Video Capture in Microsoft Windows

Yun Teng
MSc of Information Systems

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student) _______________________________
Summary

Nowadays, multimedia technology has become more and more popular. Every university in Britain has research in multimedia area, such as visualization, pattern recognition, etc. This project is also used to help visualization research group staff to do video capture application in Windows.

This project involves research into the different type of video devices and formats that are supported by Windows and the development of an image source class to capture video that can be used with an existing Image toolkit that has been developed in the School of Computing.

The aim of this project is to use DirectShow technology to capture video from camera in Windows and support movie file format. Video source could be from disk or real time capturing. A DLL library is built for video application of this project.

This report emphasises on how the project was carried out, what all techniques were used and what functions can the methods of DLL provide. It states how the project management going, and justify why the methodology used in this project is correct. It evaluates the software designed from several ways. Also this project states future works of video capture application.

The minimum requirements for the project have been achieved and exceeded in several areas.
Acknowledgements

I would like to take this opportunity to express my sincerest gratitude to all those people who have helped me to make this project a success.

In particular, I would like to mention my supervisor, Dr. Andy Bulpitt. His expert advice and individual help gave me a direction of this project, while his friendly and patient attitude always made me feel confident and go through stressful challenge of a Masters project.

I would like to thank Dr Lydia Lau. During the whole year study and this project, she gave me a lot of invaluable guidance and suggestions. She deeply influenced my mind, not only in study, but also in how to be a kind people. Especially during the project, she resolved my private problems and encouraged me to keep going. Without her help, my study and project would be not have been completed so smoothly.

I would like to thank Professor Kia, who is my assessor, gave me advices on Mid Project Report and progress meeting.

I would like to thank all my classmates and friends in Leeds for their help during this whole year.

I would like to thank my Dad and Mum, without their love, support, continual encouragement and inspiration, I couldn’t have the opportunity to be here and finish my degree.
# Content

1 **Introduction**
   1.1 Problem Statement ................................................................. 1
   1.2 Problem Domain ................................................................. 1
   1.3 Required Research ............................................................ 2
   1.4 Report Structure ............................................................... 2

2 **Project Management**
   2.1 Introduction ........................................................................... 3
   2.2 Objectives ............................................................................. 3
     2.2.1 Project Aim ................................................................. 3
     2.2.2 Minimum Requirements .................................................. 3
     2.2.3 Further Requirements ..................................................... 3
   2.3 Project Plan ........................................................................... 4
   2.4 System Development Methodology ......................................... 5

3 **Background Research**
   3.1 Introduction ........................................................................... 7
   3.2 The need for Video Capture ................................................... 7
   3.3 The application of Video Capture ............................................ 8
   3.4 Application Programming Interface ....................................... 9
   3.5 Component Object Model ..................................................... 11
     3.5.1 IUnknown Interface ....................................................... 12
     3.5.2 DirectShow COM Interface ............................................. 12
   3.6 DirectX/DirectShow ............................................................ 13
     3.6.1 DirectX ....................................................................... 13
     3.6.2 DirectShow ................................................................. 14

4 **Design and Implementation**
   4.1 Introduction ........................................................................... 19
   4.2 Case Study ............................................................................ 19
     4.2.1 DirectX Video Processing App Wizard .............................. 19
     4.2.2 Audio/Video Capture Example AMCAP ........................... 20
   4.3 System Architecture ............................................................ 21
   4.4 Utilising UML ....................................................................... 21
     4.4.1 System User-Case Diagram ............................................. 21
     4.4.2 Flow of Events .............................................................. 22
   4.5 Visual C++ Environment Configuration ................................. 23
   4.6 Choice of Platform .............................................................. 24
   4.7 Choice of Programming Language ....................................... 25
   4.8 System Set-Up ...................................................................... 25
   4.9 DLL Design ......................................................................... 26
     4.9.1 General Process ............................................................. 26
     4.9.2 Media Type of Filter ....................................................... 28
     4.9.3 Connect Filter ............................................................... 29
     4.9.4 Build Filter Graph ......................................................... 31
     4.9.5 Condition Transform of Filters ....................................... 33
     4.9.6 Enumerate Video Capture Devices .................................. 35
     4.9.7 Media Seeking ............................................................. 37
1. Introduction

1.1 Problem Statement

During the last two decades, with the rapid development of computer system’s software and hardware, the multimedia technology has been developed more and more quickly. In the multimedia technology, especially video streaming is becoming more and more popular because of its complexity and technology. It has been used in many areas in society and people’s lives. Most of time, video streaming means some consequent images and sounds information after been compressed and put in the Internet server, so that users can download them and watch or listen them at the same time. They don’t need to wait until they finish downloading and store the whole file into disk. This technology combines video/audio transferring and compression technology (encode and decode). Especially with the development of Internet, video streaming is widely used in the Internet. So how to handle with video streaming and make it easier and more efficient becomes a focus problem. Here, DirectShow which is from Microsoft gives us a good choice. The application of multimedia technology including surveillance, face recognition, fingerprint, eye scanning and voice verification. During all these applications, video capture occupied a very large percentage. The common character of these applications is they all use real time capture or store in the disk, and then based on the video they captured, do some applications, for example, change setting of camera, preview, grab frames, etc, so that these results may help decision-making. The common character of these applications is also the aim of this project.

This project will involve research into the different type of video devices and formats that are supported by Windows and the development of an image source class to capture video that can be used with an existing Imaging toolkit that has been developed in the School of Computing, University of Leeds. The Image toolkit currently supports video capture from cameras and a number of file formats under Linux. This project will develop a software class to provide the similar functionality under Windows. This needs to display video streaming and communicate with cameras, support different types of video files, store video to disk or real time capture for image processing. The details of requirements are summarised in the following parts of Section 2.

1.2 Problem Domain

The problem domain is divided into several smaller areas for further study:

**Video Capture:** Capture video using a video camera and support preview and capture video and store it in computer memory or to disk.
**1.3 Required Research**

The project involves a thorough understanding of two key areas, namely COM (Component Object Model) and DirectX/DirectShow technology. Because most of the API methods from DirectShow are based on the objects and interfaces of COM, so the solution of this project will therefore involve a combination of approaches from both domains. Most of the background research is related to these two areas. Further details of background research can be found in Section 3.

**1.4 Report Structure**

This report consists of six sections. Section two outlines the management issues of this project, such as objectives, minimum requirements, project plan and methodology. Section three outlines the background research of this project. It includes applications of video capture, basic concepts of API, COM and DirectX/DirectShow technologies applied during the project, etc. Section four is the most important section of this report because it outlines the architecture of the system, design, implementation and the main methodology used during this project. Section five is dedicated to the evaluation of this project. Section six has some conclusions and future work of this project.
2. Project Management

2.1 Introduction

Project management is one of the most important practices during the whole project. A strict project plan was followed though there was some changes to the original planned one. In this section, I will outline the objectives and minimum requirements of this project, further requirements, initial project plan and proposed methodology of it. For further justified project plan changes to it during the project are discussed in Section 4 and project management evaluation is in Section 5.3.

2.2 Objectives

2.2.1 Project Aim

The primary aim of this project is to use DirectShow to capture video from camera in Windows and support movie file format. During the process of design and implement, learn basic function details of DirectShow technology, such as how to use API functions and COM to implement video capture with DirectShow, how to use DirectShow to support video file formats, etc.

2.2.2 Minimum Requirements

The minimum requirements of this project are:

- **Learn how to use DirectShow.**
  
  This is the basic concept of this project and also the main concept of research study and learn, such as how to build filter, how to connect filters, how to build Filter Graph, etc.

- **Require support for video capture from one camera under windows.**
  
  This is the second of my project, after reading and study, provide a GUI interface of video capture, support basic functions of video capture, including preview and capture.

- **Provide support for different types of video, such as AVI, MPEG.**
  
  This is mixed with the minimum requirements and further requirements. In the first stage, when finish capture and store it to disk, it will be stored as AVI file. In the further requirements, it provides open file and play AVI video file.

2.2.3 Further Requirements

During the further study and discuss with supervisor, the further requirements of this project are:

- How to handle with the video, store in the disk or real time capture. Investigate real time capture to disk and camera synchronisation.
• Build DLL to support open file, open camera, grab frame and save frames to disk, all these applications.

About the details of the DLL, the requirements are as follows:
• This class should have methods to construct a new Image Source instance from a number of file formats and camera.
• Allow acquisition to disk.
• Get single frame into memory.
• Display single frame in Bitmap format.

### 2.3 Project Plan

About the project plan, here only show the initial project schedule plan. In Design and Implementation stage, there will be a revised project plan based on some requirements changes and improvements. Please see the revised project plan in Section 4.11.

<table>
<thead>
<tr>
<th>Date</th>
<th>Tasks</th>
</tr>
</thead>
</table>
| March - May| 1, meet supervisor and make the objectives and requirements specification clearly.  
2, background reading, get some basic knowledge  
3, Prepare for programming and install DirectX 9.0 SDK and Visual C++ |
| 1st June – 14th June | Start programming on Video Capture; build filter and connecting different filters together. |
| 14th June – 20th June | |
| 20th June – 5th July | Do some tests of Video Capture, support basic functions from one camera under Windows. (Encode, Decode) |
| 5th July – 17th July | |
| 17th July – 31st July | 1, Confirm requests for resources.  
2, Implement different types of video, store and play. (AVI, MPEG)  
3, Write Draft Report |
| 1st Aug – 2nd Aug | Deadline for Draft Report |
| 3rd Aug – 14th Aug | 1, Get feedback  
2, Start iteration  
3, Do some enhancements and make the functions better.  
4, Progress Meeting |
### 2.4 System Development Methodology

Because this project aims to develop a piece of software, so careful analysis of standard methodologies need to be undertaken to ensure that an appropriate methodology is been chosen. There are a number of different software development approaches. The most common ones include the Waterfall Model, Evolutionary Development Approaches such as Prototyping, Reuse-Oriented Approaches and the Spiral Model.

**Waterfall Model**

This is a methodology of a software life cycle that has been in existence for a long time. It is the most popular and famous methodology for individual projects. Normally, Waterfall Model help analysers to conceptualise, organise, and discuss the various tasks involved and identify different stages in the project so that people can carry them out in sequence.

The problems of Waterfall Model are also obviously, such as it takes too long to see the results and nothing is executable or demonstrable until code is produced; quality of product is measured by its closeness to the specification, so it maybe happen that brilliant solutions to the wrong problems; easily causes a gap of understanding between developers and users; it delays the detection of errors until the end; it has long development times, etc [quote from MSc module Business Information System Handouts].

**Prototyping**

This is a rapid development approach because software developer will build a first system from a rough and basic specification from users, just like a demo includes all basic functions of this system. After presents it to the user for comments, developer will do some improvements and refinements based on the suggestions from users. So developer will alter the system again and again, till the users feel satisfied.

This approach reduces the distance between developers and users, lets users have the opportunity to understand and learn the system from the early stages of its development, supports opportunities to test alternative designs before having to commit to a single development route. But the rapid nature of the
prototyping process often means if developers can’t understand the requirements exactly, the prototype will not be well designed, may have very poor performance at last [quote from MSc module Business Information System Handouts].

**Reuse-oriented soft development**
Reused components can either be code known by developer or be Off-The-Shelf systems that provide some required functionality. Component-based software development involves a number of stages: specification of system requirements, component analysis, requirements modification, and system design with reuse, development and integration. Reuse-oriented software development has the major advantages of cost savings, reduction of development risks, and faster delivery times [quote from MSc module Business Information System Handouts].

**Spiral Model**
This model combines the Waterfall Model and Prototyping, and is a risk-attacking strategy consisting of many small waterfall life-cycles. At the end of each cycle, a decision is taken whether to proceed or scrap the work so far and start again, hence attacking the risk.

**Rational Unified Process**
This is an object-oriented system development methodology, which is essentially based on the Spiral Model, but introduces four phases of development, namely Inception, Elaboration, Construction and Transition. It is a relatively new approach better suited for large project. Each phase usually consists of two or more iterations so for this project, two iterations should provide a solid ground for future development. Carrying out iterations allows the project to be continuously evaluated. UML (Unified Modelling Language) has become the standard in designing information systems. Because this video capture system isn’t a large project and after compare all these approaches, it will use Waterfall Model to justify and design changes and decision. For further details of justify the system development methodology are discussed in Section 4.
3. Background Research

3.1 Introduction

After describing the aim and requirements of this project in the last two sections, this section provides an overview of the literature review in the research areas that have been carried out, such as video capture application, COM, DirectX/DirectShow, and also some simple knowledge of API functions.

3.2 The need for Video Capture

Video capture have been used widely in this world, not only for videotaping, but also for analysing, such as in security, clinical care, face recognition, gesture recognition, etc. It can provide valuable insights for enhancing people’s safety through improvements, enhancing data analysis and decision-making in some special areas.

