第6章 VFW软件开发包

6.1 VFW简介/6.2 AVI文件流操作(1)

VFW（Video for Windows）是Microsoft推出的关于数字视频的一个软件开发包，VFM的件标准。AVI（Audio Video Interleave）文件中的视频数据帧交错存放。围绕AVI文件，VFW完整的视频采集、压缩、解压、回放和编辑的应用程序编程接口（API）。由于AVI文件格式在数字视频技术中有广泛的应用，所以VFM仍有很大的实用价值，且有进一步发展之势。

在VFM的基本结构中，本章重点介绍以下模块的调用：

（1）AFIFILE.dll：支持标准多媒体I/O函数及宏调用。

（2）Avfcap.dll：它提供视频采集功能与外接硬件（如摄像头）打交道。

（3）VCM视频压缩管理器：它负责视频压缩/解码，是为应用程序调用底层压缩驱动程序

（4）MSVIDE.dll：在视图中画出视频数据图像。

在VC++开发环境中调用VFW与使用其他开发包没有什么不同，只是需要将VFW32.lib文中，但需要注意的是在开发视频捕捉与压缩管理程序时需要其他软硬件设置，具体情况将在述。另外，在VC++安装目录下提供了一些小的工具程序，可以用于测试硬件设备性能，如果入的研究，可以参阅开发文档。

6.2 AVI文件流操作

VFW为AVI文件提供了丰富的处理函数和宏定义，AVI文件的特点在于它是典型的数据流视频、音频流和文本流组成。所以对AVI文件的处理主要是处理文件流。

本节我们将依次结合具体实例讲述一下常用操作：

（1）打开/关闭文件流；

（2）读/写数据操作；

（3）在文件中定位；

（4）用剪贴板编辑数据流。

有关AVI文件处理的详细信息可在MSDN中的SDK/GRAPHICS AND MULTIMEDIA SERVICES/VIDEO FOR WINDOWS中找到。

6.2.1 打开/关闭文件流

首先要用AVIFILEINIT函数初始化AVIFILE库。下面的例程演示了打开和关闭一个AVI文件的过程。

```
BOOL LoadAVIFileThenClose(LPCSTR strFileName, HWND hwnd)
// LoadAVIFile - loads AVIFile and opens an AVI file.
//
// szfile - filename
// hwnd - window handle
//
{
   LONG hr;

   PAVIFILE pfile;

   BOOL success;

AVIFileInit();         // opens AVIFile library

success = AVIFileOpen(&pfile, strFileName, OF_SHARE_DENY_NONE , 0L);

//Returns zero if successful or an error otherwise
 if (success != 0)
 {
 ErrMsg("Unable to open %s", szFile);
  return success;
    }

    // Place functions here that interact with the open file.
   AVIFileRelease(pfile);  // closes the file

   AVIFileExit();         // releases AVIFile library
   return success;
}
```

注意：AVIFILEPOPEN中的第三个参数为打开方式，表示程序可与其他线程读写该文件。

6.2.2 读写文件数据

下面我们演示如何打开文件数据流并写数据。首先用OpenAllAVIFileStream例程打开文件的数据流，打开数据流的句柄放入gapavi[i]中。gapavi[i]设为全局变量，其类型为Pavistream类型。为了操作数据流，首先得到数据流的信息。

下面的GetStreamTypes例程从数据流中得到数据类型。

第6章 VFW软件开发包

6.2 AVI文件流操作(2)

在以下读写数据流的操作中，我们从一个数据流中拷贝数据，并压缩，然后把压缩数据写建的文件中。

```
void SaveSmall(PAVISTREAM ps, LPSTR lpFilename)
// SaveSmall - copies a stream of data from one file, compresses
// the stream, and writes the compressed stream to a new file. //
// ps stream interface pointer
// lpFilename - new AVI file to build
//
 {
   PAVIFILE           pf;
   PAVISTREAM         psSmall;
   HRESULT            hr;
   AVISTREAMINFO      strhdr;
   BITMAPINFOHEADER  bi;
   BITMAPINFOHEADER  biNew;
   LONG               lStreamSize;
   LPVOID             lpOld;
   LPVOID             lpNew;
  // Determine the size of the format data using
  // AVIStreamFormatSize.
  AVIStreamFormatSize(ps, 0, &lStreamSize);

   if (lStreamSize > sizeof(bi)) // Format too large?

   return;

   lStreamSize = sizeof(bi);

   hr = AVIStreamReadFormat(ps, 0, &bi, &lStreamSize); // Read format

   if (bi.biCompression != BI_RGB) // Wrong compression format?

   return;

   hr = AVIStreamInfo(ps, &strhdr, sizeof(strhdr));
   // Create new AVI file using AVIFileOpen.

   hr = AVIFileOpen(&pf, lpFilename, OF_WRITE | OF_CREATE, NULL);

  if (hr != 0)       return;
```

```
// Set parameters for the new stream.
biNew = bi;
biNew.biWidth /= 2;
biNew.biHeight /= 2;
biNew.biSizeImage =
((((UINT)biNew.biBitCount * biNew.biWidt + 31)&~31) / 8) * biNew.biHeight;

SetRect(&strhdr.rcFrame, 0, 0, (int) biNew.biWidth,
      (int) biNew.biHeight);

// Create a stream using AVIFileCreateStream.
hr = AVIFileCreateStream(pf, &psSmall, &strhdr);

if (hr != 0)
  {          //Stream created OK? If not, close file.
   AVIFileRelease(pf);
   return;
  }

// Set format of new stream using AVIStreamSetFormat.
hr = AVIStreamSetFormat(psSmall, 0, &biNew, sizeof(biNew));
if (hr != 0)
  {
    AVIStreamRelease(psSmall);
    AVIFileRelease(pf);
    return;
  }

// Allocate memory for the bitmaps.
lpOld  = GlobalAllocPtr(GMEM_MOVEABLE, bi.biSizeImage);
lpNew = GlobalAllocPtr(GMEM_MOVEABLE, biNew.biSizeImage);

// Read the stream data using AVIStreamRead.
for (lStreamSize = AVIStreamStart(ps); lStreamSize <
   AVIStreamEnd(ps); lStreamSize++)
{
   hr = AVIStreamRead(ps, lStreamSize, 1, lpOld, bi.biSizeImage,
      NULL, NULL);
  // Place error check here.


  // Compress the data.
   CompressDIB(&bi, lpOld, &biNew, lpNew);

// Save the compressed data using AVIStreamWrite.
```

```
        hr = AVIStreamWrite(psSmall, lStreamSize, 1, lpNew,
        biNew.biSizeImage, AVIIF_KEYFRAME, NULL, NULL);
      }
       // Close the stream and file.
      AVIStreamRelease(psSmall);
      AVIFileRelease(pf);

     //free the memory that have been malloced
      GlobalFreePtr   (lpOld);
      GlobalFreePtr   (lpNew);
    }
```

