Hauppauge WinTV card.

The Video For Windows architecture allows basic control of a video device. In the case of the WinTV card, the user is able to select which of the inputs (Tuner, Composite, SVideo) to use. However, there is no way to manipulate the tuner itself, or turn on the audio. These functions can be accomplished by use of the Hauppauge WinTV SDK.

The SDK works by providing a handle to the card, with which a number of properties can be set. The handle is retrieved with the instruction:

```
AVHdlW INAPI AV_InitByDetail()
```

The arguments to this function are used to define the type of WinTV card and features expected of the application. In this case, the only specific parameter expected is that the board is of type hcwHW_BT_WinTVPCI.

The SDK only has 3 other entry points:

```
DWORD W INAPI hcwAV_Get()
DWORD W INAPI hcwAV_Set()
DWORD W INAPI hcwAV_Free()
```

The first two are used to set and retrieve the many properties of a Hauppauge card. The last is used to release the hardware when the application is finished with it.

After initialisation, the primary use of the Hauppauge commands is to allow the user and timer to change channels on the tuner. Additionally, support for the composite input is added to allow for a user to record from a source other than the tuner, such as a satellite or digital television decoder. Access to television channels is via the menu bar, where the desired channel can be selected directly from a drop down list. Also the toolbar features buttons for browsing the channels.

The final function of the Hauppauge SDK used is the muting of audio whenever a dialog is launched. This is so the user is encouraged to return to the main screen if he/she wishes to watch television. Also, it allows the dialogs to run their own video playback without worrying about the audio clashing with the tuner output.

Figure 4.3: SoftVCR interface

The channel settings are stored in an external configuration file. There is no provision currently for editing these settings in the program, but the file is a simple text file with a channel identifier and channel number, separated by a whitespace.

4.2.3 Interface Changes

Figure 4.3 shows the interface to SoftVCR. Nominally, it still has the same look and feel as VidCap. The fundamental change is seen with the toolbar. From left to right, the buttons are:

- Play - Plays the last recorded video.
- Guide - Launches the Electronic Programme Guide.
- Timer - Launches the Timer dialog.
- Lib - Launches the Video Library dialog.
- Rec - Start recording now.
- -/+ - Navigate tuner channels.

Recording has been simplified from the process needed in VidCap. The capture frame rate is hard coded to be 25 frames/sec in the program. The audio is set to the value read from the Windows configuration file. The compressor codec is chosen at launch and remains consistent for the remaining session. When capturing is started, VidCap requires the user to confirm or change these settings, then confirm the start of capturing. These steps have been removed

to allow a 'One Touch' record function. Once started, video will be recorded until the Escape key is pressed. A feature of VFW means that these hot keys are global to all windows, no matter which window on the users desktop has the current focus. Therefore, the Escape key was chosen as an easy to remember keystroke, which a user will have minimal need to press in other windows while recording. A note in the SoftVCR status bar reminds the user that the Escape Key stops recording.

Finally, the filename for videos are generated so that they are unique for each recording event. This is done by appending the current date and time to 'softvcr'. this is done at the start of the recording event, so the time and date will reflect the start time of the recording. A filename with the path:

```
softvcr20000604174601.avi
```

Indicates that the file was created at 17:46:01 on 4th June 2000.

4.3 Video Playback

With recording simplified and set to generate unique files, some method of playing these files back within the application is required. This is accomplished by creating a DLL that contains a single dialog featuring the Microsoft Media Player ActiveX control. The DLL has a single point of entry:

```
BOOL playVideo()
```

This takes a single argument giving the path of the video to play. This command will simply instantiate the dialog class and display it as a modal dialog, one which removes control from the parent window until closure. The video will then automatically start playing. Figure 4.4 shows the dialog window for the video player. Even though Media Player provides comprehensive transport controls, it was felt that the dialog would benefit from an explicit control to show the video in full screen mode. Once in full screen mode, the standard Media Player menu can be accessed via the right mouse button.

In use, the DLL is only loaded when necessary using Explicit Linking. This saves on resources, and allows the module to be used in any part of the program without the need for Header or .lib files.

One of the functions of the Video Library, to be described below, is the ability to recompress programmes into RealVideo to save disk space. Therefore, it is

Figure 4.4: Video Player Screenshot

necessary for the program to play RealVideo files as well as AVI files. Unfortunately, the current version of Media Player is unable to support RealVideo to the the desired level (version 8.0) . The solution to this is to launch RealPlayer itself via a shell command if the file type is seen to be RealVideo. This will be further described below in the Video Library.

4.4 Timer

The timer of a video recorder is perhaps one of its most important features. However, it is often regarded as being one of the most difficult to use for many people. Statistics show that as many as 20% of VCR owners are unable to set a timer[1]. Technology has improved for traditional video recorders in recent years. For example, VideoPlus codes are often published in listings magazines. The codes are a unique number that will set a video to record a specific channel at a specific time. All the user need do is enter this number onto their VCR to program it. Unfortunately, the implementation of VideoPlus is a commercial secret so support in SoftVCR would be impossible at this time. Also, there has been a huge improvement in the use of On Screen Displays for VCRs allowing step by step guides for users to set the VCR. The restriction though is often the method of input. VCR remote controls can be complex pieces of equipment in terms of layout and their diminutive size. Every manufacturer will have their own design for a remote control, but their overall flexibility is restricted by their size.

However, it would be highly beneficial to use a recognised interface such as Microsoft Windows, along with a Qwerty Keyboard and Mouse, to create a

[1]Source: BRMB, cited by Mintel, 1994 -
http://www.commerce.uq.edu.au/isworld/research/msg.05-08-1998.html

timer that is as easy to program as it is to type a letter or check email.

Setting a timer in SoftVCR can be accomplished via two methods. The first, a custom timer dialog, will be described here. The other way is via the Electronic Programme Guide and shall be described in a later section.

4.4.1 Object Persistence

The Timer is the first module of the program that requires and uses persistence between sessions. It is necessary for the timer list to be stored in stable storage so that the program can be stopped and started without losing settings. Visual C++ provides a mechanism for giving objects persistence using serialization. Under MFC, as long as a class is derived from CObject then it can use the serialize( ) function to dump its member variables to a stream. Serialization also handles all other object state parameters such as pointers or circular references. CArchive is then responsible for making sure this stream can be reconstructed into an object and acts as an intermediary between serialize( ) and the CFile object. The provisions for serialization to work under Visual C++ are[27]:

- Each object is responsible for serializing itself. Therefore a class must implement the virtual function serialize( ) ;
- When deserializing an object, its exact type must already be known.
- When deserializing an object, there must be memory already allocated for the data.

4.4.2 Serializing the timer

The timer is represented by a time ordered list of events. For the purposes of inserting new items, it is beneficial to use an array. Each event holds a start time, end time, programme name, channel name, channel ID (for the video recorder) and a frequency. The frequency describes how often a timer event should take place and can be either *once*, *daily*, *weekly*, *week days* or *weekends*.