Traditional ‘Video Capture’ means converting video signals such as those captured by a video camera, into a digital format and then storing the digital video on a computer’s mass storage device. Video capture from analog devices requires a special video capture card that converts the signals into digital form and compresses the data. There are also digital video devices that can capture images and transfer them to a computer via a standard serial or parallel interface. Currently, most customers who use multimedia productions want high quality images and video streaming to provide excellent view of video, etc. So this needs high performance of video camera and software and hardware.

Applications of video capture include:

- Multimedia publication
- Video conferencing
- Internet broadcasting
- Distance learning with video
- Editing home movies with camcorders
- Video capture of clinical care to enhance patients safety
- Gesture recognition
- Machine inspection and machine vision systems
- Interactive TV
- Video capture astronomy
- Security (biometrics application)

The details of some of these applications of Video Capture will be shown in section 3.3.
3.3 The application of Video Capture

Video conferencing
Reliable video conferencing can fill the gap of face-to-face interaction and increase productivity and teamwork. Video conferencing can actually enhance business collaboration by enabling the many kinds of business and personal interactions that used to take place face-to-face: meeting, information sharing, and project development. Video conferencing requires very good streaming technology so that everyone, who participant this meeting, can see each other and monitor process of meeting quickly and smoothly, and doesn’t care about the location.

Medical care
Video capture technology is also used in medical care area. With real time observers or video recording, doctors can observe changes of patients’ illness so that they can make further decision about the treatment. Some modern hospitals use video camera system with several cameras to support simultaneous capture of vital signs of patients and do some analysis based on the video they captured, either in real time capturing or view the previous recorded video. Doctors will store the clinical data in a database and they can identify and review these data to help patients to recover quickly [3].

Security
Nowadays, anti terrorism actions are happening all over the world, especially in USA and Britain, video captures technology, image processing and video processing are becoming more and more noticeable. Most of these technologies are for automatically identifying individuals using their distinct physical characteristics, such as fingerprint, eye scanning, etc. Fingerprint identification uses a system to analyse a scanned image of the fingerprint to pick out key features, get virtual storage in a computer with the aid of laser technology. So in future when the person’s fingerprint will be scanned again, a match of print to name is verified through information systems. Such as used in eye scanning, voice verification and hand geometry. They all use biometrics technology to enhance the safety and resolve the security problem [4] [5]. Some shops sell hardware for fingerprint identification, for Internet stock trading, Internet shopping, ATM Machine Fingerprint Verification. For example, Precise 100 SC with Smart Card, which is used to store all fingerprint information instead of using a hard disk. It will help users to protect computer with combination of password and real-time fingerprint processing.

Face Recognition
Face recognition systems identify an individual by analysing the unique shape, pattern and positioning of facial features with image processing algorithm. (this algorithm is very complicated) There are
essentially two methods of processing the data: video and thermal imaging. Standard video techniques are based on the facial image captured by a video camera. But this kind of system still has its problem, for instance, if under conditions of adverse lighting, or if there is a high red content to the background of the image then the system often becomes confused, with many false detection. Also, it can’t distinguish twins or triplets, not recognize people after a haircut. So if the securities want to keep higher level of security, maybe they should use DNA identification in future.

**Machine Inspection**

Machine Inspection is another useful application of video capture, with the goal of improving quality and productivity in the manufacturing process. It is used in food processing, wood and paper, plastics, electronics or semiconductor inspection, flaw detection, assembly quality, material analysis, production and quality control, and other industries.

Machine Inspection System is PC-based, using a group of devices to receive, analyse and interpret the image of a real scene. Typically, it has an integrated camera, frame grabbers, image capture, processing storage, analysis and control system for automated inspection and measurement in production environment. In machine inspection, images should be sharper and more accurate, so system can supply better detect and discriminate between objects, remove backgrounds and perform spectral analysis.

Machine Inspection System can determine location or the position of an object, measure dimensions within thousandths-of-an-inch accuracy, count items such as pills in a bottle or cells in a dish, identify or recognize an object, inspect objects and identify flaws in manufactured goods and verify that an object’s quality meets standards. It can be used in the conditions where light levels are too low or too bright for human vision, or where no visible electromagnetic radiation such as X-rays or infrared is required. [6]

### 3.4 Application Programming Interface

Application Programming Interface (API) is under Microsoft Windows 32 bit platform. When the Windows Operating System become the priority part, program and develop the applications under Windows become essential for the programmers. But during the first few years of development of programming under Windows, the only programming tool that the programmers have is API method functions. Windows supplies API for interfaces of application programs and operating system. With API, it is just like build a building with different sizes of stones; programmers can create different interfaces, strong functions of application programs. So most of time, programmers treat API as the ‘cornerstone’ of building the whole Windows. About the architecture of Windows is shown as figure 3.1.
In general, Win32 API is separated into seven different types:

- **Manage windows**: these API functions supply methods for creating and managing users’ interfaces, we can use these functions to build interface of program.
- **General control windows**: system SHELL support some control functions, with these API functions we can control the windows and make them have different outlook. General control is supplied by COMCTL32.DLL.
- **SHELL specificity**: application programs can use these functions to enhance functions of SHELL system.
- **GDI**: Graphic Device Interface, supply graphic processing, display devices these functions.
- **International**: These API will help us to program ‘international ‘ code, supply Unicode and more than one languages.
- **Internet Service**: this allows different application programs of different computers in the Internet to communicate with each other, aim at creating and managing the connection for sharing resources.

There are some other applications uses API to support visual applications, such as Microsoft Vision Toolkit, Intel’s Vision Toolkit, Video for Windows (for example, WDM) and Real Time Image Library in Computing School, University of Leeds.

For Video for Windows application, for example, WDM (Windows Driver Model), support TV receives signal, video conferencing, multi-video streaming, output simultaneous, etc. Because DirectShow has a set of advanced architecture, it also supports WDM.

For Real Time Image Library in University of Leeds, it is for Linux Operation System and provides a simple to use common API for reading from, processing and displaying image sources including movie files, live video input, image files, API/Documentation. The API documentation has image source and manipulation functionality, image display functionality, which is based on OpenGL.


Source code for Linux is available for download from [http://www.comp.leeds.ac.uk/vision/software/libRTImage/libRTImage1.0.tar.gz](http://www.comp.leeds.ac.uk/vision/software/libRTImage/libRTImage1.0.tar.gz)
3.5 Component Object Model

Underpinning much of Microsoft’s development strategies in the last few years is the Component Object Model, COM for short. It is widely used component software model and provides the rich set of integrated services.

Generally speaking, it is very hard to overestimate the importance of COM in the Microsoft environment. COM underlies a large majority of the new code developed for Windows and Windows NT operating systems, whether created by Microsoft or by others. There are some of the facts about COM today:

- COM is in use on well over millions systems worldwide.
- COM consists of a well-defined, mature, stable, and freely available specification, as well as a reference implementation, which has been widely tested and adopted worldwide as a de-facto standard.
- COM provides the richest set of existing services for applications today, as well as the largest set of development tools available from a wide variety of vendors.
- COM supports thousands of available applications, including all of today’s highest volume applications. Major system vendors such as Digital Equipment Corporation and Silicon Graphics have announced plans to ship COM on UNIX and other operating systems, with more vendor commitments on the way.
- COM is supported by the largest number of development tools available for any component or object model on the market.

COM is a mechanism for identifying and communicating between component applications in a generic manner. COM communication is done via interfaces or groups of functions implemented by COM objects. A group of functions is defined (including all parameters and return values) and identified by a unique 128-bit interface ID (a globally-unique id or GUID). A COM object is created by a call to a Win32 API (CoCreateInstance), and what is returned is a pointer to a specific interface. The interface will be returned as a pointer to a table of function pointers, pointing to the functions defined in this interface. This is based on the C++ abstract class: like an abstract base class, the interface definition defines a layout for the table of function pointers, which is the v-table, and derived classes (the COM objects) provide implementations. One object can implement many different interfaces: each interface includes the QueryInterface method that lets you find another interface on the same object.

Most parts of API in DirectShow are based on the objects and interfaces of COM. COM are a standard of interfaces. Many API of DirectShow are created as instances of COM Object. An object just like a black box, objects communicate with each other via interfaces. Programmer sends orders to objects or receives orders from objects via COM interfaces. These sorts of orders are called methods. For instance,
This line of code is to display the position of windows via IVideoWindow method. If get the COM object of one object, and also know the interfaces supported by this object, then the application program or other objects will be able to make sure the service of the first object. All COM objects inherited one method called Query; it will show the interfaces and points for create these interfaces.

### 3.5.1 IUnknown Interface

All the COM interfaces are derived from an interface called IUnknown. This interface supplies DirectShow for the control of object’s living times and operates multi-interfaces. It has three methods:

- **AddRef**: When an interface or another application is added to an object, use AddRef method to increase the index data of this object.
- **QueryInterface**: check objects via pointer that points to specific interfaces support the characters.
- **Release**: decrease the index data of object. When index data equals to 0, the object will be released from memory.

From these three methods, AddRef and Release methods are responsible for maintaining the index of objects. For example, if build an IGraphBuilder object, the index data of this object is enactive to be 1. Every time one method receives return value of this object, this method will use AddRef method add 1 to the index data of the object via return pointer. Every AddRef method has a Release method. When the index data equals to 0, this object will be released and all its interfaces can’t be used anymore. QueryInterface is used to check whether this object support the specific interfaces. If so, QueryInterface will return a value to the pointer of that interface. Then programmer can use the methods of that interface to communicate with object. If QueryInterface return the pointer successfully, it will automatically use AddRef method to increase the index data.

### 3.5.2 DirectShow COM Interface

This project is using DirectX/DirectShow to implement and if want to use DirectShow technology, people should know something about COM. To be honest, it is not necessary to know all about COM, because DirectShow application is a client program of COM, the only problem is how to use the COM interfaces, including how to create COM components, how to get interfaces methods and how to manage COM, etc. All the interfaces of objects of DirectShow are from IUnknown interface. The creation of basic objects is handled by special methods of DLL.

The following code is the basic procedures of COM:
3.6 DirectX/DirectShow

This project uses DirectShow technology, which is developed from DirectX Media, one part of DirectX 6.0, to support multimedia application.

3.6.1 DirectX

DirectX is one of the most popular applications of Window 98 and Windows NT. DirectX is a programming environment for multimedia application programs and hardware enhancement. At first, Microsoft designed it for adapting different formats of multimedia. It is part of Microsoft SDK for programmers who develop the higher-class computer game and cartoon. Then DirectX became part of Windows 98 that means it is part of Operating System now. The first purpose for Microsoft to develop DirectX is to improve the facility of design and implement computer games in Windows environment. Before DirectX, most of the PC games are designed based on MS-DOS; game programmers have to program different code for different hardware interfaces. Though the codes are nearly the same, but they have to do a lot of same workload. With DirectX, PC game programmers can be independent of hardware and access hardware directly. It resolves the problem of identifying new hardware in PC. Every DirectX component is the total of API that can be used by users. With these interfaces supplied by DirectX, programmers just need to use independent of hardware and don’t need to think about the details of them. Via DirectX, application programs can use Hardware Accelerator (HA for short), if
HA doesn’t work, DirectX can simulate HA so that it can supply very powerful multimedia environment. DirectX has become a standard. The establishment of this standard supplies the strategy for hardware development. Hardware factory authorities have to develop productions based on this standard, the developers can enjoy the high performance of hardware via the interfaces that are supplied by DirectX and don’t need to care about execution details of hardware. A computer system can be separated into four layers:

- **Hardware/Internet layer**: multimedia devices, including hardware accelerator, sound card, input devices or devices for communication in the network, etc.
- **Basic layer of DirectX**: supply multimedia service for image, sound and other devices.
- **Multimedia layer of DirectX**: supply API functions for cartoon, audio or video applications.
- **Component layer**: include control and application of ActiveX; it uses the advantages of DirectX API functions to supply multimedia services for customers.

In the third layer, it has Direct3D, DirectAnimation, Direct Model, DirectPlay and DirectShow. DirectShow is used for designing computer games, and is developed from Active Movie 2.0, which is one component of DirectX. DirectShow is an API method or an ActiveX control component. It is used to help developers to transfer high quality audio or video signals in the network. It supports most popular media types, for instance AVI, MIDI, MPEG, and QuickTime. For more details of DirectShow, please read the following part. In DirectX SDK 9.0 DirectShow supports whole schema for handling different types of media files replay, video capture, audio captures or some other highly required for multimedia applications.

### 3.6.2 DirectShow

Why we need DirectShow? What will DirectShow do for us? There are some challenges the programmers should face to during the development of multimedia application.