注意：在操作完成后，要释放已分配的内存，另外要检验每个函数调用后的返回结果。需据不同结果设置处理代码。

第6章 VFW软件开发包

6.2 AVI文件流操作(3)

6.2.3 利用剪贴板编辑文件

利用剪贴板编辑文件流是VFW的一大特色。接下来的例子中,我们从一组数据流中剪切、据。把剪切和拷贝下来的数据合并到一个新的文件中，最后把它放到剪贴板上。VFW提供了EditStreamClose和EditStreamCut来支持这些功能。（剪贴板是Windows操作系统内定义的一块Windows API有专用于针对剪贴板的操作例程，利用它还可以实现不同线程的数据通信，具体MSDN中的Visual C++ Documention。）

```
// Global variables
// gcpavi - count of streams in an AVI file
// gapavi[] - array of stream-interface pointers, used as data source
// gapaviSel[] - stream-interface pointers of edited streams
// galSelStart[] - edit starting point in each stream
// galSelLen[] - length of edit to make in each stream
// gapaviTemp[] - array of stream-interface pointers put on clipboard //
// Comment:
//    The editable streams in gapaviSel have been
//    initialized with CreateEditableStream.

case MENU_CUT:

case MENU_COPY:

case MENU_DELETE:
 {
   PAVIFILE pf;
   int     i;
   // Walk list of selections and make streams out of each section.
   gcpaviSel = 0;  // index counter for destination streams
   for (i = 0; i < gcpavi; i++)
      {
        if (galSelStart[i] != -1)
        {
        if (wParam == MENU_COPY)
           EditStreamCopy(gapavi[i], &galSelStart[i],
           &galSelLen[i], &gapaviSel[gcpaviSel++]);
        else
        {
           EditStreamCut(gapavi[i], &galSelStart[i],
           &galSelLen[i], &gapaviSel[gcpaviSel++]);
        }
```

```
    }
  }
. . . . . .
  // Put on the clipboard if segment is not deleted.

  if (gcpaviSel && wParam != MENU_DELETE)
    {
     PAVISTREAM   gapaviTemp[MAXNUMSTREAMS];
     int i;
     // Clone the edited streams, so that if the user does
     // more editing, the clipboard won't change.

     for (i = 0; i < gcpaviSel; i++)
      {
        gapaviTemp[i] = NULL;
        EditStreamClone(gapaviSel[i], &gapaviTemp[i]);
        // Place error check here.
      }
     // Create a file from the streams and put on clipboard.
     AVIMakeFileFromStreams(&pf, gcpaviSel, gapaviTemp);
     AVIPutFileOnClipboard(pf);
      // Release clone streams.
     for (i = 0; i < gcpaviSel; i++)
     {
        AVIStreamRelease(gapaviTemp[i]);
     }
     // Release file put on clipboard.
     AVIFileRelease(pf);
    }
   // Release streams created.
   for (i = 0; i < gcpaviSel; i++)
     AVIStreamRelease(gapaviSel[i]);
}
```

