A Simple C# Wrapper for the AviFile Library

Corinna John

12 Nov 2012 CPOL

Edit AVI files in .NET.

Download source files - 18.5 Kb

Download source and demo project files - 45.7 Kb

Download source and demo project files for .NET 1.1 (SharpDevelop project) - 32.1 Kb

Introduction

This is an article for everyone who does not want to spend hours messing around with the AVIFile functions, if he only wants to read or change a simple AVI video. I have wrapped the most important AVIFile functions into three easy to use C# classes that can handle the following tasks:

- Read images from the video stream.
- Decompress a compressed video stream.
- Compress an uncompressed video stream.
- Change the compression of a video stream.
- Export the video stream into a separate *.avi* file.
- Export the audio stream into a *.wav* file.
- Copy a couple of seconds from audio and video stream into a new *.avi* file.
- Add sound from a *.wav* file to the video.
- Create a new video stream from a list of bitmaps.
- Add frames to an existing video stream, compressed or not.
- Insert frames into a stream.
- Copy or delete frames from a stream.

These features cover the common use cases like creating a video from a couple of images and a wave sound, extracting the sound track from a video, cutting out short clips, or grabbing a single picture from a movie.

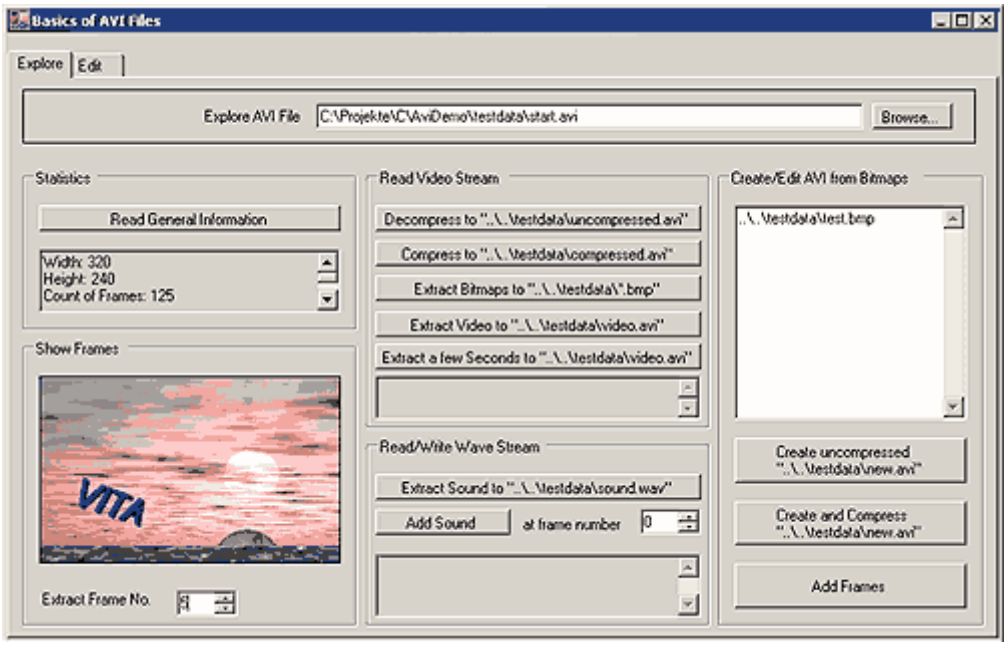
This article has got two sections:

- First, I'll explain the demo application so that you can use it to explore the library on your own.
- Then, I'll explain how the library works step by step.

How to use the library - A walk through the demo application

The Explore tab

The *Explore* tab lets you explore an AVI file, and offers simple tasks that don't need editable streams:



At the top of the form, you can choose the AVI file you want to explore. *text.avi* from the test data folder is pre-selected. On the left side, you can display header information about the video and the wave sound stream (if available). Also, you can see image frames from the video in a `PictureBox`:

The images are read by a `VideoStream` object. `GetFrameOpen` prepares the stream for decompressing frames, `GetFrameClose` releases the resources used to decompress the frame, and `GetBitmap` decompresses a frame and converts it to a `System.Drawing.Bitmap`.