The timer array is implemented as a member variable of the main timer dialog class. This is done so the process of of serialization can be concisely controlled by the lifespan of the timer dialog object. The serialization is controlled by use of two functions LoadRecords( ) and SaveRecords( ) , which respectively deal with the task of deserializing and serializing the timer array from the file on disk. LoadRecords( ) is called when the timer dialog object is constructed. SaveRecords( ) is then called whenever a member function is finished modifying the array, or on destruction of the dialog.

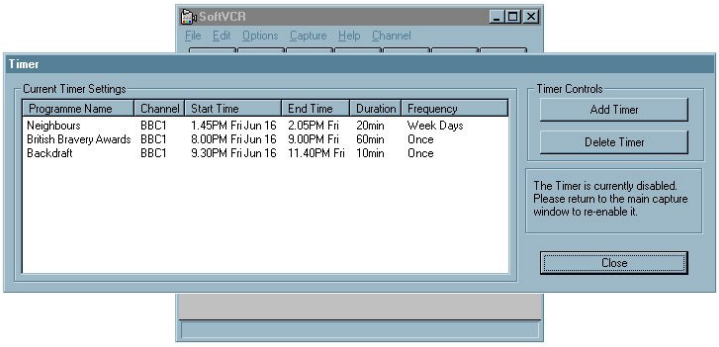


Figure 4.5: The Timer Dialog

4.4.3 The Timer Dialog

Figure 4.5 shows the main timer dialog. From the user perspective, it simply presents the current contents of the timer array in a multi-column CListCtrl view. There are two functions that can be performed on the list, either adding a new event, or removing one. The latter is the simpler of the two to implement. As the list view is drawn in the same order as the timer array, the index of the selected entry is used to delete the relevant element from the array. A confirmation is requested from the user as deletion here is irreversible.

4.4.4 Adding a Timer

Adding a timer is a two stage process. First the desired event must be constructed, then it must be inserted into the timer array. The first is handled by a separate dialog, but can also be bypassed by taking information direct from the electronic programme guide, as will be described shortly. The second is handled by a separate function, which must check whether the new event will overlap with any entry already in the array.

Figure 4.6 shows the dialog for constructing a new timer event. As discussed above, it was decided that the timer should use familiar Windows common controls for capturing user input. Upon initialisation, the date controls show the current time and date. As the user changes the times, he/she is presented with the total duration of the current selection. Upon pressing OK, the dialog validates the form to make sure that a channel was selected, and that the time selection is not in the past.

If everything is okay, then the event information is passed to the addToTimer function. The task is now to attempt to insert this event at the relevant place

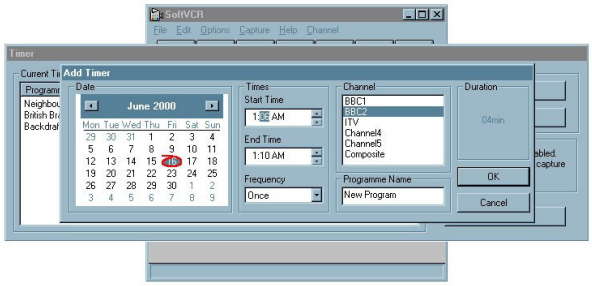


Figure 4.6: Adding a new timer in SoftVCR

in the timer array. The following pseudo code outlines the algorithm used for insertion:

```
if (array empty){
   insert newevent
}else{
for(i=size of array; i--){
   Get element at i
   if(newevent finishes before currentevent){
      if(no element before i){
         insert newevent at i
      }else{
         get element at i-1
         if(previousevent finishes before newevent starts){
            insert newevent at i
          }else{
              break
          }
       }
   }else if(newevent starts after currentevent){
      if(no element after i){
         insert newevent at i+1
      }else{
         Get element at i+1
         if (nextevent starts after newevent finishes){
            insert newevent at i+1
          }else{
              break;
          }
       }
   }
}
```

This algorithm ensures that any overlapping entries are avoided and that entries are time ordered in the array. Insertion into an array is possible due to the MFC implementation of a dynamic array class, CObArray.

4.4.5 Running a Timer

The task of running a timer is the responsibility of the video recorder. In this implementation, the video recorder polls the Timer every 10 seconds to check if the first item in the array is due to start. If the polling function is successful, it returns the parameters of the timer event: name, starttime, duration and channel identifier. The duration is the difference between the events start and end time, corrected for the polling time so as not to overrun and delay subsequent recordings. It also corrects for if the timer was only activated after a timer was due to start. Video For Windows can take a duration as a parameter for capturing, so recording will stop automatically. The polling function will finish by deleting the current event off the front of the array.

Upon completion of recording, the video recorder will add the new recording to the library. The library will be described in detail in the next section. Finally, the recorder will check the timer for the next event, in case it should start immediately, and then resume polling.

4.5 Video Library - Cataloguing and Managing Video Clips

The video library provides a method for viewing and managing the recordings made with SoftVCR. Figure 4.7 shows the main dialog for the video library. It allows a user to rename and delete videos as well as play the back. It also allows a user to recompress the videos using RealVideo. As with the timer dialog, persistence is achieved through the use of a serialized data structure. A library entry records the video path, programme name, channel, channel identifier, duration of the clip, file size and file type. Instead of an array, library entries are held in a map structure, using the video path as a key. Map (or Dictionary) structures allow fast retrieval of data if the key is known.

The video library is implemented as an isolated DLL with two entry points; *showLibraryDlg*, which instantiates and displays the library dialog as a modal dialog of which ever window called it, and *addToLibrary*, which takes a number of parameters used to create a new library entry. It will then instantiate the class and add the entry to the Library map.

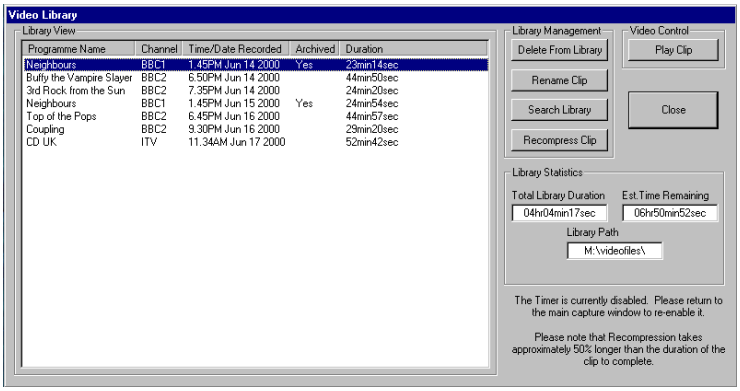


Figure 4.7: The Video Library

As with the timer dialog, the map is deserialized whenever the class is instantiated and serialized whenever a change to the map takes place.

4.5.1 The Library List

Displaying library data is accomplished using a multi-column CListCtrl view, again similar to the timer dialog. However, the CListCtrl view is now responsible for sorting ordering the library. This is because the map structure no longer guarantees the order that data will be retrieved when it is iterated through. SoftVCR will show the list in date of original recording order. The archive column is used to indicate whether a clip has been recompressed. This will be expanded on below.

Videos can be deleted from the library as well as have their programme names changed. This is especially useful if the video was added to library by the recorder after a manual recording.