- Multimedia data has very huge volume; developer needs to make sure the efficiency of handling with them.
- How to make audio and video synchronism.
- How to use simple approach to handle complex media source, including local file, computer network, broadcast, camera, or some other digital production.
- How to handle all kinds of media types, such as AVI, ASF, MPEG, DV, MOV, etc.
- How to support un-predictable hardware.

The main purpose of designing DirectShow is to set programmers free from complex data transformation, hardware differences, synchronism, etc. DirectShow will finish the main overall architecture and basic lower layer work, so under this condition, programmers don’t need to think about the details and design and implementation the multimedia application programs become easier. DirectShow support capture, play, replay multimedia streaming powerfully, so programmers can get
frame data from sources and do some further application. DirectShow has many input filters and output filters and uses one component named Filter Graph to connect all these filters together. The main architecture of DirectShow Filter Graph is shown in figure 3.2. DirectShow supports media types widely, including ASF, AVI, MPEG, DV, MP3, WAVE, etc, this made replay function of multimedia data very easily. In addition, DirectShow combine some technologies of DirectX, such as DirectDraw, DirectSound, and support DEC of DVD, also transfer data with digital camcorder. DirectShow supply a very open programming environment so that programmers can design their own components based on specific requirements.

Figure 3.2 shows the architecture of DirectShow Filter Graph. [10]

From this diagram, DirectShow allocates in the middle layer and application programs are in the upper position. Application will build relative Filter Graph with clear purpose. Then use a model named Filter Graph Manager, which is defined by DirectShow to manage the whole process of data streaming. Application programs use Filter Graph Manager to store and transfer data. About the default value, Filter Graph Manager will handle process of data streaming automatically. For example, input proper CODEC automatically, connect output pin of Transform Filter to default value filter automatically, etc. Filter Graph Manager use some components to handle with the data, these components are called Filter. Filter is about COM object. Every filter is connected together by orders in Filter Graph, named Input Pin and Output Pin. Every filter has at least one pin. There are three types of Filters: Source Filters, Transform Filters, and Rendering Filters. Source Filters are responsible for getting data; the data source could be files, capture card of computer in the Internet (drive by WDM or VFW), camera or digital camcorder. Then the data will be transfer to next filter. Transform Filters are responsible for format transform for data and also transfer data, for instance decode/encode, RGB24 to I420, etc. Then data will be transferred to next filter. Rendering Filters are responsible for the destination of data, to disk, or sound card, or display card. Filters control streaming via read/write to files, modify data and display data to output devices. In order to finish this mission, all filters should be connected as a
system. There is one thing should be noted, these three parts are not only one filter to finish the functions, sometimes each part has many filters to work together.

Pins are used to connect filters and all the Input Pins and Out Pins will negotiate to support which kind type of media. The upstream filter’s output must be the input of the following downstream filter and each filter should have at least one Input Pin. Figure 3.3 shows the connection of Filters via Pins. [10]

![Figure 3.3 Connections of Filters](image)

The main basic application of DirectShow is play a media file. The following figure 3.4 is an example of Filter Graph of playing an .AVI file.

![Figure 3.4 Filter Graph of Playing Local AVI File](image)

In this Filter Graph, File Source (Async) is Source Filters, it is used to manage the specific media file in disk and supply data for AVI Splitter. AVI Splitter and AVI Decompressor are Transform Filters. AVI Splitter is used to ask for data from File Source and then separate audio and video, send them through Output Pins. AVI Decompressor is used to decode video. Video Renderer and Default DirectSound Device both are belonged to Renderer Filters. Video Renderer is responsible for output image to windows and Default DirectSound Device is responsible for play sound synchronism.

Some knowledge of Filters is important, including Connect Filter Pins, Transfer Data Via Pins and Media Seeking. All of these important applications content are used in this project, so here only outline some general approach and thinking. For the design and implementation details, please read Section 4.

- Connect Filter Pin:
Connect Filter Pin, in fact it is a process of negotiating the Media Type between pins. This connection is always start from upstream filter’s output pin, end at downstream filter’s input pin. Figure 3.5 shows the connection approach between two filters.

The general process of connecting filters is: enumerate all the media types on the input pins, use all these types to connect output pin. If output pin accept this type of media, then the connection between these two filters are successful. If output pin doesn’t support all the media types that input pins enumerate, it will enumerate all the media types on output pin, and use all these types to connect input pin. If input pin accept one type of these types, then the connection between these two filters are also successful. If output pin doesn’t support all the media types on input pin, then the connection will be failed.

After connect pins, that means after they negotiate a media type that is accepted by both sides, it should be ready for transferring data. Before transferring data, the most important work is Allocator Negotiation. Most of time, this negotiation is started from output pin. Between DirectShow Filter, data is transferred as sample. Sample has an internal memory to load data, and the Allocator organizes sample. Because a pair of pins, which have been connected already, has the same allocator, so they don’t need internal memory to copy data when transfer data from output pin to input pin. When get data from input pin in one filter, programmers can do some applications and then fill data into output pin, the data will be then transferred to the next input pin.

- Transfer Data Via Pins:
Filter has two types of data transformation model, one is Push Model, (see figure 3.6) and the other one is Pull Model (see figure 3.7) [11].
Push Model, it means Filter (Source Filter) can generate data, and its output pin can send the data out. The following filter after Source Filter must has an interface named IMemInputPin, and when data is transferred, the upstream filter will use the Receive method of this interface.

![Figure 3.7 Pull Model](image)

Pull Model means Source Filter doesn’t have the ability to send data out. Most of time, a Parser Filter or a Splitter Filter follows it, because this kind of filter has independent output pin to get data from Source Filter, send them after applications. Typically, if Source Filter is file source, Pull Model is the better choice. The output pin of Source Filter in Push Model has an interface named IAsyncReader, the following Splitter Filter use Request method or SyncRead method of this interface to get data. Then Splitter Filter will use the downstream input pin’s Receive method of IMemInput interface to transfer data to next filter.

- Media Seeking:
  In this project, the requirement of DLL need seek media, which means when user plays video file, he can know the duration of the file and seek some frames to do some application. So here outline general knowledge about how to realise this in the level of Filter.

  Filter Graph Manager is used to control and organize the whole Filter Graph. Via Filter Graph Manager, an interface named IMediaSeeking is used to seek the media streaming. Filter Graph Manager will acquire from the last Filter, which is Renderer Filter, ask whether upstream filter’s output pin accepts IMediaSeeking interface or not. If so, return this interface; if not, acquire the up-upstream filter, till Source Filter. Most of time, output pin of Source Filter support IMediaSeeking interface. It will tell users the duration, current position and some other information of this video file in total. If the source is file in disk, Parser Filter or Splitter Filter will support this interface. If writing code of Source Filter, it is necessary to realise IMediaSeeking interface on the output pin of filter; if writing code of Transform Filter, send the require for getting interface method on current output pin to output pin of upstream filter; if writing code of Renderer Filter, send the require for getting interface method on current filter to output pin of upstream filter.

  In conclusion, Pin has the responsibility to support interfaces to connect other pins and transfer data. Pin’s interfaces support use shared internal memory or other sources to transfer data, negotiate types of data in the pin-to-pin connection and copy data at a lowest level but get a large volume of data.
4. Design and Implementation

4.1 Introduction

Following the basic introduction of Video Capture and DirectX/DirectShow technology in Section 3, this section presents the technology used in building the Video Capture System and show the main diagrams of the system, such as architecture diagram, activity diagrams, etc, and also has the details of design and implementation of this project during programming.

4.2 Case Study

Before giving details of analysis and design process, two existing example cases are described. These two examples are: DirectX Video Processing AppWizard that is developed by Yunqiang (Charles) Chen, (chenyq@ifp.uiuc.edu); the other one is an audio/video capture example named AMCAP under DirectX 9.0 SDK. These two examples were really useful during project, enhanced author’s understanding and gave some good suggestions about the main procedures of video capture and its related applications.

4.2.1 DirectX Video Processing AppWizard

This AppWizard is under environment of DirectX SDK 8.0 or later one. It creates a default application to process the video data. It includes the opening, seeking, playing an existing video file, setting and previewing a connected camera, capturing video sequences. It also supports the grabbing of the image frames. User could apply some special effect (XOR) to the media data. To generate an application, user could just create a project using this ‘DirectX Video Processing AppWizard’ and change the code according user’s own task.

If user wants to grab each image frames, this generated application will use an ISampleGrabber Filter to grab each frame. User can also do his special processing on the image frames in the frame callback function, the method is MyAppDoc::SampleCB(), and the results will be shown on the screen. To show how to do this, this generated project has a small segment of code in frame callback function to XOR the image before it is displayed. So the video shown on the screen is XORed. User also can add his own filters into the graph to process the video instead of ISampleGrabber.

After the application project is generated, user can add his own processing code to process the image data, such as apply some special effect, analysis the data, etc. If user checks to use the ISampleGrabber, he could implement own processing code in the frame callback function (i.e. CMyAppDoc::SampleCB(), which will be called on each frame in the sequence). There is a sample code there to XOR the image frames.
User also can build other kinds of Special Filter Graph. Just override some functions in CMyAppDoc (derived from CDxVideoGraphBase). For example:

- addSrcFilter() to include different source filter, such as bmp sequences.
- addExtraFilter() to include extra filters to process the video, such as colour converter.
- addRenderFilter() to do a special rendering, such as NULL render.
- And some other functions if the graph is significantly different from the default one, such as createFilterGraph(), ReleaseFilterGraph(), etc.

This application also has some toolbars on the interface, such as live preview, capture video to a file, open an existing video and play it, some operation buttons, adjust camera properties and scale the video to fit the whole window.

During study this example there is a problem of it. The property page of setting the preview resolution cannot be displayed. User has to choose the WhiteBalance button to set up the preview resolution and make it the default set up. Then user has to click the Preview button to activate the changes. This is because of the difference in the VFW driver and WDM drivers. Some USB cameras work very well, but do not support VFW camera very well.

### 4.2.2 Audio/Video Capture Example ‘AMCAP’

DirectShow SDK provides this example and it allocated ‘C:\DX90SDK\Samples\C++\DirectShow\Capture\AMCap’. It was a good example to help a lot to do this video capture project.

This example shows the following knowledge:

- Win32 application programming, such as create window dialog box, event loop and response for events, etc.
- Use Capture Graph Builder to help me build filter graph for capturing. From SampleCGB.cpp file, class ISampleCaptureGraphBuilder has a component of Capture Graph Builder, and is created when initialise ISampleCaptureGraphBuilder class.
- Set the output format of video. AMCAP example shows how to change the video capture rate in BuildCaptureGraph method and BuildPreviewGraph method.
- Support USB interface.
- Register Filter Graph to ROT, so that GraphEdit.exe can check filter graph. In this example, AddGraphToRot method and RemoveGraphFromRot method are responsible for register Filter Graph to ROT or remove from ROT.
- Enumerate all video capture devices in the system.
- Support video capture to the specific video capture devices and support AVI format.
4.3 System Architecture

Figure 4.1 displays the overall architecture of this video capture system from a high view. But there is one thing that should be noted here, that is Image Processing and the Analysis part is not included in this project. The reason author put this part in this system is because it is the further necessary research and application and also is one part of the whole video capture system, but this project only includes the GUI and DLL library elements.

![System Architecture Diagram](image)

Figure 4.1 System Architecture

4.4 Utilising UML

UML is a short name of Unified Modelling Language, which is a visual modelling language providing a set of diagrammatic tools [10] used widely in the analysis of information systems. It is a set of different diagram types that can be used alone or together to describe all or part of system. Each diagram type can be used to describe a system from a particular perspective. The project used UML to illustrate and justify any design decision.

4.4.1 System Use-Case Diagram

Figure 4.2 shows the users in the video capture system.
The main user of this video capture system is visualization research group staff. With the library this project supports, visualization research group staff could do further application, for example to capture a picture of a stop watch and then exact each frame or a fast moving object and see if each frame is extracted to show constant movement, etc. This diagram shows the main function of this system, such as real time preview, real time capture, open file, play/pause/stop video file and process. Here the process function support two functions, they are grab frames and store image. Grab frames means this system supports grab specific frames the user wants, for example, if user wants to see the number 150 frame of this video file (the duration of this video file should be 150 frames at least), just input 150 in the dialog box and the system will display the number 150 frame from the video capture device in this system. Store image mean if user is interested in single frame when playing the video file, the system supports the function to pause the video file, save the image of the single frame as .bmp file in the disk.