C# Copy Code

```
AviManager aviManager = new AviManager(txtAviFileName.Text, true);
VideoStream aviStream = aviManager.GetVideoStream();
aviStream.GetFrameOpen();
picFrame.Image =
  aviStream.GetBitmap(Convert.ToInt32(numPosition.Value));
aviStream.GetFrameClose();
aviManager.Close();
```

In the middle of the demo form, you can work with whole streams:

Decompress removes the compression from a video stream. It creates a new file and video stream, decompresses each frame from the old stream, and writes it into the new stream. The result is a large new *.avi* file with the same video but no compression.

Compress changes the compression of the video, or applies compression to an uncompressed video. It does the same as *Uncompress*, but compresses the new stream. These two functions use the same method `CopyFile`:

C# Copy Code

```
private void CopyFile(String newName, bool compress){
    //open compressed file
    AviManager aviManager =
        new AviManager(txtAviFileName.Text, true);
    VideoStream aviStream = aviManager.GetVideoStream();
    //create un-/re-compressed file
    AviManager newManager =
        aviStream.DecompressToNewFile(newName, compress);

    //close compressed file
    aviManager.Close();
    //save and close un-/re-compressed file
    newManager.Close();
}
```

Whenever an instance of `VideoStream` creates a compressed stream, it asks you for a codec and settings:

Extract Bitmaps splits the video into many separate bitmap files:

C# Copy Code

```
VideoStream stream = aviManager.GetVideoStream();
stream.GetFrameOpen();

String path = @"..\..\testdata\";
for(int n=0; n<stream.CountFrames; n++){
    stream.ExportBitmap(n, path+n.ToString()+".bmp");
}

stream.GetFrameClose();
```

Of course, you can save the images in any format. `ExportBitmap` is just a shortcut for these three lines:

C# Copy Code

```
Bitmap bmp = stream.GetBitmap(position);
bmp.Save(FileName);
bmp.Dispose();
```

Extract Video copies the whole video stream into a new AVI file. You can use it to get rid of all the other streams like MIDI, text and Wave sound.

The lower box deals with Wave sound. *Extract Sound* copies the whole sound stream into a Wave file. This is not a big task, it requires only four lines of code:

C# Copy Code

```
AviManager aviManager =
    new AviManager(txtAviFileName.Text, true);
AudioStream audioStream = aviManager.GetWaveStream();
audioStream.ExportStream( @"..\..\testdata\sound.wav" );
aviManager.Close();
```

Extract a few Seconds lets you copy video and sound between second X and second Y. First, a `CopyForm` dialog lets you enter X and Y, then these parts are cut out of the video and sound streams:

C# Copy Code

```
AviManager aviManager = new AviManager(txtAviFileName.Text, true);
VideoStream aviStream = aviManager.GetVideoStream();
CopyForm dialog = new CopyForm(0,
  aviStream.CountFrames / aviStream.FrameRate);
if (dialog.ShowDialog() == DialogResult.OK) {
  int startSecond = dialog.Start;
  int stopSecond = dialog.Stop;

  AviManager newFile = aviManager.CopyTo(
    "..\\..\\testdata\\video.avi", startSecond, stopSecond);
  newFile.Close();
}
aviManager.Close();
```

Add Sound lets you choose a *.wav* file, and adds it to the video. You can use this feature to add a sound track to a silent video, for example, re-add the sound to a video extracted with *Extract Video*. Adding sound is a simple task of three lines:

C# Copy Code

```
String fileName = GetFileName("Sounds (*.wav)|*.wav");
if(fileName != null){
    AviManager aviManager =
        new AviManager(txtAviFileName.Text, true);
    aviManager.AddAudioStream(fileName);
    aviManager.Close();
}
```

The last set of functions is about creating new video streams. Enter a list of image files in the box and animate them:

Create uncompressed builds a new video from the images, and saves it without any compression. *Create and Compress* does the same, except that it displays the compression settings dialog and compresses the images. Both methods create a new file, and pass a sample bitmap to `AddVideoStream`. The sample bitmap is used to set the format of the new stream. Then, all the images from the list are added to the video.