6.2.4 应用实例

下面将介绍的代码被封装到一个叫做CWriteAvi的类中，主要提供两个对象方法：

（1）CAVIFile(LPCTSTR lpszFileName,int xdim=-1,int ydim=-1)：初始化成员以便打开AVI文件库。

第6章 VFW软件开发包

6.2 AVI文件流操作(4)

（2）Bool AddFrame(CBitmap& bmp)：构建一帧位图数据并将其按一定格式压缩后，写
或新建的AVI数据流中。这个例子有助于大家深入了解AVIFile函数操作。读者可以在自己的
用这个类，并进一步修改完善。

```
//************************************************************
//  WriteAvi.h
//************************************************************
#include <vfw.h>

#define TEXT_HEIGHT      20
#define AVIIF_KEYFRAME 0x00000010L // this frame is a key frame.
#define BUFSIZE 260

class CAVIFile : public CObject
{
public:
    CAVIFile(LPCTSTR lpszFileName, int xdim =-1, int ydim =-1);
    virtual ~CAVIFile();

    virtual bool AddFrame(CBitmap& bmp);
    CString GetFName() const            {return FName;};
    virtual bool IsOK() const           {return bOK;};

private:
    CString FName;
    int xDim;
    int yDim;

    AVISTREAMINFO strhdr;
    PAVIFILE pfile;
    PAVISTREAM ps;
    PAVISTREAM psCompressed;
    PAVISTREAM psText;
    AVICOMPRESSOPTIONS opts;
    AVICOMPRESSOPTIONS FAR * aopts[1];
    DWORD dwTextFormat;
    char szText[BUFSIZE];
    int nFrames;
      bool bOK;};

//************************************************
```

```
//writevi.cpp
//*****************************************
#include “stdafx.h”
#include<windowsx.h>
#include<memory.h>

#include<mmsystem.h>
#include <vfw.h>#include
"writeavi.h"
static HANDLE  MakeDib( HBITMAP hbitmap, UINT bits )
{       HANDLE                  hdib ;
    HDC                     hdc ;
    BITMAP                  bitmap ;
    UINT                    wLineLen ;
    DWORD                   dwSize ;
    DWORD                   wColSize ;
    LPBITMAPINFOHEADER  lpbi ;
    LPBYTE                  lpBits ;
    //
    //The GetObject function obtains information about a specified
    //graphics object. Depending on the graphics object,the function
    //places a filled-in BITMAP structure into a specified buffer.
      //

    GetObject(hbitmap,sizeof(BITMAP),&bitmap) ;

   //
    // DWORD align the width of the DIB
    // Figure out the size of the colour table
    // Calculate the size of the DIB
    //
    wLineLen = (bitmap.bmWidth*bits+31)/32 * 4;
    wColSize = sizeof(RGBQUAD)*((bits <= 8) ? 1<<bits : 0);
    dwSize   = sizeof(BITMAPINFOHEADER) + wColSize +
           (DWORD)(UINT)wLineLen*(DWORD)(UINT)bitmap.bmHeight;

    //
    // Allocate room for a DIB and set the LPBI fields
    //
    hdib = GlobalAlloc(GHND,dwSize);
    if (!hdib)   return hdib ;

    lpbi = (LPBITMAPINFOHEADER)GlobalLock(hdib) ;

    lpbi->biSize = sizeof(BITMAPINFOHEADER) ;
    lpbi->biWidth = bitmap.bmWidth ;
```

```
    lpbi->biHeight = bitmap.bmHeight ;
    lpbi->biPlanes = 1 ;
    lpbi->biBitCount = (WORD) bits ;
    lpbi->biCompression = BI_RGB ;
    lpbi->biSizeImage = dwSize - sizeof(BITMAPINFOHEADER) - wColSize ;
    lpbi->biXPelsPerMeter = 0 ;
    lpbi->biYPelsPerMeter = 0 ;
    lpbi->biClrUsed = (bits <= 8) ? 1<<bits : 0;
    lpbi->biClrImportant = 0 ;

    //
    // Get the bits from the bitmap and stuff them after the LPBI
    //
    lpBits = (LPBYTE)(lpbi+1)+wColSize ;

    hdc = CreateCompatibleDC(NULL) ;

    GetDIBits(hdc,hbitmap,0,bitmap.bmHeight,lpBits,(LPBITMAPINFO)lpbi,
DIB_RGB_COLORS);

    // Fix this if GetDIBits messed it up....
    lpbi->biClrUsed = (bits <= 8) ? 1<<bits : 0;

    DeleteDC(hdc) ;
    GlobalUnlock(hdib);

    return hdib ;
}

CAVIFile::CAVIFile(LPCTSTR lpszFileName, int xdim, int ydim)
:   FName(lpszFileName),
    xDim(xdim), yDim(ydim), bOK(true), nFrames(0)
    //
    //Initial class members and
    //open the AVI library
    //
{
    pfile = NULL;
    ps = NULL;
    psCompressed = NULL;
    psText = NULL;
    aopts[0] = &opts;
    WORD wVer = HIWORD(VideoForWindowsVersion());
    if (wVer < 0x010A)
    {
        // oops, we are too old, blow out of here
```

```
            AfxMessageBox("the Version is too old");
            bOK = false;
        }
        else
        {
            AVIFileInit();
        }
    }

    CAVIFile::~CAVIFile()
    //
    //Disorgnazie the Object ,close the streams opened,
    //close the AVIFile library
    //
    {
        if (ps)
            AVIStreamClose(ps);

        if (psCompressed)
            AVIStreamClose(psCompressed);

        if (psText)
            AVIStreamClose(psText);

        if (pfile)
            AVIFileClose(pfile);

        WORD wVer = HIWORD(VideoForWindowsVersion());
        if (wVer >= 0x010A)
        {
            AVIFileExit();
        }
    }

    bool CAVIFile::AddFrame(CBitmap& bmp)
    {
        HRESULT hr;
        char szMessage[BUFSIZE];

        if (!bOK)
            return false;
        LPBITMAPINFOHEADER alpbi = (LPBITMAPINFOHEADER)GlobalLock
(MakeDib(bmp, 8));

        if (alpbi == NULL)   return false;
```

```
        if (xDim>=0 && xDim != alpbi->biWidth)
        {
                GlobalFreePtr(alpbi);
                return false;
        }
        if (yDim>=0 && yDim != alpbi->biHeight)
        {
                GlobalFreePtr(alpbi);
                return false;
        }

        xDim = alpbi->biWidth;
        yDim = alpbi->biHeight;

        if (nFrames == 0)
        {
                hr = AVIFileOpen(&pfile,              // returned file pointer
                 FName,             // file name
                                OF_WRITE | OF_CREATE  // mode to open file with
                                |OF_READ,
                                NULL);             // use handler  determined
                                                   // from file extension....
        `            if (hr != AVIERR_OK)
                {
                  GlobalFreePtr(alpbi);
                  bOK = false;
                  return false;
                }

                //Fill the strhdr with zero
                _fmemset(&strhdr, 0, sizeof(strhdr));
                //
  //Four-character code of the compressor handler that will
 //compresses this video stream when it is saved.
    //
    strhdr.fccType   =   streamtypeVIDEO;// stream type
    strhdr.fccHandler            = mmioFOURCC('M','S','V','C'));
                //
                //Dividing dwRate by dwScale gives the playback rate
                //in number of samples per second.
        //
        strhdr.dwScale             = 1;
                strhdr.dwRate              = 15;                    // 15 fps
                strhdr.dwSuggestedBufferSize  = alpbi->biSizeImage;

                //SetRect assigns the left, top, right,
```

```
                //and bottom arguments to the appropriate
                //members of the RECT structure &strhdr.rcFrame
                //with (0,0,alpbi->biWidth,alpbi->biHeight.
                //
        // rectangle for  stream
         SetRect(&strhdr.rcFrame, 0, 0,        (int)alpbi-
>biWidth,
                (int) alpbi->biHeight);

                // And create the stream;
                hr = AVIFileCreateStream(pfile,                 // file pointer
                                         &ps,               // returned stream pointer
                                        &strhdr);          // stream header
                if (hr != AVIERR_OK)
                {
                      GlobalFreePtr(alpbi);
                      bOK = false;
                      return false;
                }
               //Fill the opts with zero
                _fmemset(&opts, 0, sizeof(opts));
                if (!AVISaveOptions(NULL, 0, 1, &ps,
                          (LPAVICOMPRESSOPTIONS FAR *)&aopts))
                {
                      GlobalFreePtr(alpbi);
                      bOK = false;
                      return false;
                }

                hr = AVIMakeCompressedStream(&psCompressed, ps, &opts, NULL);
                if (hr != AVIERR_OK)
                {
                      GlobalFreePtr(alpbi);
                      bOK = false;
                      return false;
                }

                hr = AVIStreamSetFormat(psCompressed, 0,
                                        alpbi,        // stream format
                                  alpbi->biSize +   // format size
                                  alpbi->biClrUsed * sizeof(RGBQUAD));
                if (hr != AVIERR_OK)
                {
                      GlobalFreePtr(alpbi);
                      bOK = false;
                      return false;
```

```
        }

        // Fill in the stream header for the text stream……
        // The text stream is in 60ths of a second……

        _fmemset(&strhdr, 0, sizeof(strhdr));
        strhdr.fccType              = streamtypeTEXT;
        strhdr.fccHandler           = mmioFOURCC('D', 'R', 'A', 'W');
        strhdr.dwScale              = 1;
        strhdr.dwRate               = 60;
        strhdr.dwSuggestedBufferSize  = sizeof(szText);

        SetRect(&strhdr.rcFrame, 0, (int) alpbi->biHeight,
                (int) alpbi->biWidth, (int) alpbi->biHeight + TEXT_HEIGHT);

        //……and create the stream.
        hr = AVIFileCreateStream(pfile, &psText, &strhdr);
        if (hr != AVIERR_OK)
        {
                GlobalFreePtr(alpbi);
                bOK = false;
                return false;
        }

        dwTextFormat = sizeof(dwTextFormat);
        hr = AVIStreamSetFormat(psText, 0, &dwTextFormat, sizeof
          (dwTextFormat));
        if (hr != AVIERR_OK)
        {
                GlobalFreePtr(alpbi);
                bOK = false;
                return false;
        }

}

hr = AVIStreamWrite(psCompressed,       // stream pointer
        nFrames * 10,                   // time of this frame
        1,                              // number to write
        (LPBYTE) alpbi +                // pointer to data
                alpbi->biSize +
                alpbi->biClrUsed * sizeof(RGBQUAD),
                alpbi->biSizeImage,     // size of this frame
        AVIIF_KEYFRAME,                 // flags....
        NULL,
        NULL);
```

```
    if (hr != AVIERR_OK)
    {
        GlobalFreePtr(alpbi);
        bOK = false;
        return false;
    }

    // Make some text to put in the file ...
    //LoadString(hInstance, IDS_TEXTFORMAT, szMessage, BUFSIZE );

    strcpy(szMessage, "This is frame #%d");

    int iLen = wsprintf(szText, szMessage, (int)(nFrames + 1));

    // ... and write it as well.
    hr = AVIStreamWrite(psText,
            nFrames * 40,
            1,
            szText,
            iLen + 1,
            AVIIF_KEYFRAME,
            NULL,
            NULL);
    if (hr != AVIERR_OK)
    {
        GlobalFreePtr(alpbi);
        bOK = false;
        return false;
    }

    GlobalFreePtr(alpbi);
    nFrames++;
    return true;
}
```

第6章 VFW软件开发包

6.3 用DrawDib绘制图像(1)

在整个VFW体系中，DrawDib模块用于在视频终端（如显示器）上回放视频数据，这里的视频数据可以是从摄像头直接采集的,也可以是位图文件。DrawDib的函数DIBs（Device_Independent Bitmaps）格式的图像显示，其色彩深度包括8位、16位、24位、32位。另外DrawDib直接写入视频缓冲区，不依赖于GDI函数。

6.3.1 DrawDib的性能特点

总的来说，DrawDib函数提供图像显示、图像缩放以及色彩变换等功能，另外还提供图像解压缩和视频流功能。在使用过程中要注意，DrawDib只能显示使用DIB_RGB_COLORS格式的图像，并只能以拉伸方式显示位图，对于其他情况要考虑用StretchDIBits函数。但另一方面，DrawDib支持16色VGA显卡、256色SVGA显卡，并有高质量的色彩适配功能（如能在16色显卡上显示256色图像）。同样值得注意的是，DrawDib可以显示任何Windows压缩标准（如RELE、411YVV、INDEO等）的图像。