Playing the videos will first check the file type of the selected video. If the file is an Indeo AVI, the library will invoke the Player DLL incorporating Media Player. If the file type is an archived RealVideo, the library will pass the filename to the shell so that the registered player can handle it. This will usually launch RealPlayer, from which the user can have full control of playing back the video.

4.5.2 Showing Statistics

One of the most basic functions of the library is calculating and displaying a number of statistics about the library. For this implementation, the Library path, total library duration and estimated time remaining are shown. The

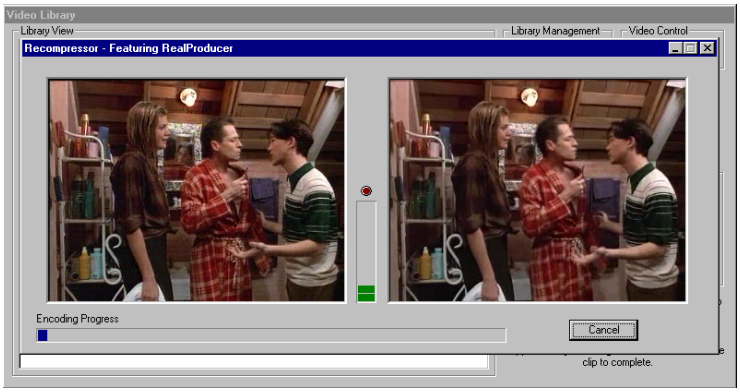


Figure 4.8: Recompression Dialog in SoftVCR

library path is simply passed to the Library dialog as an argument during construction by the video recorder. The library duration is calculated by iterating through the library map, summing the filesize attributes.

The estimated time remaining aims to give the user a rough idea of how much disk space is remaining. Although showing the actual disk space would be a lot more accurate, it would be meaningless when trying to judge if an hour long show can be fitted into the library. Calculating the remaining library space is accomplished by finding the remaining disk space in the library directory. Then if the library is empty, the disk space is divided by a best guess bitrate for the Indeo Quick Compressor at 320x240. This is hard coded as 400'000 bytes per second. However, if the library has entries, the average bitrate of the entries is calculated by dividing their filesize by their duration. This average bitrate is then used in estimating the remaining time. This method also gives a more realistic view on remaining time if the user has a mix of Indeo and RealVideo files in the library.

4.5.3 Recompressing Videos

Perhaps the most powerful feature of the library is the ability to archive videos by recompressing them with RealVideo. The choice of RealVideo was down to the availability of an SDK and ActiveX control for their new Version 8.0 video codec, giving extremely high quality video at 500kbps. This is approximately 1/6th of the bitrate of the Indeo AVI files being captured by the recorder.

Recompression is handled by the RealProducer ActiveX control, inserted into the Recompressor dialog shown in Figure 4.8. The ActiveX control gives full control over the quality settings for the desired session. So as to keep the

use of SoftVCR simple, these settings are set to a default at compile time. RealProducer uses the notion of Target Audience to set default settings for a certain bitrate. For this application, the Target Audience is set to 512 kbps with variable bit rate encoding to keep the quality high. Variable bit rate mode does not allow the bitrate to go above the target audience setting.

Encoding time is dependent on the computer being used. It is not necessary for the recompression to be done in real time so constraints on performance are more relaxed than those used for the capture codecs. On the lab test platform, a Celeron 433MHz, encoding takes approximately 50-70% longer than the length of the clip.

RealProducer does not allow an encoding session to be resumed if it is interrupted. This leads to a number of considerations for the recompressors use. First, the nature of the program architecture requires that timers are disabled during recompression. Therefore a user should not start a session if there is an impending timer event. The user is warned of this, along with a reminder of the time it takes to compress, and the slight loss in quality, before encoding begins. The user can then choose to proceed or return to the library to perform other tasks. This implementation also only performs recompression when the user explicitly starts it. The implications of this will be discussed in the Future Work section on page 65.

To help the user judge how much time is remaining in the current encoding session, the dialog features a progress meter that is updated by an event handler of the ActiveX control.

The dialog object also takes responsibility for updating the entry in the library. The file type is changed and the file size is updated with the new value.

In conclusion, the recompressor provides a useful method of archiving material. Even though there is a slight loss in quality, the file size has been reduced to 1/6th of the original AVI file.

4.6 Electronic Programme Guide

The Electronic Programme Guide provides a way of navigating and using television listings to set timers. The main dialog is shown in Figure 4.9. Here, the main feature is another CListCtrl view which shows the listings for a chosen television channel and date. Additionally, the dialog shows the current Timer list in the lower pane. As the timer and programme guide share a DLL, incorporating the timer in this dialog is simply a case of including the timer class

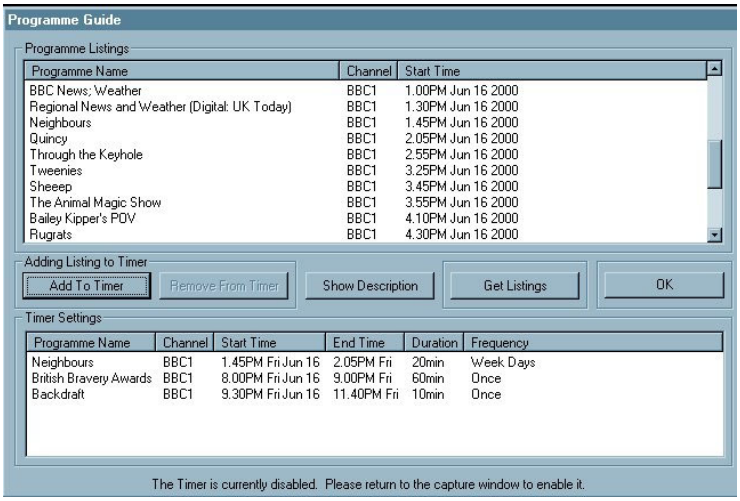


Figure 4.9: The Electronic Programme Guide

header file. The timer and programme guide share the data structure used to represent a single programme event.

Adding a programme to the timer list is simply a case of selecting the programme from the guide, and clicking on the 'Add To Timer' button. The handler performs the same checks on space and collisions as the manual timer dialog. The user is also able to delete timers from the list. As the timer and programme guide share the same data type, it is simply a case of copying the structure over to the timer list. Care must be taken not to copy only the pointers to the structures but their entire contents as well. This is to prevent problems when destroying the list on exit.

As with other data structures in the programme, the current listings are saved to disk using serialization.

4.6.1 Getting Listings

To update the main pane with listing for the desired listing, the user will click on the 'Get Listing' button to bring up the dialog shown in Figure 4.10. From here, the user must choose which source to get the listings from, the desired channel and the desired date. Upon pressing 'Okay', the parser will connect to the website and retrieve the full days listings.