4.4.2 Flow of Events

Figure 4.3 is the Activity Diagram of this video capture system.
4.5 Visual C++ Environment Configuration

Because DirectShow SDK supply a set of C basic source code and these codes reduce the complexity of programming Filter. So first, we should configure Visual C++ environment as follows:

- Make sure the basic source code that has been complied can generate a library. (Normally, Debug will generate Strmbasd.lib, Release will generate Strmbase.lib)
- Use the guide of VC to generate a common WIN32 library as Filter Project, so choose File/New/Projects, and then choose ‘Win32 Dynamic-Link Library’.
- The source code should have StdAfx.h file.
- Normally, use _stdcall method to negotiate. Choose Project/Settings/CC++, in the ‘Category’ dialog box choose ‘Code generation’, and then choose ‘_stdcall’ in ‘Calling convention’.
Configure necessary library-connect file. Choose Project/Settings/Link, and then in the ‘Category’ dialog box choose General, input the following code in Object/Library modules:

We can set a method in VC so that we don’t need to use .def file to bring DLL out or definite entry method of DLL. Choose Project/Setting/Link, in the ‘Category’ dialog box choose ‘Output’, and then input ‘DllEntryPoint@12’ in ‘Entry-point symbol’ box.

In addition, it is necessary to configure DirectX SDK’s Include directory and Lib directory to the system directory of VC, so that the compile machine can get the latest source files. The way to do this is: Open Visual C++, choose Tools/Options/Directories, in the ‘Show directories for’ dialog box choose ‘Include files’, and then input as follows (my DirectX SDK is installed in C:\DX90SDK, VC is installed in

```
C:\Program Files\Microsoft Visual Studio):  
C:\DX90SDK\Include  
C:\DX90SDK\SAMPLES\C++\DIRECTSHOW \BASECLASSES  
C:\DX90SDK\Samples\C++\Common\INCLUDE  
C:\Program Files\Microsoft Visual Studio\VC98\Include  
C:\Program Files\Microsoft Visual Studio\VC98\MFC  
C:\Program Files\Microsoft Visual Studio\VC98\ATL  
Then choose ‘Library files’ in ‘Show directories for’, input:  
C:\DX90SDK\Lib  
C:\DX90SDK\Samples\C++\DirectShow\BaseClasses\Debug  
C:\DX90SDK\Samples\C++\DirectShow\BaseClasses\Release  
C:\Program Files\Microsoft Visual Studio\VC98\Lib  
C:\Program Files\Microsoft Visual Studio\VC98\MFC\Lib
```

### 4.6 Choice of Platform

Because this project is to develop a video capture system include a library in Windows, Windows 2000 is chosen to do this project. Windows 2000 has better support for hardware devices such as cameras and video capture card, but of course some existing issues can still affect the overall performance. A Real Time Image Library for Linux is already available from the computing school website.
4.7 Choice of Programming Language

There are two object-oriented languages that can be used during the development of this project, they are Java and C++. A brief comparison is presented as below:

Object oriented programming is compulsory in Java. Java provides large class library enabling software to be developed quickly. Executing a Java program requires a run-time environment to be present so the program can be interpreted. The result is cross-platform independence at the cost of performance [11].

C++, built on top of C language, bridges the high-performance optimised program with strong object oriented ideals [2]. C++ programs generally run faster than programs written in Java. Performance is important in this project as it may be used in real time capture application in the future. The Real Time Image Library provided in School of Computing is written in C++ so all software development of this project also use C++.

4.8 System Set Up

The software of this project is developed on personal laptop. The configuration of laptop is:

- X86-based PC @ 800+Mhz
- 64+MB RAM
- 20+GB of Free HD space
- DirectX 9.0 SDK: This is a set of lower-level application programming interfaces (API) for creating games and other high-performance multimedia applications. It includes support for two-dimensional and three-dimensional graphics, sound effects and music, input devices, and networked application such as multiplayer games. In Direct9.0 SDK, it has some new techniques, such as Video Mixing Render Filter9 and new encoder API Specification, and also some new interfaces. It also supports a test tool – Graph Edit, which can display the running and workflow of streaming. During this project, most of the programming is based on the filter graph displayed by this test tool. [9]
- The main camera used in this project is Siemens Maxell MaxPen Camera, but also use Philips PCVC 750K Camera for test and evaluation.

These are the minimum requirements to perform reliable video capture. It is entirely possible to do video capture with lower configuration, but good results cannot be guaranteed. Obviously, a faster CPU, more RAM, and more HD space are a good thing. Windows 98/ME has one problem, which is the FAT32 file system has a limitation preventing files from being larger than 4 GB, so prefer using a Windows 2000/XP machine since the NTFS file system has no such file size limitation.
4.9 DLL Design

4.9.1 General Process

A typical general process of writing code of DirectShow application is shown in figure 4.4.

![Figure 4.4 Process of Design with DirectShow](image)

From this figure, general process of design video capture project with DirectShow. It include several steps:

1) Create a component of Filter Graph Manager, the code is:

```cpp
IGraphBuilder *pGraph = NULL;
HRESULT hr = CoCreateInstance(CLSID_FilterGraph, NULL,
CLSCTX_INPROC_SERVER, IID_IgraphBuilder, (void **) &pGraph);
```

2) Build a completed Filter Graph based on the requirements. For example, if want to play a AVI file from local disk, the simple and fast way is:

```cpp
hr = pGraph ->RenderFile(L"C:\Example.avi", NULL);
```

3) Use the interfaces on the Filter Graph Manager to control and finish the interaction between Filter Graph Manager and applications. For example, use IMediaControl interface to control condition transform of Filter Graph, the code is:

```cpp
IMediaControl *pControl = NULL;
hr = pGraph ->QueryInterface(IID_IMediaControl, (void **) &pControl);
hr = pControl ->Run(); // Run Filter Graph
```
In the second step, before build a Filter Graph, use GraphEdit.exe to improve that the Filter Graph is reasonable and correct. Then start programming based on the Filter Graph model displayed in GraphEdit.exe.

In conclusion, the main process of programming is:

- Using API method `CoCreateInstance()` to create an instance of Filter Graph;
- Using API method `QueryInterface()` to get points of Filter Graph and IMediaEvent component;
- Controlling Filter Graph and be response for events.

In Windows 2000 GraphEdit.exe also supports the functions that display the Filter Graph built by other applications so that if the Filter Graph doesn’t work immediately programmer could check how the filter connect each other in the program. So after build Filter Graph Manager, register Filter Graph to Running Object Table (ROT) so that GraphEdit.exe can work. The details of this code is shown in program named CImage::AddGraphToRot().

After this, Filter Graph will be displayed in the windows box of GraphEdit.exe. Figure 4.5 shows the dialog box of GraphEdit.exe.

![Figure 4.5 Dialog box of GraphEdit.exe](image)

Before close the application program, it is necessary to close this GraphEdit program. The approach is to remove the previous registration from ROT. The code is as follows:

```cpp
void CImage::RemoveGraphFromRot(DWORD dwRegister)
{
    CComPtr<IRunningObjectTable> pROT;
    if (SUCCEEDED(GetRunningObjectTable(0, &pROT))) {
        pROT->Revoke(dwRegister);
    }
}
```
4.9.2 Media Type of Filter

Filter is used to handle some types of streaming. In DirectShow, it defines ‘Media Type’ to determine what kind of type of media streaming will be used.

Media Type is actually a data structure AM_MEDIA_TYPE that is defined by DirectShow, its code is:

```c
typedef struct _MediaType {
    GUID majortype;
    GUID subtype;
    BOOL bFixedSizeSamples;
    BOOL bTemporalCompression;
    ULONG ISampleSize;
    GUID formattype;
    IUnknown *pUnk;
    ULONG cbFormat;
    /* [size_is]*/ BYTE *pbFormat;
} AM_MEDIA_TYPE;
```

This code shows that in DirectShow use three parts to determine the media type, they are: majortype, subtype and formattype. These three parts have their own GUID to identify them. Majortype is used to choose whether this is a video (MEDIATYPE_Video) or audio (MEDIATYPE_Audio) or stream (MEDIATYPE_Stream) file, or some other types; Subtype is used to assist majortype and specific which type, for example, if majortype is video, subtype could specific this is RGB24 (MEDIASUBTYPE_RGB24) or RGB32 (MEDIASUBTYPE_RGB32) or YUY2 (MEDIASUBTYPE_YUY2), etc. If major is audio, subtype could specific this is PCM format (MEDIASUBTYPE_PCM) or some other formats; Formattype is used to specific information of the format in details, include size of video image, frame rate, etc.

This project uses the following media type in the code:

```c
ZeroMemory(&mt, sizeof(AM_MEDIA_TYPE));
mt.majortype = MEDIATYPE_Video; //video is majortype (digital)
mt.subtype = MEDIASUBTYPE_RGB24;//RGB24 is subtype
mt.formattype = FORMAT_VideoInfo; /*specific general video information, use VIDEOINFOHEADER data structure.*/
```
4.9.3 Connect Filter

The connection between filters is in fact a connection between Pins in Filter. As mentioned in section 3.6.2 about filter’s pin, this connection is a negotiation of media type between filters. This process is very important and also is the most complicated part during this project. Every pin has IPin interface and pin uses this interface to connect other pins. Figure 4.6 shows the process of Pin connection.

The whole process need several methods to support, they are basic class method CBasePin::Connect(), CBasePin::AgreeMediaType(), CBasePin::TryMediaTypes(), CBasePin::AttemptConnection(), CBaseOutputPin::CompleteConnect(), and CBaseOutputPin::DecideAllocator().

- **CBasePin::Connect()**
  In this method, it first gives the optional media type, and then checks whether this pin has been connected, include pointer of this pin, size of it, auto lock condition of it and display the information of this pin. If this pin has been connected, the dialog box will show ‘Already Connected’. Because Filter can only be connected when it is stopped, so then check whether this filter has been stopped. After it has checked this, Filter will check the media type between output
pin and input pin. The dialog box will show the text of either ‘Connection Succeeded’ or ‘Failed to agree type’. This method doesn’t really support real connection of pins. This is the first step and only checks parameter’s condition. The real connection process should check all the other methods.

- **CBasePin::AgreeMediaType()**
  This method first checks the validation of the specific media type. It uses a pointer named pmt (pointer of MediaType) to check whether pmt (pointer of optional media type) is a totally specific media type. If it is, then use this type to use AttemptConnection() method to try to connect directly. The connection will stop no matter whether this connection will be succeed or failed. But if pmt is a NULL pointer or not a totally specific media type, the really negotiation process is starting now. First, enumerate all the media types that are supported by the pin. Here, the ‘for loop’ time is 2, and the initial value of m_bTryMyTypesFirst from output pin is false, which equals to 0. This means it will get the attempt connection from media type to enumerate device from input pin. If this fails, it will try to attempt another connection from output pin. If it also fails (invalid or type not accepted), it will return failure.

- **CBasePin::TryMediaTypes()**
  This method is used to use all the media types that have been enumerated from Pins to attempt connection (use AttemptConnection() method). During the enumeration process, if one media type succeeds, then the whole connection process is successful; if all types failed, TryMediaType() method will return an error value.

- **CBasePin::AttemptConnection()**
  In this method, first use CheckConnect() method to check connection of output pin, for example, check whether the pin which has been connected support specific special interface, etc. If CheckConnect() failed, use BreadConnect() method. If succeed, use CheckMediaType() method from output pin to check media types for connection. This will display type information of pins. If CheckMediaType() also succeed, store points of input pin and current media type in the objects of output pin and then use ReceiveConnection() method from input pin to check input pin. If ReceiveConnection() succeed, the last action is to use CompleteConnect() method from output pin.

Up to this point, the whole negotiation of media type used in pins is finished. But this only means filters have found a media type that both side can accept it, but still can’t transfer data between filters, for example, how to allocate the internal memory to transfer data? How to manage memory? All these types of activities work should be done by CompleteConnect() method.
CBaseOutputPin::CompleteConnect()

The source code of this method is:

```cpp
HRESULT CBaseOutputPin::CompleteConnect (IPin *pReceivePin)
{
    UNREFERENCED_PARAMETER (pReceivePin);
    return DecideAllocator (m_pInputPin, &m_pAllocator);
}
```

In this code, DecideAllocator () method is used to complete negotiation for allocator for transfer data between pins. In DirectShow, the unit for transferring data is called Sample (Sample is also a COM and manage a data memory). Sample is managed by Allocator (also a COM). The Pin connect each other must use the same Allocator but which pin should create this Allocator also need negotiation. The next method shows this process.

CBaseOutputPin::DecideAllocator()

In this method, first it asks the requirements of Allocator on input pin. If input pin has created Allocator, get this Allocator and determine the size of memory of each Sample, number of Sample (complete in DecideBufferSize () method on output pin). If the process mentioned all succeed, the negotiation of Allocator is completed. But if it failed, output pin will create an Allocator and after use DecideBufferSize () method, use NotifyAllocator () method of input pin to tell which Allocator will be choosed.

After this point, the whole connection of Pin is completed.