6.3.2 DrawDib的函数

应用程序可调用DrawDib函数来建立并管理一个DrawDib Dc（Device Contxt），显示或更新图像，调色板，以及关闭DrawDib DC。另外，DrawDib也支持计时功能,这一功能只用于调试版的应用程序，可以测定显示特性。

表6.1列出了几个函数及其主要功能。

表6.1 DrawDib函数

函 数	解 释
库操作:	
DrawDibOpen	打开DrawDib库, 创建DrawDib DC
DrawDibClose	关闭DrawDib DC
DrawDibProfileDisplay	获取当前系统处理DrawDib类函数的模式
图像显示:	
DrawDibDraw	绘制 DIB 图像
DrawDibGetBuffer	获得解压图像的缓冲区
DrawDibUpdate	显示缓冲区的最后一幅图像
图像序列:	
DrawDibBegin	改变一个DrawDib DC 或重新初始化一个DrawDib DC

续表

函 数	解 释
DrawDibEnd	清除对DrawDib DC所作的修改
DrawDibStart	为视频流回放准备DrawDib DC
DrawDibStop	示范用于视频流回放的资源
调色板:	
DrawDibRealize	根据DC 设置调色板
DrawDibSetPalette	设置显示DIB 的调色板
DrawDibGetPalette	获得调色板
DrawDibChangPalette	改变调色板
DrawDib的时间调试:	
DrawDibTime	获得显示操作的时间信息

6.3.3 DrawDib的运行方式

首先在做任何操作之前，我们要调用DrawDibOpen来装载动态链接库（dll），分配内存资DrawDib DC。它返回一个HDRAWDIB类型的对象指针，以供后面的函数使用，该指针指向对象。当一切操作完成后，相应地用DrawDibClose函数释放。为应用程序，可以同时创建多特别是当你要同时绘制多个位图时，但最好给不同的DC赋予不同的特征值，以便用来识别或能。我们也可以在同一个视图中建立2个DC：一个用来显示位图全貌，另一个放大显示其中

为了方便运行，函数需要了解显卡适配器及其驱动程序的有关信息。可使用函数DrawDibProfileDisplay，根据它的返回值来判断系统对位图格式的支持情况。

6.3.4 位图文件格式

要装载一张位图，首先要了解它的格式。图6.1描述了一个位图文件的结构（不管它是在磁盘上）。

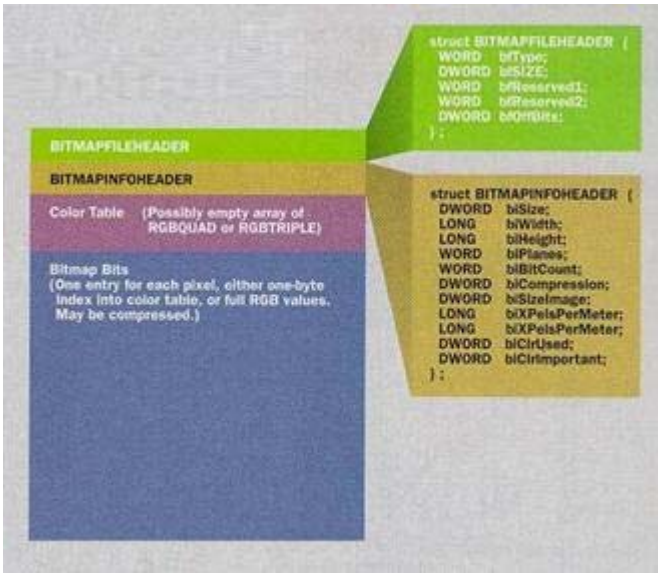

图6.1 位图文件的结构

一般来讲，一个位图文件有三部分组成：一个BITMAPINFOHEADER结构、一个颜色表主体。描述位图文件性质以及文件格式的信息写在文件头结构中。如果一个位图拥有颜色表由首字节编入颜色表索引中，但如果是256色图像，则没有颜色表，它们用RGB（RED，GR BLUE）三色值表示色彩，这样的文件体积大，但效果好。

6.3.5 使用Palette增强图像

DrawDib中调用调色板的函数已列在表6.1中。可以把调色板想像成Windows管理色彩显示一个调色板实体的数据结构定义如下：

```
typedef struct tagPALETTEENTRY
{
BYTE peRed;
BYTE peGreen;
BYTE peBlue;
```

```
BYTE peFlags;
} PALETTEENTRY;
```

最后一位PeFlags定义了调色板实体的使用方式，其余三个元素定义了各种基本颜色的显
称以上调色板为系统调色板。另外，还有一种调色板叫做逻辑调色板LOGPALETTE，其结构

```
typedef struct tagLOGPALETTE
{
  WORD        palVersion;
  WORD        palNumEntries;
  PALETTEENTRY palPalEntry[1];
} LOGPALETTE;
```

但一般程序并不需要设定逻辑调色板。

在有关调色板编程中，应用程序提供两个消息调用接口：

（1）WM_QVERYNEWPALETTE（无参数）

它用来在一个视图窗口获得“焦点”（处于激活状态）时，给窗口一个实现逻辑调色板的
可以暂时使窗口无效，好让DrawDibDraw重绘视图。

（2）WM_PALETTECHANGED：参数hwndpalchg

它用来在激活窗口实现了逻辑调色板（从而改变了系统调色板）之后，通知所有打开的窗口，更新他们的
意，如果这个窗口在此时实现它自己的调色板，那么这个窗口线程就会陷入死循环。可以用消息参数hwn
表引起系统调色板变化的那个窗口的句柄）判断是否可以调用DrawDibRealize。

第6章 VFW软件开发包

6.3 用DrawDib绘制图像(2)

下面一段程序简单描述了消息的处理方法：

```
case WM_PALETTECHANGED:

  if ((HWND)wParam == hwnd)   //if it is me that cause the
                              //system palette changed,
        break;                //doing nothing;otherwise
                              //execute the programme    below;

case WM_QUERYNEWPALETTE:
{
   hdc = GetDC(hwnd);

   f = DrawDibRealize(hdd, hdc, FALSE) ;

   ReleaseDC(hwnd, hdc);

   if (f)
     InvalidateRect(hwnd, NULL, TRUE);
      break;
     hdc = GetDC(hwnd);
 }
```

激活调色板的过程很简单，以下程序用DrawDibRealize、DrawDibChangPalette和DrawDib现了这一过程。

```
//allow palette changes, specify the
//DDF_ANIMATE flag in the call to DrawDibBegin

DrawDibBegin(hdd, ……, DDF_ANIMATE);

DrawDibRealize(hdd, hdc, fBackground);

DrawDibDraw(hdd, hdc, ……, DDF_SAME_DRAW|DDF_SAME_HDC);

//set the color table values from the palette entries
//by using DrawDibChangePalette.
// lppe is an address of thePALETTEENTRY array containing the new colors,
DrawDibChangePalette(hDD, iStart, iLen, lppe);

//do you own job. . .
ReleaseDC(hwnd, hdc);
```

总之，用调色板可以方便对我们的位图资源实施颜色变幻，可以追求自己喜欢的色调。

第6章 VFW软件开发包

6.4 视频捕捉(1)

熟悉视频电话和NetMeeting的人对电脑的视频捕捉能力都有印象。实际使用的视频捕捉
摄像头或数码照相机，它们都可以把外部的光信号变成数字信号并通过接口传给计算机，进
放。视频捕捉的应用很广泛，可用于日常生活，如制作家庭录相；也可用于远程监控（被采
过网络传送）；甚至可在装有摄像头的电脑自控系统上实现视觉功能。