It is also here where the user can add support for any additional data sources. Clicking on 'Add' will prompt the user for the path of an LPF (Listing Parser Format) file. If the file is valid, the user will then be prompted with a series

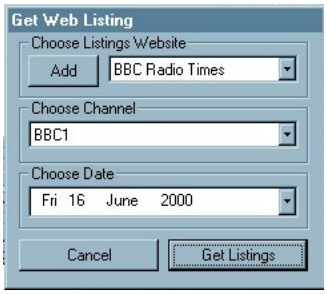


Figure 4.10: Get Listing Dialog

Figure 4.11: Parser Setup Dialog

(a) In-line Description (ie. Cableguide.co.uk)

(b) External Description (ie. radiotimes.beeb.com)

Figure 4.12: Showing a Program Description

of dialogs such as the on in Figure 4.11. Each of these asks the user to select which channel on the website relates to each channel on the tuner. In this implementation, it is a one to one mapping, so when a channel is selected, it is removed from the list for the rest of the process. Once the user has performed the mapping, they can now select the new source from the Get Listing dialog. The settings for the parser are recorded through serialization.

4.6.2 Showing a Programme Description

An additional feature is the ability to display a description of the currently selected programme. As will be shown in the next section, programme descriptions generally come as in-line text, or as an external link. To support this, the 'Show Description' button will either display the text in a Message Box, or bring up a dialog containing the Microsoft Internet Explorer ActiveX control so the external link can be displayed. Figure 4.12 shows the two alternatives.

4.7 Parsing HTML Television Listings

Building on the prototype WebParser seen in section 4.1, it is now necessary to expand the parser to allow it to be reconfigured via script files. Again, parsing

is divided into two tasks, constructing the necessary query and processing the results. Both of these are described in the Listing Parser Format.

4.7.1 Listing Parser Format

The scripting language created to describe a parser is designed as much for ease of processing as it is for ease of programming or reading. To keep it as simple as possible, statements are kept to only one per line. HTML-like tags are used to delineate different sections of the file, while attributes consist of colon separated parameters.

Although the parser that reads in the file is not restricted to a certain ordering of sections, it is recommended that the file keeps the following format:

- <PARSERINFO> - General parser information such as name and base URL.
- <CHANNEL> - List of channels with their numeric identifiers. In the format *NumID:ChannelName.*
- <URL> - Sequence describing URL description. Attributes are in the format *Type:Content.*
- <URLATTRIBUTE> - Used to describe the separation of pages necessary for acquiring a full day of listings. For example, Cable Guide requires 4 pages, as they provide 6 hours of listings per page. This section indicates which attribute needs to be updated for each iteration to get a full day of listings.
- <ATTRIBUTE> - Describes the format of the times, dates and descriptions on the pages.
- <STATETABLE> - A series of state descriptions describing the machine. It consists of two types:
  - <INITIALIZE> - The string that the parser skips forward to before starting the main process.
  - <STATE> - The actual state description consisting of a name (*state*), next state (*next*), trigger (*trigger*), and action (*action*). If the action is non-null, then additionally a *searchstring* and *attribute* are specified

A full description of the language can be found in Appendix B, but some of the key features will be described below.

The channel section is in fact only used by the Parser Setup routine described in section 4.6.1 when determining the mapping of tuner channels to listing identifiers.

4.7.2 Describing a URL

The <URL> section instructs the parser how to construct a query URL for the given website. As in the prototype WebParser, the URL is built up section by section with relevant user data inserted in the correct places. This is described in the LPF file as a list of elements describing which data to insert:

- STRING tells the parser to insert the string on this line directly in without modification.
- STARTTIME, ENDTIME, DATE are used to indicate that a parameter of the given time should be written to the URL. The attribute is the strftime style identifier for the desired time part. For example, %Y is the 4 digit year or %H is the hour in 24hr format.
- RELATIVEDAY is used to support web queries where the day is specified as a number relative to today. For example, 0 represents today while 1 represents tomorrow, etc..
- PERIOD is used to support web queries that divide the day into set periods. No attribute is needed, but the period type is set up in the URLATTRIBUTE section.
- CHANNELID is used to tell the parser to insert the current channel identifier in the URL. This will always be a string, so no attribute is necessary.

It is impossible to verify whether the above attributes can account for every variation of channel listings. Descriptions of successful implementations will be given below.

4.7.3 Parsing the Page

The states are written to to a Map structure, with one element for each state. Each element contains arrays to hold the trigger, next states, actions, searchstrings and attributes. Arrays are used for two reasons. First, a state may have more than one way of moving on. Secondly, using arrays keeps indexing consistent between the different attributes, something that can not be guaranteed with Maps.

The above structure differs from the prototype by first searching for the current state and reacting to tags. This was done in reverse by the prototype.

The parsing algorithm can be described by the following pseudo-code:

```
currentstate = 'initial';
parsermachine.Lookup(currentstate, stateinfo);
while(getTag succeeds){
   for(int i=0; i < stateinfo.trigger.GetSize(); i++){
      if(stateinfo.trigger[i] found in currenttag){
         if(stateinfo-> action[i] == 'TERMINATE') {
            return
         }
         if(stateinf-> action[i] == 'NEW ENTRY'){
            create new programme guide item
         }
         if(stateinfo-> action[i] != NULL){
            check attribute[i] and use searchstring[i] to extract text
            and save to correct attribute of the programme guide item.
            If we have found a time. Use as finish time (minus 5 sec) of
            previous guide entry and save previous guide entry to list.
            Then use as start time of current guide item.
         }
         if(stateinfo-> action[i] == 'ENDENTRY'){
            previousGuideItem = currentGuideItem;
         }
         currentstate = stateinfo-> nextstate[i];
         parsermachine.Lookup(currentstate,stateinfo);
         //move on to next state
      }
   }
}
```

If the parser does not successfully reach the terminate state, then it will return an error. This would be the case if the query was invalid or there are no listings for the specified time.

A problem with this architecture is that the parser is seen to operate as a series of string pattern matching statements. Although this frees the parser from having restricted enumerated types as in the prototype WebParser, it does increase the processing cost of the parser. However, on the test platform, the delay is barely noticeable.

To give an example of how the parser works in practice, the configuration for the Cable Guide will now be discussed.

4.8 Case Study: Cable Guide

Cable Guide (cableguide.co.uk) is one of a number of UK based websites that provide comprehensive listings for most terrestrial and satellite channels.

Beginning with the construction of the Cable Guide URL, the format is:

```
http://www.cableguide.co.uk/bread/go?
date=17/01/2000&time=03%A00&theme=&alias=8&type=list
```

This can be broken down into the following LPF attributes:

```
STRING:http://www.cableguide.co.uk/bread/go?date=
DATE:%d
STRING:/
DATE:%m
STRING:/
DATE:%Y
STRING:&time=
STARTTIME:%H
STRING:%3A
STARTTIME:%M
STRING:theme=&alias=
CHANNELID:
STRING:&type=list
```

No end time is specified as the site is designed to return the listing for the following 6 hour period. Therefore the following <URLATTRIBUTE> is defined to instruct the parser that 4 pages must be retrieved for a full days listings:

```
ATTRIBUTE:DATE
PERIODDURATION:6
PERIODDURATION:6
PERIODDURATION:6
PERIODDURATION:6
```

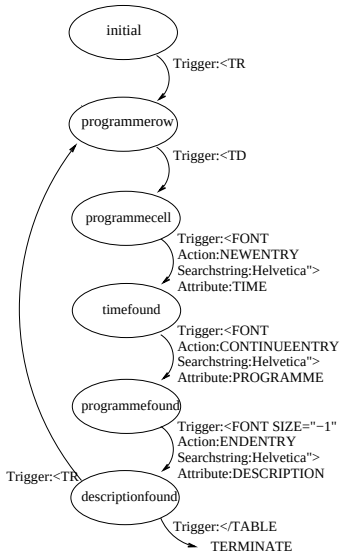


Figure 4.13: State Diagram for Cable Guide

Finally the format of information on the cableguide pages is described by the following <attribute> section. DATEDELIM and TIMEDELIM describe the delim-inating characters in the strings in addition to the space character. DESCRIP-TIONDELIM describes the terminating character when reading in the description.

```
DATE:%A %dxx %b
DATEDELIM :
TIME:%I:%M %p
TIMEDELIM ::
DESCRIPTIONTYPE:INLINE
DESCRIPTIONDELIM ≺
```

Finally the state map can be defined. The intialization string refers to the tag beginning the programme table. Figure 4.13 shows the state diagram used to parse the page.