C# Copy Code

```
//load the first image
Bitmap bitmap = (Bitmap)Image.FromFile(txtFileNames.Lines[0]);
//create a new AVI file
AviManager aviManager =
    new AviManager(@"..\..\testdata\new.avi", false);
//add a new video stream and one frame to the new file
VideoStream aviStream =
    aviManager.AddVideoStream(true, 2, bitmap);

Bitmap bitmap;
int count = 0;
for(int n=1; n<txtFileNames.Lines.Length; n++){
    if(txtFileNames.Lines[n].Trim().Length > 0){
        bitmap =
```

```
            (Bitmap)Bitmap.FromFile(txtFileNames.Lines[n]);
        aviStream.AddFrame(bitmap);
        bitmap.Dispose();
        count++;
    }
}
aviManager.Close();
```

Add Frames appends the images to the existing video stream. To an uncompressed video stream, we could append frames by simply opening the stream and adding frames as usual. But a compressed stream cannot be re-compressed. `AVIStreamWrite` - used by `AddFrame` - would not return any error; but anyway, it would add the new frames uncompressed and produce nothing but strangely colored pixel storms. To add frames to a compressed stream, the existing frames must be decompressed and added to a new compressed stream. Then the additional frames can be added to that stream:

C# Shrink ▲ Copy Code

```
//open file
Bitmap bmp = (Bitmap)Image.FromFile(txtFileNames.Lines[0]);
AviManager aviManager =
    new AviManager(txtAviFileName.Text, true);
VideoStream aviStream = aviManager.GetVideoStream();

//streams cannot be edited - copy to a new file
AviManager newManager = aviStream.DecompressToNewFile(
                        @"..\..\testdata\temp.avi", true);
aviStream = newManager.GetOpenStream(0);

//add images
Bitmap bitmap;
for(int n=0; n<txtFileNames.Lines.Length; n++){
    if(txtFileNames.Lines[n].Trim().Length > 0){
        bitmap =
            (Bitmap)Bitmap.FromFile(txtFileNames.Lines[n]);
        aviStream.AddFrame(bitmap);
        bitmap.Dispose();
    }
}

aviManager.Close(); //close old file
newManager.Close(); //save and close new file

//delete old file, replace with new file
System.IO.File.Delete(txtAviFileName.Text);
System.IO.File.Move(@"..\..\testdata\temp.avi",
                            txtAviFileName.Text);
```

Now that you know how to use the AVIFile wrapper classes, let's have a look at the background.

The Edit tab

The *Edit* tab demonstrates tasks for editable AVI streams, like pasting frames at any position in the stream, or changing the frame rate:

When you have chosen a file to edit, an editable stream is created from the video stream, and the editor buttons become enabled. A normal video stream is locked; for inserting and deleting frames, you need an editable stream:

C# Copy Code

```
AviManager file = new AviManager(fileName, true);
VideoStream stream = file.GetVideoStream();
EditableVideoStream editableStream =
           new EditableVideoStream(stream);
file.Close();
```

On the left side, you can copy or cut frame sequences, and paste them at another position in the same stream:

Copying frames from one stream, and pasting them into another or the same stream, is only two lines of code:

C# Copy Code

```
//copy frames
IntPtr copiedData = editableStream.Copy(start, length);
```

```
//insert frames
editableStream.Paste(copiedData, 0, position, length);
```

There is no other method for deleting frames than just cut and forget them:

C# Copy Code

```
//cut and paste frames
IntPtr copiedData = editableStream.Cut(start, length);
editableStream.Paste(copiedData, 0, position, length);

//delete frames == cut without paste
IntPtr deletedData = editableStream.Cut(start, length);
```

In the middle of the dialog, you can insert frames from image files anywhere in the stream, and change the frame rate to make the video play back slower or faster:

We can paste only streams, not bitmaps, so the bitmaps from the list are written into a temporary AVI file and then pasted as a stream:

C# Copy Code

```
//create temporary video file
String tempFileName = System.IO.Path.GetTempFileName() + ".avi";
AviManager tempFile = new AviManager(tempFileName, false);

//write the new frames into the temporary video stream
Bitmap bitmap =
  (Bitmap)Image.FromFile(txtNewFrameFileName.Lines[0].Trim());
tempFile.AddVideoStream(false, 1, bitmap);
VideoStream stream = tempFile.GetVideoStream();

for (int n=1; n<txtNewFrameFileName.Lines.Length; n++) {
   if (txtNewFrameFileName.Lines[n].Trim().Length > 0) {
       stream.AddFrame(
         (Bitmap)Image.FromFile(txtNewFrameFileName.Lines[n]));
   }
}

//paste the video into the editable stream
editableStream.Paste(stream, 0,
   (int)numPastePositionBitmap.Value, stream.CountFrames);
```