4.9.4 Build Filter Graph

After complete connection of Pin, the next step is to build the Filter Graph for own application. The way to build Filter Graph is shown as follows:

- IFilterGraph::AddFilter: this parameter provides a Filter object, add it in the Filter Graph.
- IFilterGraph::ConnectDirect: this parameter provides output pin, input pin and media type so that can connect directly.
- IGraphBuilder::AddSourceFilter: this parameter provides source file’s name and add a Source Filter into Filter Graph automatically.
- IGraphBuilder::Connect: this parameter provides the connection of output pin and input pin. If the connection failed, it will attempt to insert some necessary format transform filters between these two filters automatically.
• IGraphBuilder::Render: this parameter provides output pin, add necessary filters to complete rest of the process of building Filter Graph, till connect to Rendering Filter.

• IGraphBuilder::RenderFile: this parameter provides source file’s name, add necessary filters to complete the Filter Graph for playing file.

During these ways, the last four has the automatically function. In DirectShow this automatically function is called Intelligent Connect. For example, in this project code the CImage::OpenFile() method use Intelligent Connect to create Filter Graph supports Grab Frames. The code is shown as follows:

```cpp
JIF(m_pGraphBuilder->RenderFile(m_wcOpenFile, NULL)); // Intelligent Connect
JIF(m_pGraphBuilder->QueryInterface(IID_IFilterGraph, (void**)&m_pFilterGraph));
JIF(m_pGraphBuilder->QueryInterface(IID_IMediaControl, (void**)&m_pMediaControl));
JIF(m_pGraphBuilder->QueryInterface(IID_IVideoWindow, (void**)&m_pVideoWindow));
JIF(m_pGraphBuilder->QueryInterface(IID_IMediaEvent, (void**)&m_pMediaEventEx));
```

In Intelligent Connect, IGraphBuilder::AddSourceFilter is used to give a source file path so that Filter Graph Manager can provide a Source Filter for this source file. This is a process of analysis and check registration table ‘HKEY_CLASSES_ROOT’. IGraphBuilder::RenderFile gives a file name, uses the same algorithm to find correct Source Filter. Then start from each output pin of this Source Filter to complete this Intelligent Connect. IGraphBuilder::Render is also used for complete Intelligent Connect, but starts from some specific output pin in the current Filter Graph. IGraphBuilder::Connect is used to check a pair of output pin and input pin.

These are the general ways to build Filter Graph, but if the programmer wants to add a video effect filter during playing video or changes media type of streaming when it is running in Source Filter, programmer should add new Decode Filter, or add another video streaming in the Filter Graph, under these conditions, it is necessary to do some improvements of the Filter Graph. Normal ways is to stop Filter Graph first, do some improvements and changes, and then restart Filter Graph. But under the following conditions, Filter Graph doesn’t need to be stopped:

• Only change media types for Filter Connection.
• Add or Remove Filter, connect related Filters again.
Do some applications of a Filter Chain. Filter Chain is a Filter Graph that has some filters connect each other and each filter has at most one input pin that has been connected, at least one output pin that has been connected. For example, in figure 4.7, A-B, C-D, F-G-H, F-G, G-H are all Filter Chain.

![Figure 4.7 Filter Chain](image)

After building the Filter Graph, it is also necessary to release this Filter Graph after finish using and stop it. In this project the following code are used to release it:

```cpp
void CImage::ReleaseFilterGraph()
{
    // Destroy the whole filter graph
    StopPreviewOrCapture();
    SafeRelease( m_pMediaControl );
    SafeRelease( m_pMediaSeeking );
    SafeRelease( m_pSrcFilter );
    SafeRelease( m_pRenderFilter );
    SafeRelease( m_pVideoWindow  );
    SafeRelease( m_pFilterGraph  );
    SafeRelease( m_pGraphBuilder );
    SafeRelease( m_pBuilder );
    SafeRelease( m_pGrabberBaseFilter );
    SafeRelease( m_pSampleGrabber );
    SafeRelease( m_pMediaEventEx );
}
```

### 4.9.5 Condition Transform of Filters

Filter has three conditions: Stopped, Paused and Running. Paused means the data is ready now and it is a necessary condition between Stopped and Running. In the code of this Video Capture System, it
uses IMediaControl interface from Filter Graph Manager to control the condition transform of Filter Graph. IMediaControl interface has several interfaces, such as Run, Pause and Stop, etc. Actually, Filter Graph Manager also uses IMediaFilter::Run, IMediaFilter::Pause and IMediaFilter::Stop interfaces from all Filters in Filter Graph.

If Source Filter is working under Push Model (not Live Source). When Filter Graph transform the condition from Stopped to Paused, Source Filter starts data thread, and use IMemInput::Receive method from input pin of Transform Filter and transfer data as Sample. When Transform Filter receives Sample, it will transfer Sample to Rendering Filter in the data thread of Source Filter. Now, Rendering Filter is under Paused condition, when it receives the first Sample, it will block data thread. If it is Video Renderer, it will show the Sample frames in the video windows.

When Filter Graph transforms the condition from Paused to Running, Rendering Filter only need to un-block data thread and let whole Filter Graph running again. Video Renderer will display the Sample frames blocked under Paused condition, every time it receives Sample, it will show the Samples based on different time.

Filter Graph Manager start condition transform from Rendering Filter, it goes along the Filter Graph till Source Filter.

- **Stopped -> Paused:** Start Paused Condition Transform from Rendering Filter. Filter use the Active method from all its pins to initialise (allocate Sample, start data thread of Source Filter). Rendering Filter is ready now. Source Filter is the last one finishes and be ready. When Source Filter start data thread and transfer data to following filters. When Rendering Filter receives the first Sample and then blocks it. So now finish Condition Transform. When all Rendering Filters in this Filter Graph receive the first Sample after Paused, it means the transform really finished.

- **Paused -> Stopped:** When Filter is under Stopped condition, it will use Inactive method from all its pins (Release Sample, stop data thread of Source Filter). All Filters should release all the blocked Samples, un-blocked IMemAllocator::GetBuffer from upstreaming, stop all waiting in Receive method so that get return value of Receive method of upstreaming. Under Stop condition, Filter won’t receive any Samples. So when it reaches Source Filter all the data thread will be stopped.

The following code shows function of play, pause and stop in this project:
void CIImage::Play()
{
    if (m_pMediaControl)
        m_pMediaControl->Run();
}

void CIImage::Pause()
{
    if (m_pMediaControl) {
        m_pMediaControl->Pause();
        m_pMediaControl->StopWhenReady();
    }
}

void CIImage::Stop()
{
    if (m_pMediaControl) {
        m_pMediaControl->Stop();
        m_pMediaControl->StopWhenReady();
        StopPreviewOrCapture();
    }
}

4.9.6 Enumerate Video Capture Devices

This DLL library uses the advantage of DirectShow that it supports unknown hardware to enumerate video capture devices. In the interface, the video capture devices that are used in this system will be shown in a small dialog box with number and name.

Figure 4.8 shows the main process of enumerate devices.
From this general diagram, the main processes of enumerate devices include:

1) Use CoCreateInstance method to create object of system device enumerator (CLSID is CLSID_System-DeviceEnum), and get the ICreateDevEnum interface.

2) Use ICreateDevEnum::CreateClassEnumerator to create a enumerator for Filter registration category, and get IEnumMoniker interface.

3) Use IEnumMoniker interface enumerate all devices’ moniker under specific category.

4) Use IMoniker::BindToStorage interface can visit attribution of device moniker, such as Display Name, Friendly Name, etc.

5) Use IMoniker::BindToObject interface to create a DirectShow Filter with device moniker, and then use IFilterGraph::AddFilter and add it to Filter Graph so that it works in this Graph.

In the code, CImage::GetNumCapDev() method is used to get the number of video capture device and CImage::GetDevName() method is used to get the number of video capture device, and in this method, it is necessary to change narrow char to wide char because in DirectShow all the methods support wide char but the normal system or VC6.0 support narrow char. For details of the code please read these two methods. From device moniker, it will provide two names: Display Name and Friendly Name. The previous one can be got from IMoniker::GetDisplayName() method, it is a complicated string with CLSID and doesn’t show to user. The latter one is a string that can be understood by user easily and it will be shown in the dialog box. The enumerate device function in video capture system’s interface is shown in figure 4.9 as follows:
4.9.7 Media Seeking

The DLL library provides some similar functions named Grab frames; Get next frame and Skip frames. All these are related to media seeking problem, which means that a user can seek the video file and choose which frame he is interested and display this frame. For the details of media seeking in code, it will be discussed in section 4.10 and Section 5.

The main process for seeking media is done on Filter, because applications get IMediaSeeking() interface from Filter Graph Manager to realise seeking streaming. Figure 4.10 shows the process to get IMediaSeeking interface.

When applications get IMediaSeeking interfaces from Filter Graph Manager, Filter Graph Manager will start this operation from Renderer Filter first. But normally Renderer Filer can’t execute the seeking operation, so it passes the seeking requirement to output pin of upstream Transform Filter. This process is to get the pointer of output pin of Transform Filter from input pin of Renderer Filter, and then use QueryInterface method to get IMediaSeeking interface on this output pin. If Transform Filter can complete this seeking operation, then this operation will be executed in this filter and return value. If not, Transform Filter will pass this require to the output pin of upstream, till the last one Source Filter.

Because this video capture system requires data from file in disk and also from real time capture, it uses both Pull Model and Push Model, and the seeking operation in these different model has different approaches. In fact, under Push Model, output pin of Source Filter can support seeking operation;
under Pull Model, output pin of Parser Filter or Splitter Filter can support seeking operation. In this
project, only use output pin of Splitter Filter to support this function under Pull Model.
If it needs to get data from real time capture, which means choose Push Model, these are still some
different conditions. If write Source Filter, it should realise IMediaSeeking interface on the output pin
of this Source Filter; If write Transform Filter, only need to send the seeking operation on the output
pin to the input pin of the upstream; If write Renderer Filter, that need to send the seeking operation
requirement to the output pin of upstream because Renderer Filter doesn’t have output pin. In addition,
there is one problem after seeking operation, which is synchronization of streaming. So if the filter has
IMediaSeeking interface, has more than one output pin, one of them must support seeking operation.
Application can do seeking operation to Filter Graph at anytime, whenever Filter Graph is stopping or
pausing or running. But when Filter Graph is running, Filter Graph Manager will pause Filter Graph,
after it completes the seeking operation Filter Graph will re-running again.
IMediaSeeking interface supports some types of seeking operation, include:
  o TIME_FORMAT_FRAME: this is seeking as frame;
  o TIME_FORMAT_SAMPLE: this is seeking as Sample;
  o TIME_FORMAT_FIELD: this is seeking as field;
  o TIME_FORMAT_BYTE: this is seeking as setoff in the byte streaming;
  o TIME_FORMAT_MEDIA_TIME: this is seeking as time and 100ns is the smallest unit.
During all these types of seeking operation, TIME_FORMAT_TIME is the basic type and also the
default type. The Filter that supports IMediaSeeking interface can support several types of seeking
operation at the same time, but not all the types. In this video capture system, it chose
TIME_FORMAT_FRAME as seeking operation. Before using some types of seeking operation except
TIME_FORMAT_TIME, it is better to use IMediaSeeking::IsFormatSupported to confirm it.

This system uses CI mage::Seek () method to provide this seeking function. It has three ways to
confirm the position of the frame in the media file, seek to the start position, seek to the current
position and seek to the end position. Figure 4.11 shows the result of seeking and grabbing single
frame 201 and 202. Then user can compare them and find difference between these two frames and
help him to make decision.
4.9.8 Grab Frame and store as .bmp file

In this project, grab frame is a useful function during previewing and capturing. This function is if user is interested in the current frame, he can grab this frame and store it to the disk as a .bmp image file. Normally there are two ways to grab frame, the first one is using IBasicVideo::GetCurrentImage() method, the other way is using Sample Grabber Filter.

- **IBasicVideo::GetCurrentImage()**
  This method is very simple, firstly it acquires the size of the bitmap image, and then allocate the buffer size based on the size of the bitmap image and get the data of the image frame. The last steps are to write bitmap file header and bitmap information header and then delete buffer and return the image.

  In this method, IBasicVideo interface should be acquired from Filter Graph Manager, but it is really realised on Renderer Filter. If Renderer Filter is normal Renderer Filter, it is not reliable to use GetCurrent Image to grab frame, because this method will failed when using DirectDraw to accelerate it and this method must be paused when using it. So using IBasicVideo::GetCurrentImage() method has some restrictions.

- **Sample Grabber Filter**
  This way is more complicated than the first one. It uses Sample Grabber Filter, which is a Trans-In-Place Filter. Sample Grabber can grab all types of Samples and this project chose this method to support Grab Frame function.

  This method has some steps to support grab frame, include:

  1) Create Sample Grabber and add it to Filter Graph, the basic code is as follows:
2) Set the media types on the pins for Sample Grabber

In this system, it wants to grab RGB24 image so the code is as follows:

```c
AM_MEDIA_TYPE mt;
ZeroMemory(&mt, sizeof(AM_MEDIA_TYPE));
mt.majortype = MEDIATYPE_Video;
mt.subtype = MEDIASUBTYPE_RGB24;
mt.formattype = FORMAT_VideoInfo;
JIF(m_pSampleGrabber->SetMediaType(&mt));
```

For the media type details, please read section 4.9.2 Media Type of Filter.