The main problem with the Cable Guide parser is the amount of time needed to acquire a full days listings. It is necessary to parse 4 pages where as the BBC Radio Times only needs to parse a single page.

4.9 User Evaluation

As the implementation is meant for consumer use, it was decided that evaluation by actual users would be beneficial to the overall evaluation process. A select

group of people were chosen to use the program during an individual session. the session would have the following form:

- The user would be shown the operation of the program during a brief walkthrough.
- The user would then be asked to perform a number of simple tasks with the program.
- All comments are recorded during both stages.

Presented here is an overview of the feedback received from the users.

With regards to the main capture window:

- Several users noted that the advanced features left in from VidCap were confusing and could be detrimental to the application. They should either be removed or changed.
- One user tried to resize the capture window to enlarge the screen. This is not possible because of the VidCap architecture.
- +/- buttons could be ambiguous. Are they a volume control or a channel control? Should there be a volume control here?
- Pressing Escape to stop recording was found to be confusing for some users. Some suggested having a stop button.
- Play button should be next to record button.

With regards to the library:

- A user noted that they were not able to continue to watch television while using the library.
- It was suggested that a program could be archived while you are watching it.
- A user requested some type of editing facility to remove adverts or lead ins.
- Other UI enhancements suggested: A trash can for deleting videos. More use of features such as dragging, right clicking or double clicking to perform functions.

- The search function should complain if the search date is in the future.

With regards to the timer:

- The EPG and Timer dialogs should be reachable from each other rather than having to return to the main capture window.
- Manual timers and recordings should automatically get their programme names from the EPG.
- Warning dialogs were noted by many to be confusing in the nature of their errors and how to go about solving them. For example, when a conflict is detected, the dialog should tell the user what it is conflicting with and offer a range of options for resolving the conflict.
- On setting the timer, users had little trouble understanding how to add one. Some were frustrated by the up/down controls on the standard Date-Time Control.
- It was suggested that setting a timer could be more of a step by step wizard to help the user

With regards to the Electronic Programme Guide:

- Several users asked why a time range could not be selected when retrieving listings.
- Several users noted that the guide should either not display old programs, or have some way of discriminating old programs from programs yet to come.
- Several users tried to show the description of a timer entry. Something that is not supported at present.
- Most users said that this was the best feature of the program. However many suggested that the programme information should be used throughout the program.
- Given that the EPG only displays information for one channel at a time, displaying the channel as a column seems redundant. Same with date.

With regards to the program in general:

- It was noted that the idea of recording onto a computer hard drive was a good idea because of the efficiency of use of space. It is often hard to find a traditional cassette with spare room to tape a program, but with SoftVCR, all spare space can be used and the estimated time left is shown.
- One user noted that they would be cautious in using their computer as a VCR because they feared it might crash, where as a traditional VCR has no such problems.
- It was noted that there was no fast forward or rewind buttons on the video players, and that these might be preferable over the seek bar employed by Media Player and Real Player.
- Several users felt that buttons could be better represented by images.

Overall, feedback was positive for the idea of the program and the functionality. The above comments mainly outline the problems users had with the User Interface and their suggestions for improving it.

Chapter 5

Conclusion and Future Work

This project has examined the area of desktop video recording and its application to the replacement of a separate hardware video recorder. In addition, the ability to use external data sources for acquiring information about programmes was investigated, concentrating on the use of TV listing web sites. The implementation of a desktop recorder, named SoftVCR for Windows, was described here. It incorporates a web page parser that can be reconfigured for a wide range of sources based on a configuration file format called Listing Parser Format (LPF).

The achievements of the project can be summarised as follows:

- The Microsoft sample Video For Windows capture application, VidCap, was successfully adapted for use as high level video recorder and television application with support for the Hauppauge WinTV PCI capture card.
- Real time, VHS quality video can be captured to disk using the Intel Indeo 5.11 codec at a much lower processing overhead than than equivalent MPEG based codecs. Space required by Indeo is in the region of 1 hour per gigabyte.
- Further savings in disk space can be made by recompressing videos using RealVideo 8.0. The space required is 1/6th of Indeo with only a slight loss in quality.
- Modules written in Visual C++6.0, using the MFC libraries, were successfully integrated with the VidCap C architecture using DLLs to encapsulate functionality. The program also successfully uses available ActiveX controls to provide enhanced functionality without excessive programming.

- An Electronic Programme Guide which allows a user to browse current TV listings was implemented. These listings can then be used to set recording timers as desired.
- A reconfigurable parser for extracting TV listings from websites was produced. The parser was based on a Finite State Machine implementation, designed to quickly extract relevant information from an HTML page.
- The program uses serialization to give a level of data persistence that would otherwise require database integration.

Another goal of this project was to implement a VCR using freely available technologies that are free from restrictive or expensive licensing. This was largely achieved by use of the free Intel Indeo codec and RealProducer ActiveX control.

Feedback during the User Evaluation was generally favourable for SoftVCR. Of particular interest to many was the Electronic Programme Guide, which was seen as making timer setting much easier. Many of the changes suggested during User Evaluation were User Interface changes, rather than functionality flaws.

5.1 Commercial Potential

Although other commercial applications are now appearing on the market that provide VCR functions for people with TV cards, this program is unique in it's approach to acquiring programme information from public Internet sites. Also, the use of Indeo 5.11 provides VCR quality video at a lower overhead than commercial MPEG-1/2 encoders, giving the application wider support than rival products. The success of applications such as RealJukebox also show that there is a demand for applications that can effectively manage multimedia stored on a home PC.

There are no plans to take this program beyond the confines of personal and academic use. Issues of copyright have been largely overlooked with this implementation. Websites such as Radio Times and Cable Guide rely on revenue brought from advertising on their sites, so this effective filtering may provide a barrier to this product being used commercially. Also, there is now a demand for audio and video data to be secured to protect the distribution rights of the content providers. There is no provision for this in Video For Windows or Indeo.

On a personal note, I may continue to develop the program to take advantage of new compression technologies and improve on the interface as suggested during the user evaluation. The following section, on extensions and future work will

describe some of the features that could bring this implementation closer to a commercial offering, as well as some future areas of investigation.