VFW为开发人员提供了较丰富的接口函数调用，它适用于开发USB（Universal Serial Bu
像设备。本节主要介绍视频捕捉AVICap的功能及调用方法并编制一个简单的程序来使用小型
头。

6.4.1 开发AVICap应用程序的软硬件配置

首先对于主机来说，要有与采集设备相连接的接口或插槽。有些设备可直接接到机箱上
USB摄像头），而有专用设备可能要占据一个PCI插槽。

在软件方面，操作系统必须安装该设备的驱动程序。设备厂家通常都会提供自己设备程
安装驱动程序会占用系统资源（如内存、I／O中断等）。

在安装完驱动程序后，要确认设备是否可正常工作。如果设备供应商没有提供一个测试
有），在Windows 98或NT DDK中src\videocap\tools里有个vidcap32程序，它可用来检验用
是否正常。

6.4.2 AVICap主要功能

AVICap的主要功能是通过一个采集窗口（Capture Windows）来提供的。在概念上讲，采
于一些标准控件，例如按钮、列表框等。各功能函数的调用都需要采集窗口的句柄，AVICap
递给采集窗口的。通过系统窗口可使用以下基本功能：

（1）设置系统参数，如采集速率（frame/s）。

（2）实时地获取视频数据，可用Preview模式对输入视频进行实时显示，可用Capture功
数据写入文件中。

（3）动态地控制数据流功能：可以动态地连接或断开设备；可以在采集过程中切换采集
过视／音频格式对话框与压缩驱动程序对话框设置各项参数。

（4）暴露了采集到的数据体，为用户对数据进行更深处理（如使用调色板）提供了机会

6.4.3 Captune Window的操作方法

通过上一节，我们已对Captune Windows有了初步认识。下面介绍关于Captune Windows
这里主要讲三个方面：

（1）建立采集窗口；

（2）连接一个采集设备；

（3）获取窗口状态。

首先，建立采集窗口是进行一切工作的前提。使用CapCreateCaptuneWindows函数创建一

口。采集窗口使用WS_CHILD| WS_VISIBLE风格，下面一个例子演示了采集窗口的参数配

```
    hWndC = capCreateCaptureWindow
(
    (LPSTR) "My Capture Window", // window name if pop-up
    WS_CHILD | WS_VISIBLE,       // window style
    0, 0, 160, 120,              // window position and dimensions
    (HWND) hwndParent,
   (int) nID /* child ID */)
```

这个函数通常在程序初始化时调用，并且同一个程序可创建多个采集窗口，以用于连接

备。

接下来要做的是与一个采集设备取得联系。CapDriuerConnect和WM_CAP_DRIVER_CONNEC

现这一功能：

```
fOK = SendMessage (hWndC, WM_CAP_DRIVER_CONNECT, 0, 0L);
//
// Or, use the macro to connect to the MSVIDEO driver:
// fOK = capDriverConnect(hWndC, 0);
//
// Place code to set up and capture video here.
// capDriverDisconnect (hWndC);
```

使用CapDriverDisConnect可断开连接。

如果想得到有关设备的详细信息，可以使用CapGetDriverDescription得到设备描述符，

CapDriverGetCaps宏得到设备性能：

```
char szDeviceName[80];
char szDeviceVersion[80];
CAPDRIVERCAPS CapDrvCaps;

for (wIndex = 0; wIndex < 10; wIndex++)
{
    if (capGetDriverDescription (wIndex, szDeviceName,
        sizeof (szDeviceName), szDeviceVersion,
        sizeof (szDeviceVersion))
    {
        // Append name to list of installed capture drivers
        // and then let the user select a driver to use.
    }
}
  SendMessage (hWndC, WM_CAP_DRIVER_GET_CAPS,
  sizeof (CAPDRIVERCAPS), (LONG) (LPVOID) &CapDrvCaps);
// Or, use the macro to retrieve the driver capabilities.
// capDriverGetCaps(hWndC, &CapDrvCaps, sizeof (CAPDRIVERCAPS));
```

窗口状态被定义在一个叫做CAPSTATUS的数据结构中，它包含了大量的有用信息，必须动

以满足应用程序的各种操作要求，可以通过CapGetStatus或发WM_CAP_GET_STATUS消息得到C

针：

```
CAPSTATUS CapStatus;

capGetStatus(hWndC, &CapStatus, sizeof (CAPSTATUS));

SendMessage (hWndC, WM_CAP_GET_STATUS ,
&CapStatus, sizeof (CAPSTATUS));
//
//read or write information by the handle
//of CapStatus.............
```

当一个采集窗口建成后，下面要做的是设置回调函数（CallBackFunctions）。

6.4.4 回调函数简介

回调函数应在程序初始化时制定。当应用程序向设备，更确切地说，向设备驱动程序发出
后，回调函数被定时或不定时地用来完成相应的通信功能，如视频流回调函数定时地从USB接
帧数据以便回放。应用程序可以在采集窗口中注册以下四种回调函数，它们的名字可由应用
定。

第6章 VFW软件开发包

6.4 视频捕捉(2)

（1）状态回调函数：取回采集状态信息，包含在文本字符串里。

```
statusCallbackProc: status callback function
// hWnd:                capture window handle
// nID:                 status code for the current status
// lpStatusText:        status text string for the current status //

LRESULT PASCAL StatusCallbackProc(HWND hWnd, int nID, LPSTR lpStatusText)
 {
    if (!ghWndMain)         return FALSE;
    if (nID == 0)
     {
         // Clear old status messages.
        SetWindowText(ghWndMain, (LPSTR) gachAppName);
        return (LRESULT) TRUE;
    }
}
```

（2）错误回调函数：可以把错误信息包含在文本字符串里。

```
// ErrorCallbackProc:    error callback function
// hWnd:               capture window handle
// nErrID:              error code for the encountered error
// lpErrorText:           error text string for the encountered error

LRESULT PASCAL ErrorCallbackProc(HWND hWnd, int nErrID,     LPSTR lpErrorText)
{
    if (!ghWndMain)         return FALSE;
    if (nErrID == 0)             // Starting a new major function.
        return TRUE;           // Clear out old errors.

    // Show the error identifier and text.
    wsprintf(gachBuffer, "Error# %d", nErrID);
    MessageBox(hWnd, lpErrorText, gachBuffer,
        MB_OK | MB_ICONEXCLAMATION);

    return (LRESULT) TRUE;
}
```

（3）单帧采集回调函数，参数lpVHdr指向视频数据块的头信息。

```
// FrameCallbackProc: frame callback function
// hWnd:               capture window handle
// lpVHdr:             pointer to struct containing captured
```

```
//                    frame information //
LRESULT PASCAL FrameCallbackProc(HWND hWnd, LPVIDEOHDR lpVHdr)
 {
     if (!ghWndMain)         return FALSE;

    wsprintf(gachBuffer, "Preview frame# %ld ", gdwFrameNum++);
     SetWindowText(ghWndMain, (LPSTR)gachBuffer);

     return (LRESULT) TRUE ;

 }
```

VIDEOHDR结构体的定义如下：

```
typedef struct videohdr_tag {
    LPBYTE      lpData;                 /* pointer to locked data buffer */
    DWORD       dwBufferLength;         /* Length of data buffer */
    DWORD       dwBytesUsed;            /* Bytes actually used */
    DWORD       dwTimeCaptured;         /* Milliseconds from start of stream */
    DWORD       dwUser;                 /* for client's use */
    DWORD       dwFlags;                /* assorted flags (see defines) */
    DWORD       dwReserved[4];          /* reserved for driver */
} VIDEOHDR, NEAR *PVIDEOHDR, FAR * LPVIDEOHDR;
```