3) Complete creation of Filter Graph

Because Sample Grabber has set a media type, but other Filters must use this media type to connect Sample Grabber, this system uses Intelligent Connect approach to simplify the creation of Filter Graph. Basic code is shown as follows:

```c
JIF(m_pGraphBuilder->RenderFile(m_wcOpenFile,NULL));// intelligent connect
JIF(m_pGraphBuilder->QueryInterface(IID_IFilterGraph,(void**)&m_pFilterGraph));
JIF(m_pGraphBuilder->QueryInterface(IID_IMediaControl,(void**)&m_pMediaControl));
JIF(m_pGraphBuilder->QueryInterface(IID_IVideoWindow, (void**)&m_pVideoWindow));
JIF(m_pGraphBuilder->QueryInterface(IID_IMediaEvent, (void**)&m_pMediaEventEx));
```
In this code, it only supports Grab Frame and no video preview, then there should be a NULL Renderer Filter following after Sample Grabber and its CLSID is CLSID_NullRenderer. It is not a good idea to connect normal Video Renderer after Sample Grabber because Sample Grabber is a Trans-In-Place Filter, which means its input pin and output pin have the same buffer memory, and this memory maybe allocate on the display card. It is obviously that to read data from display card buffer memory is much slower than from computer memory and this will reduce the quality of video image.

4) Run Filter Graph
Sample Grabber has two work models, Buffer Model and Return Model. Buffer Model means after putting Sample into buffer memory it then transfers Sample to the next filter. Return Model means when there is an input Sample, it will use application to set the return method of Sample. In this system, it chose Buffer Model because Return Model will affect the efficiency of the whole Filter Graph and easily cause a lock problem. Also added a method namely ISampleGrabber::SetOneShot, which is used to stop Filter Graph after Sample Grabber gets a Sample. The code is shown below:

```c
// Set One-shot mode and buffering.
hr = pGrabber->SetOneShot (TRUE);
hr = pGragger->SetBufferSamples (TRUE);
pControl->Run(); // Run the Graph.
PEvent->WaitForCompletion (INFINITE, &evCode); // Wait till it is done.
```

5) Get data of Sample
Under Buffer Model, this system used ISampleGrabber::GetCurrentBuffer method to get Sample data. The unit is byte. It also used another method named GetCurentFreamSize to combine with GetCurrentBuffer to support this. They are shown as follows:

```c
// find the required buffer size
long cbBuffer = 0;
hr = pGrabber->GetCurrentBuffer (&cbBuffer, NULL);
char &pBuffer = new char [cbBuffer];
if (!pBuffer)
{
    // Out of Memory. Return an Error code.
}
hr = pGrabber->GetCurrentBuffer (&cbBuffer, (long*) pBuffer);
```
Another method named GetCurrentFrame() is also used to help these two methods to finish grab frames function. The details of these three methods are shown in the next section.

### 4.10 Project Plan

This is a revised project plan based on some requirements changes and improvements during design and implementation.

<table>
<thead>
<tr>
<th>Date</th>
<th>Tasks</th>
</tr>
</thead>
</table>
| March – May        | 1, meet supervisor and make the objectives and requirements specification clearly.  
                     | 2, background reading, get some basic knowledge.  
                     | 3, Prepare for programming and install DirectX 9.0 SDK and Visual C++. |
| May – June         | Prepare for exams.                                                    |
| 1st June – 20th June| Start programming on Video Capture; build filter and connecting different filters together. |
| 21st June – 1st July| 1, Finish Video Capture in Windows, including Preview, Capture and Play these three functions. Support AVI file. Do some tests of Video Capture.  
                     | 2, Write main architecture of report and show it to supervisor and do some improvements. |
| 5th July – 17th July| 1, Get further requirements of this project, which is create a DLL library to support Open file, Open camera, Grab frames, Get next frame and Save frames. Start programming of these new requirements.  
                     | 2, Start writing introduction section and background reading section. |
| 18th July – 2nd Aug | 1, Confirm requests for resources.  
                     | 2, Keep programming, finish enumerate video capture devices, include name and number, and finish preview and capture and control camera, for example, open camera, set configuration of camera, media type and dialog box for displaying.  
                     | 3, Finish Draft Report and show it to supervisor. |
| 3rd Aug – 20th Aug  | 1, Start programming Grab frames function, including seeing, get current frame size and Get current frame these three parts. |
| 21st Aug – 31st Aug | 1, Finish DLL design and implementation (further requirements).  
2, Get feed back from Supervisor and Assessor. Start evaluation and improvement of the DLL.  
3, Finish demo of project report except Evaluation part. |
| 1st Sep – 7th Sep | 1, Finish evaluations of the video capture system.  
2, Finish Project Report  
3, Finish Feedback Form |
| 8th Sep – 9th Sep | 1, Submit .pdf report  
2, Write provisional feedback/assessment forms |

### 4.11 Methodology

In section 2.4 ‘System Development Methodology’, discussed some basic and popular software development approaches. This project used Waterfall Model to justify any design changes and decisions. The reason for this is Waterfall Model has become a standard of information system development. During design and implementation of this individual project, it also has several various stages, such as GUI part, only provides preview and capture functions; DLL design stages, provides several small stages, including open file, grab frames, etc. This is similar to Waterfall Model methodology. Though Waterfall Model has its disadvantages, but this project plan added risk attacking at the end of each stage so that this can make sure that the video capture system should be good enough and match the requirements of this project. With this risk attacking, it is not necessary to do large improvement of this system or make a good solution to a wrong understanding of problems in the end and prevent failure of this project.

During the design and implementation of this project, Waterfall Model was used to justify several stages and produce the final deliverables. They are: preview stage; capture stage and DLL design stage. They will be present in details with using Waterfall Model methodology as follows and also these parts include necessary revision of the project plan.

In the preview stage, firstly this system used SIEMENS Maxell Maxpen Camera to support this function. This camera is not a high-class camera and the data it gets does not need to be decompressed. It used Intelligent Connect to build the Filter Graph that is provided by DirectShow. This preview function works very well with Intelligent Connect approach. But based on the methodology this project has chosen, at the end of this stage, it needs to be tested to prevent risks at the end. After
changing the camera, another camera was used. The PHILIPS PCVC 750K Camera doesn’t support preview function. Figure 4.12 shows the error when using PHILIPS camera.

![Error Message](image)

**Figure 4.12 Error with PHILIPS Camera when previewing**

After check code carefully and getting suggestions from supervisor, it was clear that the code needed to determine the Media format automatically rather than setting it in the complied code. In previous code, it used IAMStreamConfig interface but this needed to be done before render the graph and not after render the graph. Because the data this PHILIPS camera gets is decompressed and is different from the SIEMENS one. This is also the same problem occurred when using this camera to do capture function.

Figure 4.13 shows the error dialog box of this camera.

![Error Message](image)

**Figure 4.13 Error with PHILIPS Camera when capturing**

The connection method is in CreateFilterGraph(), the code is as follows:

```c
JIF(m_pBuilder->RenderStream(&PIN_CATEGORY_PREVIEW, &MEDIATYPE_Video, m_pSrcFilter, m_pGrabberBaseFilter, NULL));
```

This is the Intelligent Connect of improvements based on the suggestions from supervisor, now this video capture system can support all kind of USB camera now. This really resolves a big problem because if this problem is not resolved, some methods in the DLL won’t work and will influence other functions.

At the end of DLL design and implementation stage, supervisor gave some suggestions about improvements and also some suggestions are from some other visualization research group staffs, such as DLL should be independent of the windows, the program is trying to do too many things and should be broken down into a number of other methods, etc. All these improvements’ details will be discussed in the Section 5, the Evaluation part.
From the experiences of using Waterfall Model, it was a wise choice to choose life-cycle methodology as this project’s methodology.
5. Evaluation

5.1 Introduction

After design and implementation of this video capture system, this section will present evaluation of this project. Evaluation is the final and critical phase of the project, it helps to determine whether this project has matched the objective and minimum requirements or not and whether it needs further improvements in the design stage or implementation stage. With evaluation phase, it will help programmer to know which part is good enough, which part still need further improvement and which part is useless. This will also help further user to know the advantage of this project and disadvantage of it.

In this section, evaluation is got from supervisor’s feed back and also from some other research group staff. Evaluation phase is separated into two parts, one is function and performance evaluation, another one is project management evaluation.

5.2 Function and Performance Evaluation

Normally, current software development evaluation has some basic criteria. So function and performance evaluation of this project also use some of these criteria to justify this solution, include:

- Usability
- Integrity
- Availability
- Unity
- Reversibility
- Efficiency
- Reliability
- Maintainability

5.2.1 Usability

Usability related to the ability of the software to meet the requirements of the user or at least meet the minimum requirements. This also reviews whether the functions of this system are good enough so that user can get the information he expects from the system for further application. This video capture system has met the requirements of the project, including minimum requirements and further enhancement requirements. The usability of this system is OK.
5.2.2 Integrity
Integrity means whether this system has all the necessary codes, documents, functions and presentation. So that it can be understood and easily operated by user. This video capture system provides good integrity, it has DLL document, supports functions that this project needs and also has a presentation interface so that use can see what it looks like and he also can use the DLL directly without interface. From the feedback, the system code should have some sort of installation instructions, for instance, what direction should the DLL library go in? Does further user need to re-compile the code, etc. In the DLL documents, a list of all methods along with their parameters and descriptions of what they do or return should be shown. The integrity of this system needs improvements.

5.2.3 Availability
Availability means whether this system provides many types of media file, so that the user can choose different applications, different aims and different areas to choose the proper media. This video capture system provides all the media files, including AVI, MPEG, Windows Media, etc when the play video file function is running. When the capture function is running, this system only supports the storing of video as AVI file automatically. This is because of requirements of this project and time limiting. Also this system provides different types of video source, include from disk or real time capture. So under this condition, the availability is appropriate, but can be further improved.

5.2.4 Unity
Unity means the buttons have same functions or similar component buttons should be in the same position or close to each other. And the same operation should have the same result. This video capture system has a simple interface because of limits of time. But in the simple interface, it also has the similar functions buttons together, for instance, it has File menu, Edit menu, Help menu and Test menu. In the test menu, it puts preview, capture and stop these three functions together, and put open file, play file, pause file and stop file these four functions together, and put the two examples which are grab the first 20 frames and retrieve frame 456 from an MPEG file together. So the unity of this system is not perfect, but it is adequate.

5.2.5 Reversibility
Reversibility means the system should provide exit function, and allow user to play backward and exit from this system at anytime. This video capture system has exit button in the File menu so user can exit it at anytime. But in the Test menu, it doesn’t support play backward function because this is not necessary based on the requirements. If the user needs perfect functions, this should need improvement. So the reversibility is not good.
5.2.6 Efficiency
Efficiency means whether user is able to get the data and information he wanted with few clicks and quickly. This video capture system did very well in this area because each button has its own function and provides few clicks to get the information the user needed. But one problem is if capture 20 frames of MPEG to disk, the system takes a little bit long time. It should be less than a second. This is not a problem in this case but for real time capture, does this capture the number of frames further user expects? For example, if user wanted to capture 100 seconds of video, does he get 2500 frames in MPEG movie and are these frames all different? From test of grabbing 20 frames, the first 12 frames are nearly the same and the other 8 frames are similar. This maybe the further work.

5.2.7 Reliability
Reliability means whether the user gets the correct information after clicking a particular button on the interface of the software. After test, this video capture system can provide correct information to user, such as correct frame image. But it has a problem with code of the system, which is a couple of methods do not have meaningful names, such as Fream and Turation. Both of them are spelling mistakes. So reliability of this system needs improvements.