5.2 Extensions and Future Work

As is the case with most projects, the functionality of the presented program has been largely limited by the time and resources available. Although the implementation of SoftVCR provides a useful illustration of how external data can be combined with a video recorder to provide an method of recording and managing video, there are a number of features that could be developed. Many of these are based on feedback from users made during evaluation.

MOVE THE VIDEO CAPTURE PROGRAM AWAY FROM VFW & VIDCAP, TO A DIRECTSHOW BASED APPLICATION:

The use of VidCap as a base for development allowed the emphasis to remain on adding extra functionality to the capture program rather than trying to get basic video capture functioning to the desired level. However, using a DirectShow or COM based architecture would yield better performance in the long run for capturing and playback through the use of DirectX and other, more modern SDKs.

EXAMINE THE USE OF OTHER COMPRESSION TECHNOLOGIES. REMOVE THE USE OF AVI AS THE RECORDING FILE FORMAT:

Intel Indeo provides a very good quality for real time compression of video at VHS quality. However, the video market is now embracing the MPEG standards as the platforms for Digital Video. It may be beneficial to use MPEG in the future, especially if Digital Television tuners allow a user to record the MPEG2 stream direct to disk. MPEG4, although designed for as an object oriented multi media framework, also provides marked improvement in compression efficiency over previous versions [23] .

The use of AVI as a file format also imposes several unnecessary constraints on the system such as a frame and file size limit of 324'000 and 2 gigabytes respectively. Although several custom extensions exist, such as the work of OpenDML[25] , a different container mechanism should be used.

Microsoft's Windows Media 7.0, announced and previewed in the closing stages of this project, may provide another viable architecture for a VCR application. It claims to be able to deliver television quality at as little as 750 kbps, a quarter of the rate being achieved with Indeo. Until further information is released, it is not clear whether this quality can be achieved in real time. Even so, it

may provide a better alternative to the RealVideo recompression being used at present. Also, the file container for Windows Media 7.0 is capable of storing as much as 17 million terabytes, virtually freeing the user from any worries over individual file size. Other features such as digital rights management may also benefit a commercial VCR application so that piracy is less of an issue.

Examine the use multimedia description frameworks such as MPEG-7 for recording metadata about videos, and use data for enhancing recording and management:

Even though MPEG-7 is still very much work in progress [24] , it aims to provide a framework for describing video and audio content. If this is embraced by the broadcast industry, this could further enhance the ability of a recorder to record relevant programs and provide a way of managing data more succinctly. The current database is restricted to recording basic programme information such as name, time and channel. This would be enhanced by recording more metadata about programs such as subtitles or keywords.

Further use and enhancement of the Electronic Programme Guide

User feedback generally said that information gathered by the EPG could be used in other parts of the programme. Features such as a 'Whats On Now' interface, or automatically labelling manual recording timers could be added.

Work on the User Interface:

Probably the most sensitive part of any consumer based application is the user interface and its ease of use. Although general feedback was favourable for the interface presented in SoftVCR, many comments were made about how the program should provide more of the Windows UI features that computer users take for granted such as:

- Tree Views for cataloguing data in the library.
- Context sensitive option menus when right mouse button is clicked.
- Playback by double clicking on items in library.
- Custom graphical interfaces, possibly with support for skins - interchangeable interfaces.

Support for transferring videos to other media:

The current application is restricted to recording and managing data in a single directory. It may be desirable to provide support for transferring to and managing videos on removable media such as CD-R, Zip or Jaz drives.

Appendix A

User Guide

Welcome to SoftVCR for Windows V1.0, the replacement for your VCR. This guide will give an overview of how to use SoftVCR's features to record your favourite programmes, then view and manage your collection of videos.

The Main Window

The main window of SoftVCR allows you view your television and gives access to the other features of the program.

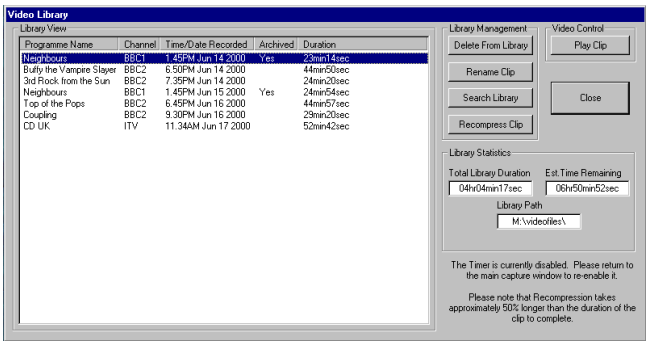
If you have not used SoftVCR before, you will have to tell it where you want to store the videos it will record. This can be done by going to File->Set Capture File and browsing to the desired directory. Once this is set, you will not have to worry about it again. You may also have to setup the TV card if you cannot see a picture. Go to Options, then select either video source or video format.

If in the UK, the Source should be set to a PAL tuner. In video format, select 24bit, 320x240. Once set, these settings will not need changing.

The main controls on the toolbar from left to right are:

- Play - Play the last recorded file.
- Guide - Launch the Electronic Programme Guide.
- Timer - Review and set manual timers for recording.
- Lib - View and manage the videos in your video library.
- Rec - Start recording now. On hitting escape, the video will be added to the library automatically.
- -/+- Browse channels. Channels can also be selected directly from the channel menu item.

Using the Library



The library is where you are able to play and manage the videos that you have recorded.

Library View - This is the list of videos in your collection. They are ordered by date, so newer programmes will be at the bottom of the list.

Video Control- Select a video and click 'Play Clip' to launch the player.

Library Management - From here, you can *delete* and *rename* videos. To use, select a video from the Library View and click on the desired operation. If necessary, you will be asked to confirm your action.

You can also search the library for clips by name or date of recording. Click Search library at any time to make your selection. Any matching items will then be highlighted in the Library View.

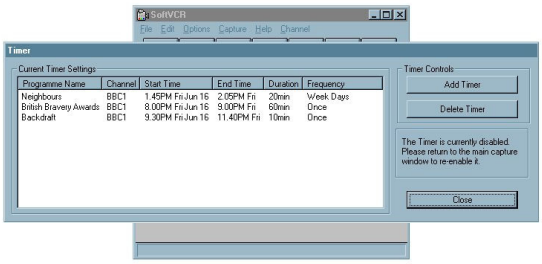
ARCHIVING - If you are running low on disk space, but do not want to delete an important video to make room, you can use the *Recompress Clip* function to reduce the video to 1/6th of its original size. Simply choose the video you wish to archive and press Recompress Clip.

Please Note: Archiving takes time. Depending on the speed of your computer, it can take as much as twice the length of the video to complete. If interrupted, you will have to restart from the beginning. Therefore, it is probably best not to archive if you wish to record or have a timer set for the required time. Timers are disabled until you return to the Main Window.