（4）视频流回调函数：它的参数与单帧采集回调函数的相同，但它用于连续读取视频数 PreView模式下，此函数以设定的采集速率返回视频数据，形成视频流，它的原型为：

```
LRESULT CALLBACK capVideoStreamCallback(  HWND hWnd, LPVIDEOHDR lpVHdr  );
```

回调函数可由应用程序根据需要动态地设定或删除，方法是使用宏：

```
BOOL CapSetCallbackOnXXX (hwnd, tpProc)
```

其中参数hwnd表示采集窗口的句柄，fpProc表示回调函数的指针，如果fpProc为NULL，则 个设定的回调函数。

下面一段代码演示了在程序初始时设定回调函数的过程：

```
case WM_CREATE: {
    char    achDeviceName[80] ;
     char    achDeviceVersion[100] ;
     char    achBuffer[100] ;
     WORD    wDriverCount = 0 ;
     WORD    wIndex ;
     WORD    wError ;
     HMENU   hMenu ;

    // Create a capture window using the capCreateCaptureWindow macro.
    ghWndCap = capCreateCaptureWindow((LPSTR)"Capture Window",
    WS_CHILD | WS_VISIBLE, 0, 0, 160, 120, (HWND) hWnd, (int) 0);
```

```
    // Register the error callback function using the
    // capSetCallbackOnError macro.
     capSetCallbackOnError(ghWndCap, fpErrorCallback);
// Register the status callback function using the
    // capSetCallbackOnStatus macro.
     capSetCallbackOnStatus(ghWndCap, fpStatusCallback);

    // Register the video-stream callback function using the
    // capSetCallbackOnVideoStream macro.
     capSetCallbackOnVideoStream(ghWndCap, fpVideoCallback);
  // Register the frame callback function using the
    // capSetCallbackOnFrame macro.
     capSetCallbackOnFrame(ghWndCap, fpFrameCallback);

     // Connect to a capture driver
     break;
      }
case WM_CLOSE:
{
// Use the capSetCallbackOnFrame macro to
// disable the frame callback. Similar calls exist for the other
// callback functions.
  capSetCallbackOnFrame(hWndC, NULL);

break;
}
```

第6章 VFW软件开发包

6.4 视频捕捉(3)

6.4.5 视频预览与采集

在前面工作的基础上，可以让采集系统真正地工作起来，让采到的视频数据显示出来或
中。

所谓预览状态，就是在显示器上显示连续或单帧图像。在预览状态时，可以调整显示模
色，或调试外设（如可调节摄像头焦距）。在满意后，就可以进入采集状态，设定保存文件
在文件中。

预览操作比较简单，下面一段代码首先设定预览速率，单位为帧 / 毫秒：

```
capPreviewRate(hWndC, 66);       //  possible 15 frames per second
capPreview(hWndC, TRUE);       //  starts preview
capPreview(hWnd, FALSE);        //  disables preview
```

其实CapPreview只是初始化了预览状态，然后它开始调用VidoCallbuekProc回调函数。
后，应用程序可用DrawDib把图像画到视图框内。如果不做处理，则下一帧数据将覆盖前一
Capture Data操作要麻烦些。首先要设置采集参数。AVICap用CAPTUREPARMS结构记录采
不愿意设置，则可以接受缺省值。

```
CAPTUREPARMS capParam;
char szNewName[] = "NEWFILE.AVI";

if( capCaptureGetSetup(  hwnd,    sizof(CAPTUREPARMS), &capParam  ))
{
//
// chang the setting of CAPTUREPARMS……
//
    if(capCaptureSetSetup(  hwnd,     &CapParm,    sizeof(CAPTUREPARMS) ))
    {
    capFileSetCaptureFile(  hwnd,    szNewName );
    capCaptureSequence(  hwnd  );
    }
}
```

6.4.6 采集设置对话框

AVICap在采集过程中可随时对视频数据源、视频格式和压缩格式进行设置。AVICap为用
的接口。可以直接显示设置对话框，操作与编程都很方便。下面介绍几个常用的函数。

```
BOOL capDlgVideoSource(hwnd);
```

使用该函数可弹出视频源对话框，如图6.2和6.3所示。

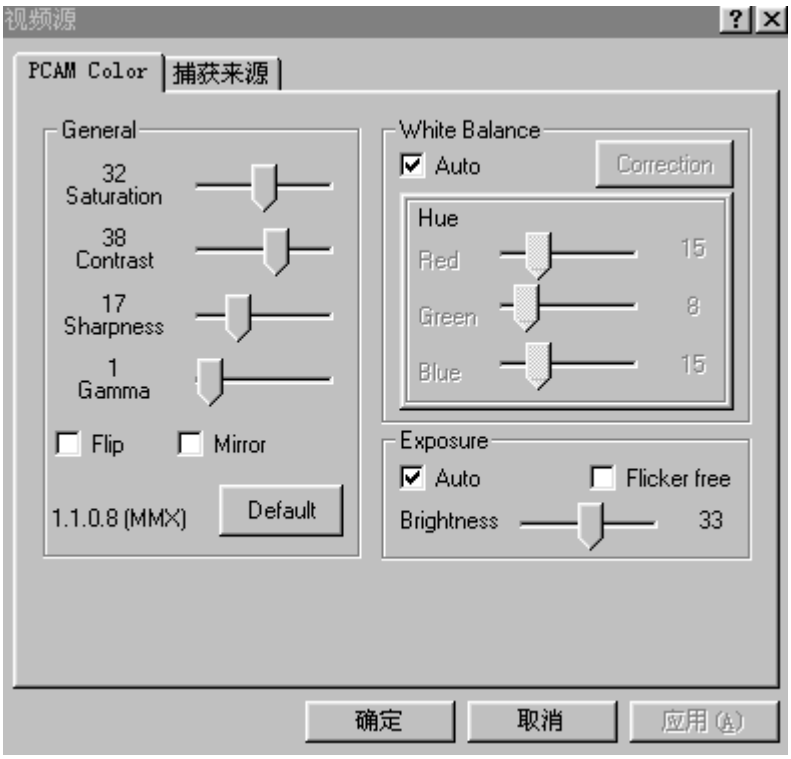

图6.2 视频源对话框－PCAM Color

图6.3 视频源对话框－捕获来源

```
BOOL capDlgVideoFormat(hwnd);
```

该函数可显示视频格式对话框，它可设置分辨率和像素深度，如图6.4所示。

图6.4　视频格式对话框

```
BOOL capDlgVideoCompression(hwnd);
```

该函数用于显示压缩驱动程序对话框。可以选择一个已安装的压缩驱动程序，如图6.5所

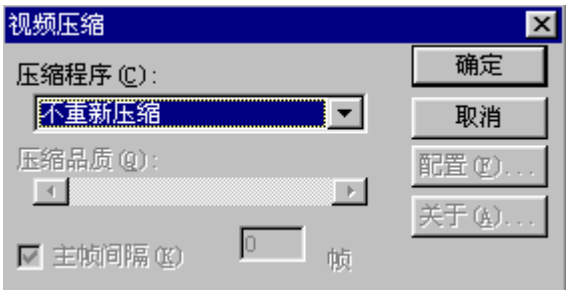

图6.5　压缩驱动程序对话框

第6章 VFW软件开发包

6.5 视频压缩管理器/6.6 视频捕捉程序示例(1)