Do you find your video too slow, or too fast? Tell the player application to play more/less frames per second:

C# Copy Code

```
Avi.AVISTREAMINFO info = editableStream.StreamInfo;
info.dwRate = (int)(numFrameRate.Value * 10000);
```

```
info.dwScale = 10000;
editableStream.SetInfo(info);
```

The last box is not for editing, it is only a preview player. You should preview your editable stream before saving it to an AVI file.

A preview player is easy to implement, you only need a `PictureBox` and the video stream you want to play. A label displaying the current frame index can be helpful, too. A start button, a stop button, and there you are:

C# Copy Code

```
private void btnPlay_Click(object sender, EventArgs e) {
    player = new AviPlayer(editableStream,
                pictureboxPreview, labelFrameIndex);
    player.Stopped += new System.EventHandler(player_Stopped);
    player.Start();
    SetPreviewButtonsState();
}

private void player_Stopped(object sender, EventArgs e) {
        btnPlay.Invoke(
           new SimpleDelegate(SetPreviewButtonsState));
}

private void SetPreviewButtonsState() {
        btnPlay.Enabled = ! player.IsRunning;
        btnStop.Enabled = player.IsRunning;
}

private void btnStop_Click(object sender, EventArgs e) {
        player.Stop();
}
```

How it works

`AviManger` manages the streams in an AVI file. The constructor takes the name of the file and opens it. `Close` closes all opened streams and the file itself. You can add new streams with `AddVideoStream` and `AddAudioStream`. New video streams are empty, Wave streams can only be created from Wave files. After you have created an empty video stream, use the methods of `VideoStream` to fill it. But what actually happens when you add a stream?

Create a video stream

There are two methods for creating a new video stream: create from a sample bitmap, or create from explicit format information. Both methods do the same, they pass their parameter on to `VideoStream` and add the new stream to the internal list of opened streams, to close them before closing the file:

C# Shrink ▲ Copy Code

```
public VideoStream AddVideoStream(
    bool isCompressed, //display the compression dialog,
     // create a compressed stream
    int frameRate, //frames per second
    int frameSize, //size of one frame in bytes
    int width, int height, PixelFormat format //format of
                                              //the bitmaps
    ){

    VideoStream stream = new VideoStream(
        aviFile,
        isCompressed,
        frameRate,
        frameSize,
        width, height, format);

    streams.Add(stream);
    return stream;
}
```

```
public VideoStream AddVideoStream(
    bool isCompressed, //display the compression dialog,
      // create a compressed stream
    int frameRate, //frames per second
    Bitmap firstFrame //get the format from this image
      // and add it to the new stream
    ){

    VideoStream stream = new VideoStream(
        aviFile,
        isCompressed,
        frameRate,
        firstFrame);

    streams.Add(stream);
    return stream;
}
```

Then, `VideoStream` uses the format data to create a new stream. It calls `AVIFileCreateStream` and, if `writeCompressed` says so, `AVIMakeCompressedStream`:

C# Shrink ▲ Copy Code

```
public VideoStream(
    int aviFile, //pointer to the file object
    bool writeCompressed, //true: create compressed stream
    int frameRate, //frames per second
    ...
    ){

    //store format information
    //...

    //create the stream
    CreateStream();
}

private void CreateStream(){
    //fill stream information
    Avi.AVISTREAMINFO strhdr = new Avi.AVISTREAMINFO();
    strhdr.fccType = Avi.mmioStringToFOURCC("vids", 0);
    strhdr.fccHandler = Avi.mmioStringToFOURCC("CVID", 0);
    strhdr.dwScale = 1;
    strhdr.dwRate = frameRate;
    strhdr.dwSuggestedBufferSize = frameSize;
    strhdr.dwQuality = -1; //default
    strhdr.rcFrame.bottom = (uint)height;
    strhdr.rcFrame.right = (uint)width;
    strhdr.szName = new UInt16[64];

    //create the stream
    int result = Avi.AVIFileCreateStream(aviFile,
                          out aviStream, ref strhdr);

    if(writeCompressed){
        //create a compressed stream from
        //the uncompressed stream
        CreateCompressedStream();
    }
}

private void CreateCompressedStream(){
    Avi.AVICOMPRESSOPTIONS_CLASS options =
              new Avi.AVICOMPRESSOPTIONS_CLASS();
    options.fccType = (uint)Avi.streamtypeVIDEO;
    options.lpParms = IntPtr.Zero;
    options.lpFormat = IntPtr.Zero;

    //display the compression options dialog
```

```
    Avi.AVISaveOptions(
      IntPtr.Zero,
      Avi.ICMF_CHOOSE_KEYFRAME | Avi.ICMF_CHOOSE_DATARATE,
      1, ref aviStream, ref options);

    //get a compressed stream
    Avi.AVICOMPRESSOPTIONS structOptions =
                            options.ToStruct();
    int result = Avi.AVIMakeCompressedStream(
        out compressedStream,
        aviStream,
        ref structOptions, 0);

    //format the compressed stream
    SetFormat(compressedStream);
}
```

AVICOMPRESSOPTIONS_CLASS is the AVICOMPRESSOPTIONS structure as a class. Using classes instead of structures is the easiest way to deal with pointers to pointers. If you don't know what I'm talking about, you probably have never used AVISaveOptions or AVISaveV in .NET. Take a look at the original declaration:

C# Copy Code

```
BOOL AVISaveOptions(
  HWND hwnd,
  UINT uiFlags,
  int nStreams,
  PAVISTREAM * ppavi,
  LPAVICOMPRESSOPTIONS * plpOptions
);
```

LPAVICOMPRESSOPTIONS is a pointer to a pointer to an AVICOMPRESSOPTIONS structure. In C#, structures are passed by value. If you pass a structure by ref, a pointer to the structure is passed. Instances of classes are always passed to methods as pointers. So a class-parameter by ref means a pointer to a pointer to the object. The C# declarations of AVISaveOptions and AVICOMPRESSOPTIONS are:

C# Shrink ▲ Copy Code

```
[DllImport("avifil32.dll")]
public static extern bool AVISaveOptions(
    IntPtr hwnd,
    UInt32 uiFlags,
    Int32 nStreams,
    ref IntPtr ppavi,
    ref AVICOMPRESSOPTIONS_CLASS plpOptions
    );

[StructLayout(LayoutKind.Sequential, Pack=1)]
    public struct AVICOMPRESSOPTIONS {
    public UInt32   fccType;
    public UInt32   fccHandler;
    public UInt32   dwKeyFrameEvery;
    public UInt32   dwQuality;
    public UInt32   dwBytesPerSecond;
    public UInt32   dwFlags;
    public IntPtr   lpFormat;
    public UInt32   cbFormat;
    public IntPtr   lpParms;
    public UInt32   cbParms;
    public UInt32   dwInterleaveEvery;
}

[StructLayout(LayoutKind.Sequential, Pack=1)]
public class AVICOMPRESSOPTIONS_CLASS {
    public UInt32   fccType;
    public UInt32   fccHandler;
    public UInt32   dwKeyFrameEvery;
```

```
    public UInt32   dwQuality;
    public UInt32   dwBytesPerSecond;
    public UInt32   dwFlags;
    public IntPtr   lpFormat;
    public UInt32   cbFormat;
    public IntPtr   lpParms;
    public UInt32   cbParms;
    public UInt32   dwInterleaveEvery;

    public AVICOMPRESSOPTIONS ToStruct(){
        AVICOMPRESSOPTIONS returnVar = new AVICOMPRESSOPTIONS();
        returnVar.fccType = this.fccType;
        returnVar.fccHandler = this.fccHandler;
        returnVar.dwKeyFrameEvery = this.dwKeyFrameEvery;
        returnVar.dwQuality = this.dwQuality;
        returnVar.dwBytesPerSecond = this.dwBytesPerSecond;
        returnVar.dwFlags = this.dwFlags;
        returnVar.lpFormat = this.lpFormat;
        returnVar.cbFormat = this.cbFormat;
        returnVar.lpParms = this.lpParms;
        returnVar.cbParms = this.cbParms;
        returnVar.dwInterleaveEvery = this.dwInterleaveEvery;
        return returnVar;
    }
}
```