Setting a Timer Manually

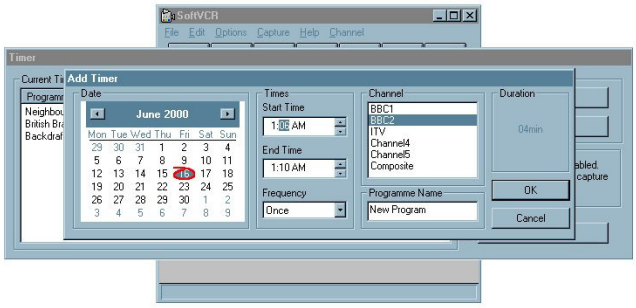
There are two ways of setting a timer: manually and using the Electronic Programme Guide. The EPG will be described later, but here is how you set a manual timer.

Clicking on *Timer* on the main window takes you to the following screen:



This screen allows you to view what Timers you have currently set. You can then choose to delete them, or add new ones by clicking *Add Timer*.

Add Timer will show this screen:



From here, simply select the date you want to record on, the start time, the end time and desired channel. You can then enter a programme name, so the timer can be easily identified, and the library will show this name when it is recorded. The duration of your current selection is shown on the right.

You can also choose if you want the programme to be recorded on a regular basis by choosing from the *Frequency* drop-down menu. A programme can be recorded either *once*, *daily*, *weekly*, *week days* or *weekends*.

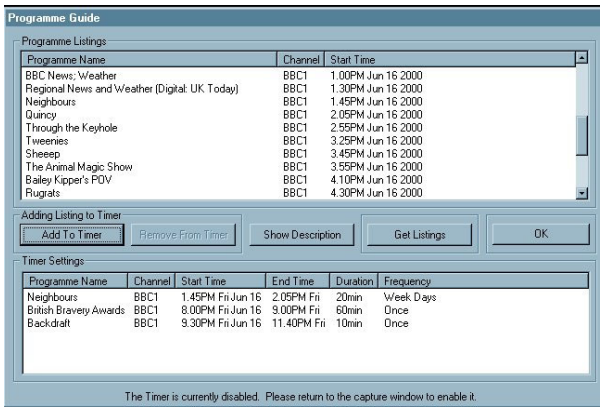
If there are no conflicts, and there is enough space in the library, the timer will be added to your list after you hit *okay*. If there is not enough space to store the programme, you will be taken to the Library where you can do any deletions or archiving until there is space. Closing the library returns you to the Timer, where you can now attempt to add the programme again.

To enable the timer. Simply return to the Main Window.

Please Note: Timers are disable while any dialogs are showing. To re-enable them at any time, simply return to the Main Window.

Using The Electronic Programme Guide

The Electronic Programme Guide (EPG) provides an easy way to see what is showing on your television channels and when. When launched from the Main Window, the EPG screen is shown as follows:



The upper window shows the listings, while the bottom window show your current Timer settings. At first, the EPG will show the listings from the last time you updated the display.

Getting New Listings - To update the listings, click on '*Get Listings*'. Choose your desired source, channel and date, then hit 'Get Listings' to retrieve the information. There may be a slight pause while the program contacts the source to get the up to date listings.

Adding a New Source - SoftVCR can retrieve listings from any website which has had a configuration file written for it. If you have acquired the configuration file (known as an LPF file) for a new source, you can add it to SoftVCR by clicking on 'Get Listings' in the EPG and then clicking on 'Add', next to the source selection. You will be prompted to choose the new LPF file. If the file is valid, you will be presented with one of these screens for each television channel on SoftVCR:

This screen asks you to tell SoftVCR which page on the new source corresponds to your television channel.

Once complete, you will now find the new source as an option on the source drop-down menu.

SHOW DESCRIPTION - If you click on a programme in the programme guide and click 'Show Description', a short description of the programme will be shown either in a small dialog box, or in a web browser dialog.

ADDING TO THE TIMER - To add a programme in the EPG to the timer, you simply select the desired programme and press 'Add To Timer'. You will be asked to confirm the action and select the desired frequency. If there are no conflicts with other timers, and there is enough space in the library, the programme will now show in the Timer Window at the bottom.

Appendix B

Listing Parser Format (LPF)

The listing parser uses a configuration file to allow it to support a wide range of of possible web-based listing sources. The file format created to configure the parser is called Listing Parser Format(LPF).This section will provide a guide to what an LPF file contains and how one should be written.

Structure

An LPF file describes 4 general pieces of information:

1. A list of the channels supported by the source
2. Details of how to construct a query URL and gather 24hrs of listings.
3. The format of numeric and time values on a page.
4. The Finite State Machine necessary to parse a page.

An LPF file is a simple format which limits the layout of the file to a single statement on each line. The file must begin with the following line as a file identifier:

```
<LPF V1.0>
```

The remainder of the file is composed of tagged sections, representing different blocks of information needed by the parser. All sections are necessary. Tag names and attributes are case-sensitive. The order of the sections is not restricted to, but is recommended to be the following:

- <PARSERINFO>
- <CHANNEL>
- <URL>
- <URLATTRIBUTE>
- <ATTRIBUTE>
- <STATETABLE>
  - A state table contains the following nested sections:
  - <INITIALIZE>
  - <STATE>

The general form of an attribute statement is:

```
attributeID:parameter
```

The statement is colon separated, but the parameter itself may contain colons if necessary.

Sections are terminated using the HTML-like syntax of </SECTION>. Individual lines can have tab and space lead-in to help improve readability.

The contents of each section will now be addressed.

ParserInfo

The <PARSERINFO> section provides general info about the parser. There are two attributes that must be provided:

```
PARSERNAME:name of parser
BASE URL:base address of parser URL
```

Channel

The <CHANNEL> section provides a list of one or more television channels that are available on the website, together with the string or numeric identifier that is used in the query URL. The format of the attributes must be:

```
id:channelname
```

Url

The <URL> section provides instructions on how to construct a query URL for the website. Attributes are assembled into a single string in a top-down order. A number of attribute types are allowed to try and accommodate the construction types of as many URLs as possible. These are:

STRING:parameter

The parameter is inserted directly in to the URL without processing.

DATE:parameter
STARTTIME:parameter
ENDTIME:parameter

The parameter here is a *strftime* style identifier for formatting part of a time into a string. Use these attributes for inserting necessary parts of the date, start and end time.

RELATIVEDAY:

This attribute takes no parameters. It instructs the parser to insert an integer which represents the requested day relative to today. For example, today will be day 0, tomorrow is day 1, etc...

CHANNELID:

This attribute takes no parameters. It instructs the parser to insert the ID of the selected channel as described in the <CHANNEL> list.

PERIOD:

This attribute takes no parameters. It instructs the parser to insert the current period as described in <URLATTRIBUTE>.

URLAttribute

The <URLATTRIBUTE> section describes any changes that must be made to URL in order to gather a full 24 hours of listings. It must begin with the following attribute:

```
ATTRIBUTE:parameter
```

Where the parameter is currently either DATE, indicating the increase in hours between pages. Or PERIOD, indicating that the following attributes should be inserted in the URL directly as period identifiers.

Following this line must be one or more of the following:

```
PERIODDURATION:parameter
```

Where the parameter is the number for the increase in hours between pages, or the string period ID to insert.

Attribute

The <ATTRIBUTE> section describes how the parser should interpret dates, times and descriptions on the webpage.