5.2.8 Maintainability
Software system needs to be flexible to meet the future requirements and applications. This video capture is built for further vision application so it should have good maintainability. For example, it provides grab frame function, but if use wants to grab more than one frame, just do a loop method in the code, and how many frames user wants, how many loops this code should provide. For example, if two frames are needed, frame 38 and frame 56, the only changes needed is to add the following code in the Seek () method:

```
int a[2]={38,56};
for (int i=0; i<2; i++) {
    seek (a[i], ture);
    GetCurrentFream();
}
```

Also if user wants to change the setting of camera, just set camera before preview or capture then it is fine. Research staff can choose which method is required in the DLL library and use it without others and also can add functions to the DLL library easily. The problem is it should have more comments in the DLL code explaining what is going on if further user was to consider the code maintainable.
5.3 Project Management Evaluation

The project management is shown in Section 2 and during this project, the management of project has been good. All stages followed the project schedule and plan but still have some problems and changes. With the changes of requirements the project became more and more complicated, it was necessary to change the schedule to fit the whole project. Because in the early stage of this project, it has clear objectives and minimum requirements, so this project didn’t have many changes during various stages. This condition made the project management not that complicated and difficult. The meeting log of every week and the individual log of each stage of learning and programming are all kept very well. The final software has code, DLL documents and explanations of the code, and it has finished minimum and enhancement requirements. Finally, the delivery of software product and report matched the project schedule and are completed and submitted on time.
6. Conclusion

6.1 Conclusion of this project

In this project, the author has successfully developed a video capture system and also developed a DLL library for Windows. They are able to fulfil the following objectives and requirements (include minimum requirements and enhancement):

- Use DirectShow to require support for video capture from one camera under windows.
- Provide support for different types of video, such as AVI, MPEG.
- Support both store video to disk and real time capture.
- Build DLL to support open file, open camera, set frame rate, grab frame and save frames to disk.
- This class should have methods to construct a new Image Source instance from a number of file formats and camera.
- Allow acquisition to disk, for example grab first 20 frames and retrieve frame 456 from an MPEG file.
- Get single frame into memory and also store single frame in Bitmap format to disk.
- Visualization research group staff can use this DLL independently in Windows for further application.

6.2 Future Work

This video capture system and DLL library produced can be viewed as a platform as future visualization research and development for tracking moving objects. During this project, because of time limiting, there is still something that can be done in future and they are outlined in this section as follows:

- Need to do some improvements based on evaluation section.
- Need to add other functions, for example, capture a picture of a stopwatch and then extract each frame, or a fast moving object and see if each frame is extracted to show constant movement.
- Need to set configuration of camera when preview or capture. This video capture system provides some configurations of camera, for instance, camera rate, size of video and position of video, etc. These configurations can only be set before start preview or capture.
- Need to design a better graphical user-interface (GUI) to enhance the Human-Computer Interaction (HCI) aspect of the application. Currently, graphical GUI is not necessary for this project because the primary aim is to use DLL library but if it is used in a software products then a graphical user-interface is essential.
Need to provide better accuracy and quality of frames. Such as capture frames very quickly, and each frame from movie should be all different, etc.
Reference


Appendix A

Personal Reflections

After I finished this project and looked back from the time I completed its objectives and requirements, I found that I have learnt a lot of skills, especially programming skills and writing skills. Before I start doing this project, I only have basic knowledge of C++ programming and have little exposure to the tools like VC++ 6.0. My biggest challenge was to learn how to use this tool and program with C++, compile and set debug for code. During the whole project my experience of more advanced concepts in C++ have improved a lot and I was successful in learning and using VC++6.0. This project also helped me in learning skills like testing, executing and software evaluation. Another important thing I learnt is the need for good project management; with it I can control what I have learnt, what I have done and what I should do next.

The modules I have learnt before, like Object Oriented Programming, Business Information Systems, Information System Engineering and Future Directions, their handouts and coursework helped me in understanding the principles of programming and how to design and implementation a system.

How to achieve a balance between background reading, design and implementation is another challenge task for me. Because background reading gives me a lot of information about video capture and image processing, but some of them are not related to this project very much. During design and implementation stage, it explored some more requirements of this project, which are not known when I made the objectives and requirements. So I had to do some relevant changes but I continually remind myself to follow the basic route of aims and objectives when I felt I need to try new ideas. At the end stage of this project, I found that everything is clear and goes well.

After this project, my advices for international students are two key elements in project success: get exactly the correct requirements and good project management. For international student whose mother language is not English, it is very important to get the correct requirements at the early stage of project because if get the wrong requirements or misunderstanding what supervisor mean, the excellent solution for wrong problem will happen and this project may be fail. After this stage, the next element is having good project management. Because unexpected things may happen during project and may affect project like illness and personal problems. I have my own personal problems during my project and really affect my feeling and enthusiasm. After I got suggestions from Lydia and my supervisor Dr. Andy, I managed this problem and keep my concentration on this project. So have a good project plan and management is very important.
It is important to note that the problem has not been fully solved and many things can be done for further research. I am pleased that students in years to come to extend the project to further application.

Appendix B
Objectives and Deliverables Form
School of Computing, University of Leeds

**MSc Project Objectives and Deliverables**

This form must be completed by the student, with the agreement of the supervisor of each project, and submitted to the MSc project co-ordinator (Mrs A. Roberts) via CSO by 18th March 2004. A copy should be given to the supervisor and a copy retained by the student. Amendments to the agreed objectives and deliverables may be made by agreement between the student and the supervisor during the project. Any such revision should be noted on this form. At the end of the project, a copy of this form must be included in the Project Report as an Appendix.

**Student:**

Yun Teag

**Programme of Study:**

MSc of Information System

**Supervisor:**

Dr. Andy. J. Bulpitt

**Title of project:**

Video Capture in Microsoft Windows

**External Organisation**: 

* (if applicable)

---

**AGREED MARKING SCHEME**

<table>
<thead>
<tr>
<th>Understand the Problem</th>
<th>Produce a Solution *</th>
<th>Evaluation</th>
<th>Write-Up</th>
<th>Appendix A</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>40</td>
<td>20</td>
<td>15</td>
<td>5</td>
<td>100</td>
</tr>
</tbody>
</table>

* This category includes Professionalism

---

**OVERALL OBJECTIVES:**

Using DirectShow to capture video from camera in Windows and support movie file formats.

**MINIMUM REQUIREMENTS:**

1. Learn about video file formats and the form different types of camera interfaced to the PC.
2. Learn basic functions of DirectShow.
3. Require support for video capture under Windows, from one camera.
4. Provide support for different types of cameras.
5. Think about how to do with the video (to disk or real time capture for image processing).

*Continued over*
SOFTWARE AND HARDWARE RESOURCES REQUIRED:
1. DirectShow
2. Using C++ language
3. PC with Windows Operation System.

DELIVERABLE(s):
1. A project report.
2. A simple interface (software) that can support functions.

Signature of student:  
Signature of supervisor:

Date:  
Date:
Appendix C

Mid Project Report

School of Computing, University of Leeds

MSC MID PROJECT REPORT

All MSc students must submit an interim report on their project to the MSc project coordinator (Mrs A. Roberts) via the CSO by 9am Wednesday 28th April 2004. Note that it may require two or three iterations to agree a suitable report with your supervisor, so you should let him/her have an initial draft well in advance of the deadline. The report should be a maximum of 10 pages long and be attached to this header sheet. It should include:

- the overall aim of the project
- the objectives of the project
- the minimum requirements of the project and further enhancements
- a list of deliverables
- resources required
- project schedule and progress report
- proposed research methods
- a draft chapter on the literature review and/or an evaluation of tools/techniques
- the WWW document link for the project log to date

The report will be commented upon both by the supervisor and the assessor in order to provide you with feedback on your approach and progress so far.

*The submission of this Mid Project Report is a pre-requisite for proceeding to the main phase of the project.*

<table>
<thead>
<tr>
<th>Student:</th>
<th>Yun Teng</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programme of Study:</td>
<td>MSc of Information System</td>
</tr>
<tr>
<td>Title of project:</td>
<td>Video Capture in Microsoft Windows</td>
</tr>
<tr>
<td>Supervisor:</td>
<td>Dr. Andy. J. Bulpitt</td>
</tr>
<tr>
<td>External Company (if appropriate):</td>
<td></td>
</tr>
</tbody>
</table>

** AGREED MARKING SCHEME **

<table>
<thead>
<tr>
<th>Understand the problem</th>
<th>Produce a solution *</th>
<th>Evaluation</th>
<th>Write-up</th>
<th>Appendix A</th>
<th>TOTAL %</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>40</td>
<td>20</td>
<td>15</td>
<td>5</td>
<td>100</td>
</tr>
</tbody>
</table>

* This includes professionalism

Signature of student: Date:

*Supervisor’s and Assessor’s comments overleaf*
Supervisor's comments on the Interim Report

The maximum requirements should be expanded to include some of the challenges you have identified.

Real-time capture of video is more challenging...

Have you considered any other approaches with Y?

The deliverables require more thought. Perhaps there should be a library of VideoSource methods for video such as...
Assessor's comments on the Interim Report

Objective and minimum requirements require such details and explanation. For example:
- provide support for different types of videos – please add the kind type of “support” and how many different types of videos found?
- will some capture be supported?

Would you provide a log file or compile for easy integration with your result?

Are you planning to query any other relevant/similar technologies for video capture with Windows, besides Direct Show?

Deliverables: “A simple interface of software can implement all the features” – please check and edit.

Various related technologies mentioned in this report. Have you found any other related background research, of yourselves, design, etc. ?

For the reference style, please read the style guide website.

Great work. I look forward to seeing the outcome.

K.
Appendix D

Interface of Software

In this appendix, I will show some interfaces of this video capture system that I have discussed in the report.

Figure Appendix 1 Interface of Preview

Figure Appendix 2 Interface of Capture
Figure Appendix 3 Interface of Pause

Figure Appendix 4 Interface of Process and store as .bmp file
Appendix E

Process of Example AMCAP

In this appendix, I will outline the running process of AMCAP, one of the example cases for my project and write some understanding after it. About the code, please read:
C:\DX90SDK\Samples\C++\DirectShow\Capture\AMCap\amcap.dsw

In this code, the process is:
1, WinMain: the entrance of Win32 program is WinMain (1 means the first layer of method and the outer layer)
2, AppInit: initialise this application
3, CoInitializeEx: initialise COM
3, DbgInitialise: initialise debug library
3, LoadAccelerators: load accelerator menu
3, RegisterClass: register window class
3, GetStockObject: get object defined by system
3, GetDC: get device context
3, SelectObject: select one object
3, GetTextMetrics: get the metrics information of text
3, ReleaseDC: release device context
3, CreateWindowEx: create window
3, statusInit: initialise status dialog box
4, statusCreateTools: create status dialog box
5, CreateSolidBrush: create a kind of brush
5, CreateFont: create font
5, GetDC: get device context
5, SelectObject: select one object
5, GetTextMetrics: get metrics information of text
5, SelectObject: select one object
5, ReleaseDC: release device context
4, RegisterClass: register window class
3, CreateWindowEx: create window
3, ShowWindow: show the window as some formats
3, GetProfileString: get string data from win.ini
3, ZeroMemory: clear one memory to zero
3, AddDevicesToMenu: add the capture devices in this system to the menu
4, GetSubMenu: get the handle of submenu
4,GetMenuItemCount: get the number of count of menu item
4,RemoveMenu: remove menu
4,IMonRelease: use the number of COM interface, if number is 0, then release the memory
4,CoCreateInstance: create a solid format object for initialising
4,CreateClassEnumerator: create an enumerator for an object
4,Reset: reset a IenumMoniker
4,BindToStorage: get a pointer to storage interface
4,Read: read the name of device
4,AppendMenu: add a menu
4,CheckMenuItem: check the item of menu
4,EnableMenuItem: check whether menu is available to work
4,SysFreeString: release BSTR format string
4,AddRef: add enumeration value for interface
3,GetProfileInt: read an integer value from win.ini
3,ChooseDevices: choose capture device
4,CreateBindCtx: provide a pointer for running IBindCtx object
4,MkParseDisplayName: change a cstring value to moniker
4,ChooseDevices: choose capture device, but the parameter is different from the former one. The former one finishes its most important works in changing methods
5,FreeCapFilters: release capture filters
6,SAFE_RELEASE: complete release work
5,InitCapFilters: initialise new capture filter
6,MakeBuilder: create builder class
7,ISampleCaptureGraphBuilder: apply a new solid format COM
6,BindToStorage: get a pointer for binding
6,Read: read name of capture device
6,MakeGraph: create class of filter graph
7,CoCreateInstance: create a solid format object
6,SetFiltergraph: create filter graph
6,AddFilter: add filter to Filter Graph
6,FindInterface: find interface of object
6,GetFormat: get the audio/video format
6,DeleteMediaType: delete useless audio/video format
6,FindInterface: fine an interface
6,EnumPins: enumerate pins
6,Next: get the next pin in this filter
6,QueryPinInfo: get information of pin
5. MakeMenuOptions: change information of menu
4. IMonRelease: release moniker
3. GetModuleHandle: get the handle of loading file
3. GetProcAddress: get interface address of method in DLL
3. SetAppCaption: set name of window
2. PeekMessage: get information
2. TranslateAccelerator: check keystroke information
2. TranslateMessage: translate message
2. DispatchMessage: send message to window
2. WaitMessage: wait for message
2. CoUninitialize: close COM library
1. Finish application
Appendix F

DLL Document

This appendix is to give explanations to video capture system code, include all the methods have been used and their functions. In the file that named ‘work’, it has two folders and also each has one project, which is named MyImageProcess and TestMyImageProcess. In this documents will introduce the general details of these code and the further details, please read the code carefully.