通过AVICap所采集的原始数据容量很大，如果直接保存或实时地从网上传输，需要占用
间和网络带宽。数据压缩在很多时候是必不可少的工作，许多软件商为用户提供了Windows
驱动程序。VCM视频压缩管理器的设计目的是为用户提供调用压缩驱动程序的统一接口，以
程。本节将简要地介绍VCM的功能，以便为实际编程打下基础。

6.5.1 VCM的功能

如上所述，VCM处于应用程序与驱动程序之间，它本身具有独立性，给用户提供编程接口
的代理服务器在概念上相似。图6.6表达了VCM在系统中的关系。

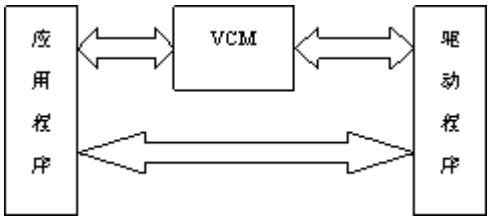

图6.6 VCM在系统中的关系

通过VCM，应用程序的编程要简单，而且可以适用于各种驱动程序，从而增强了可移植性
序VCM时，VCM将其转换成对应的消息，然后通过ICSemMessage将消息发到合适的压缩或解压
时VCM收到从驱动程序返回的信息，将其转交给应用程序处理。

通过调用VCM，可以实现如下功能：

（1）定位、打开或安装压缩／解压驱动程序。

（2）配置压缩／解压驱动程序或获得压缩／解压驱动程序的配置信息。

（3）压缩、解压或显示视频数据。

使用VCM的函数和宏可以实现以上功能，但读者可能已注意到，在关于DrawDib的例程中，我们已使用了压缩功能。可以说，通过DrawDib实现数据压缩要更简单些。

6.5.2 VCM驱动注册

Windows利用注册表保存和检索VCM驱动程序信息。每种驱动程序都用两个四字符的代码
XXXX.YYYY，其中前四个字符由系统定义，用于表示驱动程序的类别，见表6.2。

表6.2 4字符的驱动程序类别标识

4 字 符 串	描　　述
“VIDC”	压缩／解压驱动程序
“VIDS”	视频流显示驱动程序
“TXTS”	文本流显示驱动程序
“AUDS”	音频流处理程序

第二个四字符串由每个驱动程序自己制定。一般地说，第二个四字符串应与驱动程序能
型相符。

当用户打开一个VCM驱动程序时，用户应先确定要处理的数据类型。VCM将试图打开相应

如果失败，则在系统注册表中搜索，寻找与前4个字符相匹配的驱动程序，即找到一个能据的驱动程序。

6.6 视频捕捉程序示例

程序示例VideoCapture用AVICap在视频窗口中捕捉视频数据流，同时利用DrawDib在另一放所采集的视频数据，该示例程序能够适合一般摄像头应用的需求。下面介绍该Video程序的骤。

1. 利用AppWirzard生成单文档程序框架VideoCapture，此时工程文件中包含MainFrm.cp Stadafx.cpp，VideoCapture.cpp，VideoCaptureDoc.cpp，VideoCaptureView.cpp，对应的源文件VideoCapture.rc。我们将修改资源文件和MainFrm.cpp、MainFrm.h，VideoCaptureV Video ViewCapture.h文件。

2. 修改界面菜单，修改后的显示如图6.7所示。

其中：

- 设置格式：用于显示和设置视频的格式，其ID等于ID_FORMAT；
- 图像预览：用于开始或停止预览视频，其ID等于ID_PREVIEW；
- 采集窗口：用于显示或隐藏视频采集源窗口，其ID等于ID_SOURCE；
- 视频捕捉：用于开始或停止视频采集，其ID等于ID_CAPTURE；
- 退出程序：用于关闭AVICap，其ID等于ID_APP_EXIT。

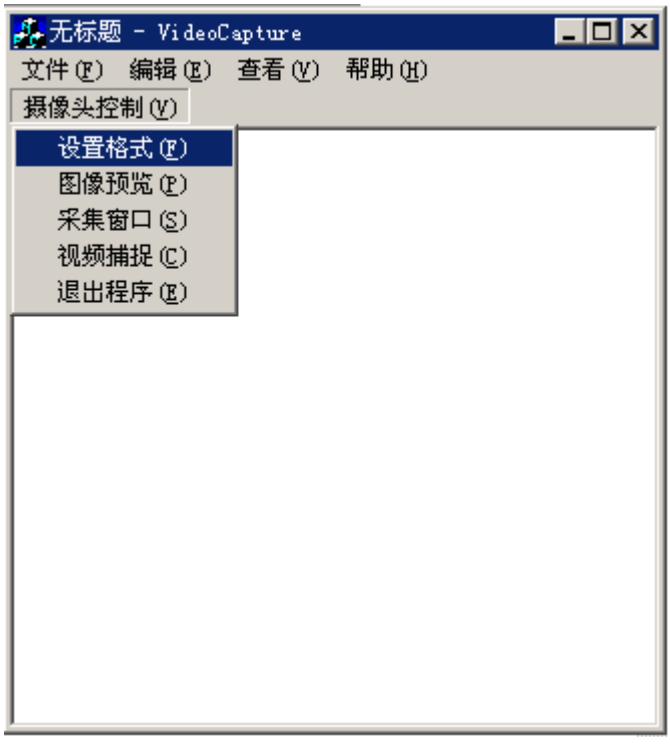

图6.7 修改界面菜单

3. 修改MainFrm.h文件，在CMainFrame类定义前包含Video For Windows头文件（vfw.h 据结构DIBINFO：

```
#include<vfw.h>
```

```
typedef struct {
int headsize; //bitmap headsize
char buffer[32000]; // bitmap head and data
} DIBINFO,* PDIBINFO;
```

DIBINFO用于存放一帧图像的信息。在此数据结构中，buffer用于存放采集到的一帧图像为简单起见，使用静态数组。

在CMainFrame的//Attributes部分增加如下变量和函数说明：

```
//Attributes
public:
DIBINFO          m_dibinfo;
CFrameWnd        m_wndSource;
HWND            m_h WndCap;
CAPDRIVERCAPS  m_caps;
Void ChangeSize ();
```

第6章 VFW软件开发包

6.6 视频捕捉程序示例(2)

4. 修改MainFrm.cpp文件。

（1）编写视频采集所需要的3个回调函数。

```
LRESULT CALLBACK EXPORT ErrorCallbackProc(HWND hWnd, int nErrID,
       LPSTR lpErrorText)
// hWnd:     Application main window handle
// nErrID:    Error code for the encountered error
// lpErrorText: Error text string for the encountered error
{
   if (nErrID==0)
       return TRUE;        //Clear out old errors …
   AfxMessageBox (lpErrorText, MB_OK,NULL);

   return TRUE;
}

LRESULT FAR PASCAL StatusCallbackProc(HWND hWnd,int nID,
   LPSTR lpStatusText)
// hWnd:    Application main window handle
// Nid:      Status code for the current status
// lpStatusText: Status text string for the crurrent status
{
   static int CurrentID;

   // the CAP_END message sometimes overwrites a useful
   // statistics message.
    if (nID==IDS_CAP_END)
{
    if ((CurrentID==IDS_CAP_STAT_VIDEOAUDIO )||
       (CurrentID==IDS_CAP_STAT_VIDEOONLY))
          return(TRUE);
}
     CurrentID=nID;
     return(LRESULT)TRUE;
}

LRESULT FAR PASCAL VideoCallbackProc(HWND hWnd,LPVIDEOHDR lpVHdr)
{

((CMainFrame *)AfxGetMainWnd())->m_dibinfo.
bitmapheader.biSizeImage = lpVHdr->dwBytesUsed;
```

```
memcpy(((CMainFrame *)AfxGetMainWnd( ))->m_dibinfo.buffer+
((CMainFrame *)AfxGetMainWnd( ))->m_dibinfo.VideoFormatSize,
lpVHdr->lpData,lpVHdr->dwBytesUsed);

(((CMainFrame *)AfxGetMainWnd( ))->GetActiveView( ))->
InvalidateRect(NULL,FALSE);

 return(LRESULT)TRUE;
}
```