The following attributes must be included:

```
DATE:parameter
DATEDELIM:parameter
```

with the parameter to DATE being the strftime representation of the date. One addition to this is *%dxx* which represents a day followed by 2 letters as in 1st, 2nd, 3rd, etc. The parameter of DATEDELIM is a list of any delimiters in the DATE parameter which are in addition to the whitespace character.

```
TIME:parameter
TIMEDELIM:parameter
```

This is the same as DATE, but for the strftime representation of the times.

Example: 9th July 2000 will be:

```
DATE:%dxx %B %Y
DATEDELIM:
```

Example: 8:40 pm will be:

```
TIME:%I:%M %p
TIMEDELIM::
```

Finally,

```
DESCRIPTIONTYPE :parameter
DESCRIPTIONDELIM :parameter
```

The parameter of DESCRIPTIONTYPE must be either INLINE or EXTERNAL, depending on whether the description on a webpage is actually on the page (inline) or accessed by following an HTML link (external).

The parameter of DESCRIPTIONDELIM is the character which terminates a description.

StateTable

The <STATETABLE> section is used to describe the Finite State Machine used to parse the page. It must begin with the <INITIALIZE> subsection, and will the be followed by one or more <STATE> subsections. These will now be described.

Initialize

The <INITIALIZE> section represents the string that the parser will seek to before starting to parse using the FSM. It has a single attribute:

```
STRING :parameter
```

Where the parameter is the searchstring.

State

The <STATE> section describes a state, with its associated trigger, attributes and actions. A state with more than one trigger must be represented by using a different <STATE> section for each trigger. The state name must remain the same between triggers if this is the case.

The following attributes must be defined in each <STATE> section:

```
STATE :parameter
NEXT :parameter
TRIGGER :parameter
ACTION :parameter
```

The parameter of STATE is the string ID of the current state.

The parameter of NEXT is the string ID of the STATE that the parser will move to if the trigger is found.

The parameter of TRIGGER is the string which must be found in the current tag for the parser to make the trigger.

The parameter of ACTION must be one of the following:

```
NULL - no entry to perform
TERMINATE - stop parser
NEW ENTRY - start a new programme entry
CONTINUEENTRY - continue with current entry
ENDENTRY - current entry ended, save entry.
```

If the ACTION is not NUIL or TERMINATE, the following attributes must be defined:

```
SEARCHSTRING:parameter
ATTRIBUTE:parameter
```

Where the parameter of SEARCHSTRING is the string immediately preceding the information to be extracted. The parameter of attribute describes the type of the information being extracted and can be one of the following:

```
PROGRAMME
DESCRIPTION
TIME
```

Example State Table - Cable Guide

The following state table is used by the *CableGuide.lpf* configuration included in the source code for SoftVCR V1.0. There is also a file for the BBC Radio Times, *BBCRadioTimes.lpf* included with the package.

```
<STATETABLE>
   <INITIALIZE>
      STRING:<TABLE WIDTH=470 BORDER=0 CELLSPACING=3 CELLPADDING=7>
   </INITIALIZE>
   <STATE>
```

```
    STATE:initial
    NEXT:programmerow
    TRIGGER:<TR
    ACTION:NULL
</STATE>
<STATE>
    STATE:programmerow
    NEXT:programmecell
    TRIGGER:<TD
    ACTION:NULL
</STATE>
<STATE>
    STATE:programmecell
    NEXT:timefound
    TRIGGER:<FONT
    ACTION:NEW ENTRY
    SEARCHSTRING:Helvetica">
    ATTRIBUTE:TIME
</STATE>
<STATE>
    STATE:timefound
    NEXT:programmefound
    TRIGGER:<FONT
    ACTION:CONTINUEENTRY
    SEARCHSTRING:Helvetica">
    ATTRIBUTE:PROGRAMME
</STATE>
<STATE>
    STATE:programmefound
    NEXT:descriptionfound
    TRIGGER:<FONT SIZE='-1'
    ACTION:ENDENTRY
    SEARCHSTRING:Helvetica">
    ATTRIBUTE:DESCRIPTION
</STATE>
<STATE>
    STATE:descriptionfound
    NEXT:programmerow
    TRIGGER:<TR
    ACTION:NULL
```

```
    </STATE>
    <STATE>
      STATE:descriptionfound
      NEXT:endfound
      TRIGGER:</TABLE
      ACTION:TERMINATE
      </STATE>
</STATETABLE>
```

Bibliography

[1] Hauppauge Web Site for information on Win/TV cards
http://www.hauppauge.co.uk

[2] Hauppauge SDK Documentation
Included with SDK, available from Hauppauge [1]

[3] BT848/848A/849A Single-Chip Video Capture for PCI Data Sheet
http://www.conexant.com/data/TechDocs/External/100117A.pdf

[4] ATI All-In-Wonder website.
http://www.ati.com/uk/Pages/products/pc/aiw_128/index.html

[5] Matrox Marvel website.
http://www.matrox.com/mga/products/marv_g400/home.htm

[6] Pinnacle Systems - Information on miroVideo.
http://www.pinnaclesys.com

[7] Cinax WinVCR website
http://www.winvcr.com

[8] Ligos Technology. Makers of MPEG codecs for licensing in other applications
http://www.ligos.com

[9] MGI Pure DIVA website
http://www.purediva.com

[10] DigiGuide website
http://www.digiguide.co.uk

[11] Microsoft DirectX homepage
http://www.microsoft.com/directx

[12] R Coelho, Maher Hawash. "DirectX, RDX, RSX, and MMX Technology"
Addison Wesley, 1998

[13] Intel Indeo Software homepage
http://developer.intel.com/ial/indeo/index.htm

[14] Cinepak website
http://www.cinepak.com

[15] Chad Fogg. "MPEG2 FAQ"
http://bmrc.berkeley.edu/frame/research/mpeg/mpeg2faq.html

[16] Real Networks website. Home of RealMedia
http://www.realnetworks.com

[17] Microsoft Windows Media website.
http://www.microsoft.com/windowsmedia

[18] Digital Video Broadcasting (DVB) Project
http://www.dvb.org

[19] L Chiariglione. “MPEG and Multimedia Communications” CSELT
http://www.cselt.it/ufv/leonardo/paper/isce96.htm

[20] MPEG Project - http://www.cselt.it/mpeg

[21] MPEG. “Short MPEG-1 Description” CSELT [20]

[22] MPEG. “Short MPEG-2 Description” CSELT [20]

[23] MPEG. ‘Overview of the MPEG-4 Standard’ CSELT [20]

[24] MPEG. ‘Overview of the MPEG-7 Standard’ CSELT [20]

[25] OpenDML AVI File Format Extensions v1.02.
http://www.jmcgowan.com/odmlff2.pdf

[26] Pegasus Imaging Corporation - Makers of PICVideo Motion JPEG codec
http://www.jpg.com

[27] Microsoft Developer Network (MSDN) Library
http://msdn.microsoft.com

[28] R Leinecker, T Archer. ‘Visual C++Bible’
IDG Books, 1998

[29] C Petzold, P Yao. ‘Windows 95 Programming’
Microsoft Press, 1996

[30] Codeguru - Visual C++Tips (source of CToken class)
http://www.codeguru.com