This system has two projects, the first one is DLL, named MyImageProcess.dll; the second one is TestMyImageProcess project, it is used for the interfaces for presentation. For the Human Computer Interface, to be honest, based on the limitation of time, it only focussed on completing the main requirements of this project and didn’t explore this problem. If the DLL can support enough interfaces for user to use, the user who will use my DLL should think about HCI and do some improvements.

MyImageProcess project has a main class named CImage, which is the most important of my project. All the mathematics functions are using the methods of this class. The definition of this class is definite in MyImageProcess.h. When it is initialise this class, also when it creates the instance of this class, it will enumerate all available video capture devices, (just the cameras), and the drives of these devices should support DirectShow. During the enumerating, it will check the number and name of devices. The number of devices will be taken from class CImage::GetNumCapDev () and the name of devices will be taken from the class CImage::GetDevName ( char *name, int index). Here, when index is 0, it represents the first device’s name. For the details of these classes, please read code of CImage::CImage(). When user knows the information of video capture devices, he can choose device to do preview or capture after he know the number of them. (Only when index > 0, it means there is device in this system).

● Preview Function

One class named CImage::PreviewLive(HWND hWnd, RECT *pRect, int CapID, bool fSetup) is used to realise the functions of preview. The meaning of each parameter is: hWnd represents the window for preview, if it is NULL, it will be displayed in the screen of computer. pRect represents the size of preview window, here user can set the size of the window. CapID means which device will user use in this system, because this system will need some video capture devices, so user can choose which camera to preview, if this parameter is 0, it means the first camera, if it is 1, it means the second camera, etc. fSetup means whether to set the parameters of video capture devices or not. It includes the number of frames per second, the format of image, size of image, etc. With this parameter, the user can change the setting of camera by hand or by inputting code, so that he can control the camera. In
the code, TRUE means user need to set the camera, FALSE means he uses the setting that are automatically set and doesn’t need to control the information of camera. About the procedures of this mathematics function, figure Appendix 5 shows it.

![Figure Appendix 5 Process of Preview](image)

This is the filter graph for preview. In the code, it built the filter of camera first, which is source filter. Then add Sample Grabber Filter; the responsibility of this filter is to give user the information of a single frame. Basically, it is possible to get all the frames’ information, just use GetCurrentFrame() in the project code. After Sample Grabber Filter, it is Smart Tee Filter, which is used to separate single into two parts, one is preview, and another is capture. Under the condition of not destroy data, it just like copy and won’t influence on the rate of preview or capture. The last filter is Video Renderer Filter, it is used to display all the frames on the screen, and realize the function of preview. About the code details, please read MyImageProcess.cpp file, CImage::PreviewLive(HWND hWnd, RECT *pRect, int CapID, bool fSetup) class.

- **Capture Function:**

In the code, the CImage::CaptureToFile (TCHAR *vname, HWND hWnd, RECT *pRect, int CapID, bool fSetup) realize the functions of capture. About the parameters, vname means the full path of one file, which is user will capture. It includes the path of this file, so does its name. hWnd mean when user is capturing, the window will display the preview image. All the others parameters are the same meaning as in preview part. The procedures of this function please see figure Appendix 6.

![Figure Appendix 6 Process of Capture](image)

Compare with the preview filter graph, the output of Smart Tee Filter is not only preview, but also has capture. Mux Filter is used to transform the video format into AVI format and write the name of the file, finish the capture function. So, here, except preview graph, users still have capture graph. For the
details of this function, please read the CI mage:: CaptureToFile (TCHAR *vname, HWND hWnd, RECT *pRect, int CapID, bool fSetup)

When finish preview and capture, use StopPreviewOrCapture () to stop the filter graph and release all the filters.

- **Get frame function:**

The class GetCurrentFreamSize () and GetCurrentFream (long BufferSize, char *pBuffer, int *iWidth, int *iHeight) are used to realize the functions of capture current frame. The first method is used to get the duration of the data of the frame. With this, user will know how size of buffer should be applied for, and give the address pBuffer to GetCurrentFream, and then get the data of this frame. In addition, user also can get the width and height of the data.

- **Play the video file:**

Here also means the Go back function. Replay function is not necessary because if the user doesn’t want to watch the video file anymore, just press ‘stop’ and stop this file. If user wants to see it again, just open the file again and play it again. If he wants to see one part again, he can use the ‘get frame’ functions that is below the open, stop, pause buttons. In the code, it used OpenFile (TCHAR *vname, HWND hWnd) to do the play video file function. If user uses this method successfully, it will open the file, and then user can choose to play, to pause or to stop. All these three functions, please read CI mage::Play(), CI mage::Pause () and CI mage:: Stop (). Figure Appendix 7 shows the filter graph of play a file.

![Filter Graph](image)

**Figure Appendix 7 Process of Playing Video File**

In this filter graph, user will get frame data from Capture.avi, separate audio and video via AVI Splitter. If the video file has no audio stream, the filter graph is the same of this one, only video stream. After AVI Splitter, go into Grabber, it is the same of the Grabber in preview filter graph, for getting the data of frame. Then is Color Space Converter, which is used to transform different format of data, for instance from I420 to RGB24, etc. The last one is Video Renderer, for displaying data of each frame on the screen so that we can see the video file.

In this OpenFile method, it also support some other interfaces, they are: GetFileTuration (), for getting the total duration of the video file.
Seek (long offset, bool start=true). This method is used for jump to given position frame. Offset is the
distance of frame; start means the offset is start from original start position or from current position.
TRUE means from start position 0, and FALSE means from current position.
SeekToStart () means jump to start of this video file.
SeekToEnd() means jump to end of this video file.

With Waterfall Model this project has chosen, after got feedback and suggestions from supervisor and
assessor, these are some improvements of the DLL design. All these improvements are made only for
made the software better for further application. The feedback has four main problems of this video
capture system and the improvements are shown as follows:

- DLL independent

The revised program is independent of the windows and also has a test program for presentation. The
new interface doesn’t have set-up functions in the dialog box but it has the following functions
mentioned in the feedback document, such as open a camera, set up the camera frame rate, show how
to grab the first 20 frames, show how to retrieve frame 456 from an MPG. For open a camera function
is shown in OpenCamera () method. Set up the camera frame rate is shown in SetCameraRate ()
method. Camera rate is defined as: if ((rate>0)&&(rate<=100)) in the code because rate cannot be
minus and cannot be bigger than 100 because the common camera’s rate cannot be high like this. The
camera rate is chosen bigger than zero and smaller than 100 is enough for this project. The other grab
frames functions are shown in the ChildView.cpp of TestDll.dsw file. For display of the configuration
of camera, the following preview function code shows it:

```cpp
void CChildView::OnPreview()
{// TODO: Add your command handler code here
  if (m_image.GetNumCapDev()<=0)
    return;
  if (m_image.IsGraphRunning())
    {m_image.StopGraph();}
  m_image.OpenCamera(0);
  m_image.SetRenderWindow(this->GetSafeHwnd());
  m_image.SetVideoSize(50,50,600,450);//video size of display
  m_image.SetAddToRot(true);
  m_image.SetCameraRate(15);//camera rate is 15
  m_image.Preview();
  /*m_image.CreateFilterGraph();
   m_image.StartGraph();*/
} 
```
Camera method

The original camera method of DLL library is trying to do too many things and is not broken down into a number of other methods so that user can use these methods directly. This provides two ways to use the method. An example is preview, the first way is open camera first, then configure camera, the configuration includes frame rate of camera, size and position of video dialog box, etc. The last step is using preview method to provide preview function. Another way is also open camera, configure camera, but then not using preview method and use some different methods, for instance CreateFilterGraph and StartGraph these two methods, they are the same as Preview method because Preview method uses these two methods, but can give the user more accurate control. Capture method is the same.

Create image in memory

In the revised program, it provides two ways to get frame data. One is to store it to disk; another way is to create an image in memory. CImage::GetCurrentFream (long BufferSize, char *pBuffer, int *iWidth, int *iHeight) is used to store frame data to disk, to pBuffer. Another method char* CImage::GetCurrentFream(int* width, int* height) is used to store frame data in memory and only return the address pointer of where internal memory starts. The following bytes are all data of this frame and the number of these bytes can be provided by another method named long CImage::GetCurrentFreamSize(). So now, it can create an image in memory and the user can display it or do applications to it.

Seek frame

The details of seeking have been discussed in section 4.9.7. In the code, it has three methods to provide this function, they are CImage::SeekToStart(), CImage::SeekToEnd(), LONGLONG CImage::Seek(long offset, bool start). The first method is used to seek to the start, second one is to seek to the end and the last one is used to seek to specific frame, either from start position or from current position. With these three methods, this system can seek any frame in the video file.

To grab the first 20 frames, firstly open a video file, when user wants to seek the first 20 frames, it will start SeekToStart method, build a loop for 20 times. In each loop, it will seek to specific, the first loop seeks to 0 frame, second loop seeks to 1st frame, third loop seeks to 2nd frame, etc. The last loop seeks to the 19th frame. Then it will get the frame data and store them to the disk or in memory. The detail of this process is shown in method CChildView::OnCapture20Frames().

To retrieve frame 456 from an MPEG, the process is nearly similar to grab the first 20 frames. Seek to the 456th frame and then get the data of this frame, store it to disk or in memory.

All of these are the interfaces that the DLL supply to the users and a list of all the methods used in the DLL is shown as follows:
- CImage::CImage(), enumerate the video capture devices, get the number and name of them.
- CImage::~CImage(), destroy the CImage() class.
- CImage::GetDevName(), get the name of devices
- CImage::GetNumCapDev(), get the number of devices.
- CImage::PreviewLive(), support function of preview.
- CImage::CaptureToFile(), support function of capture
- CImage::CameraVideo(), support function of open camera, set the parameters of camera, the way to get data of frame and the way of displaying video image.
- CImage::OpenFile(), support function of open file, set the size of window, create play filter graph, set format of video, connect filter graph, etc.
- CImage::GetCurrentFreamSize(), apply for the buffer space.
- CImage::GetCurrentFream(), get the current frame data, the width and height of image, etc.
- CImage::SeekToStart(), seek media from the start position.
- CImage::GetFileTuration(), get the duration (total frames of this video file) of the video file.
- CImage::Seek(), this is also for seeking media that provides more powerful than SeekToStart and SeekToEnd methods.
- CImage::SeekToEnd() all these 4 methods are relative to OpenFile().
- CImage::ReleaseFilterGraph(), release filter graph.
- CImage::errorHandler(), if there is any errors of filter graph, this window will show the error.
- CImage::GetPin(), get the pin of the filters in this Filter Graph.
- CImage::ConnectFilters(), connect two filters in the Filter Graph.
- CImage::AddGraphToRot, register filter graph and add graph to running object table.
- CImage::RemoveGraphFromRot(), delete the registration of filter graph.

The following functions are requirements of DLL and it provides the requirement that this video capture system can support.

- Open file: in the code, OpenFile() support this requirement.
- Open camera: user can use preview or capture to open camera, CameraVideo() support this requirement.
- Grab n frames: user can use GetCurrentFream() to get one frame. And user can use it for n times, or user decides when he will get the current frame. But here, get n frames once doesn’t make any sense, because first, user doesn’t know the size of each frame; second, user doesn’t know how big is this ‘n’, if it is too big, for example, n=2000000, there isn’t not enough internal buffer space; third, when user previews or captures, because of real time, he can’t get the frame which is not been previewed. The frames before the current position can be captured definitely. In the process
of play video file that is stored in the disk, user can get any frame very easily. Just open the video file, seek to that frame you want, and get the data of this frame.

- Control camera: user can use preview or capture that has set-up parameters. Set the parameters of camera before start preview or capture, include number of frame per second, format of colour, such as I420, RGB24, etc, and size of window. But the system has a shortcoming with this code, which is it can’t set up the parameters after start preview or capture. If user wants to change them, the only way is to stop the preview or capture. Figure Appendix 8 shows the setting of the Maxell Maxpen Camera of this project.

![Figure Appendix 8 Setting of Maxell Maxpen Camera](image)

- Get next frame: this function is only for play video file. User cannot do this when he is previewing or capturing. Because when he is previewing or capturing, he cannot get the frame that is later than current time. The same reason for Grab n frames. But for disk video file, system can support this function.
- Skip n frames: Seek () method can support it.
- Save frames: after user gets the data of frame, he can save it as .bmp file.
- Go back: this is just for play video file. OpenFile () and Play () support it.

The test project is total for presentation of using the interfaces of DLL, also an example for users. First, initialise the message handlers, user gets the number and name of video capture devices and do some preparations before he starts running this project. For example, if there is no device in this system, the button can’t be pressed. When it is confirmed that this system has some video capture devices, user can press buttons for preview or capture. The window for display has been set automatically as 640*480 size and users can set by themselves.