其中VideoCallbackProc回调函数最为重要。系统采集一帧图像后调用该函数。在示例程将在主窗口中回放采集到的图像数据。

（2）实现ChangeSize( )函数，该函数用于调整视频源窗口的大小，以适应实际视频格

```
void CMainFrame::ChangeSize()
{

  static UINT uiVideoX=0;
  static UINT uiVideoY=0;
  CAPSTATUS cs;
  RECT rc;

  m_wndSource.GetWindowRect(&rc);
  capGetStatus(m_hWndCap,&cs, sizeof(cs));

  if(uiVideoX!=cs.uiImageWidth||uiVideoY!=cs.uiImageHeight)
{
  m_wndSource.MoveWindow(rc.left, rc.top,
  cs.uiImageWidth+4,cs.uiImageHeight+28);

  uiVideoX=cs.uiImageWidth;
  uiVideoY=cs.uiImageWidth;
}

  //Get video format
  m_dibinfo.VideoFormatSize=capGetVideoFormatSize(m_hWndCap);
  capGetVideoFormat(m_hWndCap,&m_dibinfo.bitmapheader,m_dibinfo.VideoFormatSize);
}
```

（3）修改CMainFrame的OnCreate函数，增加以下代码，用于初始化视频。

```
if(!m_wndSource.CreateEx(WS_EX_TOPMOST, NULL, "Source",
WS_CAPTION,CRect(100,100,150,180),NULL,0))
return -1;
m_wndSource.ShowWindow(SW_HIDE);
m_hWndCap=capCreateCaptureWindow((LPSTR)"Video Capture Window",
WS_CHILD|WS_VISIBLE,0,0,300,240,m_wndSource.m_hWnd,0);
```

```
capSetCallbackOnError(m_hWndCap,(FARPROC)ErrorCallbackProc);
capSetCallbackOnStatus(m_hWndCap,(FARPROC)StatusCallbackProc);
capSetCallbackOnVideoStream(m_hWndCap,(FARPROC)VideoCallbackProc);
capDriverConnect(m_hWndCap,0);//connect to the first VIDEOdriver
capDriverGetCaps(m_hWndCap,&m_caps,sizeof(CAPDRIVERCAPS));

ChangeSize();
```

第6章 VFW软件开发包

6.6 视频捕捉程序示例(3)

（4）利用ClassWizard处理WM_CLOSE消息，结束相应的视频处理。

```
void CmainFrame::OnClose()
{
//TODO:Add your message handler code here and/or call default
void CMainFrame::OnAppExit()
{
          // TODO: Add your command handler code here
         capCaptureAbort(m_hWndCap);//stop capture
    capSetCallbackOnError(m_hWndCap,NULL);
    capSetCallbackOnStatus(m_hWndCap,NULL);
    capSetCallbackOnVideoStream(m_hWndCap,NULL);
    capCaptureAbort(m_hWndCap);//stop to capture
    capDriverDisconnect(m_hWndCap);

    CFrameWnd::OnClose();
}
```

（5）利用ClassWizard增加对菜单项Format的处理，显示视频格式对话框。

```
void CmainFrame::OnFormat()
{
       // TODO: Add your command handler code here
       if(m_caps.fHasDlgVideoFormat)
       {
     capDlgVideoFormat(m_hWndCap);
     ChangeSize();
       }
}
```

（6）利用ClassWizard增加对菜单项Preview的处理，预览视频。

```
void CMainFrame::OnPreview()
{
       // TODO: Add your command handler code here
       static BOOL bPreview=FALSE;
   bPreview=!bPreview;
   capPreviewRate(m_hWndCap,1000/10);//10 frames/second
   capPreview(m_hWndCap,bPreview);

}
```

（7）利用ClassWizard增加对菜单项SourceWindow的处理，显示或隐藏视频源窗口。

```
void CMainFrame::OnSource()
{
         // TODO: Add your command handler code here
        static BOOL bShow=FALSE;
    bShow=!bShow;
    m_wndSource.ShowWindow(bShow?SW_SHOW:SW_HIDE);

}
```

（8）利用ClassWizard增加对菜单项Capture的处理，开始或停止视频采集。

```
void CmainFrame::OnCapture()
{
        //TODO:Add your command handler code here
    static BOOL  bCapture=FALSE;
    CAPTUREPARMS CapParms;
    bCapture=!bCapture;
   if(bCapture)
   {
   //Get the defuault setup for video capture from the AVICap window
   capCaptureGetSetup(m_hWndCap, &CapParms,sizeof(CAPTUREPARMS));
   //set the defaults we won't bother the user with
   CapParms.dwIndexSize=324000;//3(hour)*30(fps)*60*60
   CapParms.fMakeUserHitOKToCapture=!CapParms.fMCIControl;
   CapParms.wPercentDropForError=100;
   CapParms.wNumVideoRequested=5;
   CapParms.wChunkGranularity=0;
   CapParms.fYield=TRUE;
   CapParms.fCaptureAudio=FALSE;//don't capture audio
   CapParms.vKeyAbort=0;

   CapParms.fAbortLeftMouse=CapParms.fAbortRightMouse=FALSE;
   CapParms.dwRequestMicroSecPerFrame=1000000/10;
   capSetCallbackOnYield(m_hWndCap,NULL);

   //set the new setup info
   capCaptureSetSetup(m_hWndCap, &CapParms, sizeof(CAPTUREPARMS));
   capCaptureSequenceNoFile(m_hWndCap);
   }
   else capCaptureAbort(m_hWndCap);
}
```

5. 修改VideoCaptureView.h文件，在文件头增加：

```
#include<vfw.h>
```

在CVideoCaptureView类中增加变量说明：

```
HDRAWDIB m_hdd;
```

6．修改VideoCaptureView.cpp文件，使其包含头文件mainfrm.h：

```
#include “mainfrm.h”
```

在CVideoCaptureView的构造函数中初始化m_hdd；在CVideoCaptureView:: CVideoCaptu
中增加一行：

```
m_hdd = DrawDibOpen();
```

在CVideoCaptureView 的析构函数中关闭m_hdd；在CVideoCaptureView:: ~CVideoCaptu
中增加一行：

```
DrawDibClose(m_hdd);
```

7．修改CVideoCaptureView的OnDraw函数，具体实现如下：

```
void CVideoCaptureView::OnDraw(CDC* pDC)
{
        CVideoCaptureDoc* pDoc = GetDocument();
        ASSERT_VALID(pDoc);

         // TODO: add draw code for native data here
    RECT rect;
    GetClientRect(&rect);
    DrawDibDraw(m_hdd,pDC->GetSafeHdc(),
         rect.left, rect.top,rect.right, rect.bottom,
    &(((CMainFrame*)AfxGetMainWnd())->m_dibinfo.bitmapheader),
        NULL,0,0,-1,-1,0);
}
```

8．运行程序，在保证摄像头正确安装后，单击摄像头控制/采集窗口菜单会弹出视频源，
示。

图6.8 运行结果