With this workaround, we are able to call `AVISaveOptions` and (later on) `AVISaveV` in C#. Now, the new stream can be filled with image frames using `AddFrame`:

C# Shrink ▲ Copy Code

```
public void AddFrame(Bitmap bmp){
    bmp.RotateFlip(RotateFlipType.RotateNoneFlipY);

    if (countFrames == 0){
       //  the format of the first frame defines the format of the stream
       CopyPalette(bmp.Palette);
       SetFormat(writeCompressed ? compressedStream : aviStream,
                 countFrames);
    }

    //lock the memory block
    BitmapData bmpDat = bmp.LockBits(
        new Rectangle(
        0,0, bmp.Width, bmp.Height),
        ImageLockMode.ReadOnly, bmp.PixelFormat);

    //add the bitmap to the (un-)compressed stream
    int result = Avi.AVIStreamWrite(
        writeCompressed ? compressedStream : aviStream,
        countFrames, 1,
        bmpDat.Scan0,
        (Int32)(bmpDat.Stride * bmpDat.Height),
        0, 0, 0);

    //unlock the memory block
    bmp.UnlockBits(bmpDat);

    //count the frames, so that we don't have to
    //call AVIStreamLength for every new frame
    countFrames++;
}
```

Now, we are able to fill an empty stream with images. But what can we do to add frames to an existing stream? Well, first, we have to open the stream with the third constructor.

Re-compress a video stream

C# Copy Code

```
public VideoStream(int aviFile, IntPtr aviStream){

    this.aviFile = aviFile;
    this.aviStream = aviStream;

    //read the stream's format
    Avi.BITMAPINFOHEADER bih = new Avi.BITMAPINFOHEADER();
    int size = Marshal.SizeOf(bih);
    Avi.AVIStreamReadFormat(aviStream, 0, ref bih, ref size);
    Avi.AVISTREAMINFO streamInfo = GetStreamInfo(aviStream);

    //store the important format values
    this.frameRate = streamInfo.dwRate / streamInfo.dwScale;
    this.width = (int)streamInfo.rcFrame.right;
    this.height = (int)streamInfo.rcFrame.bottom;
    this.frameSize = bih.biSizeImage;
    this.countBitsPerPixel = bih.biBitCount;

    //get the count of frames that are already there
    int firstFrame = Avi.AVIStreamStart(aviStream.ToInt32());
    countFrames =
       firstFrame + Avi.AVIStreamLength(aviStream.ToInt32());
}
```

If you are sure the video stream is not compressed, you can call `AddFrame` now. Otherwise, you have to decompress the existing frames, and recompress them into a new stream:

C# Copy Code

```
public AviManager DecompressToNewFile(String fileName,
                                      bool recompress){
    //create a new AVI file
    AviManager newFile = new AviManager(fileName, false);

    //create a video stream in the new file
    this.GetFrameOpen();
    Bitmap frame = GetBitmap(0);
    VideoStream newStream =
        newFile.AddVideoStream(recompress, frameRate, frame);

    //decompress each frame and add it to the new stream
    for(int n=1; n<countFrames; n++){
        frame = GetBitmap(n);
        newStream.AddFrame(frame);
    }

    this.GetFrameClose();

    return newFile;
}
```

`DecompressToNewFile` creates a writeable copy of the stream in a new file. You can add frames to this new stream, close the new `AviManager` to save it, and then add the sound stream from the old file to complete the copy. Adding frames to a video is not easy, but this way it works.

Separate a stream

Sometimes, you might have a video file with sound, but you only need the silent video, or only the sound. It is not necessary to copy each frame, you can open the stream as usual and export it with `AVISaveV`. This works with all kinds of streams, only the compression options are different:

C# Copy Code

```
public override void ExportStream(String fileName){
    Avi.AVICOMPRESSOPTIONS_CLASS opts =
                new Avi.AVICOMPRESSOPTIONS_CLASS();

    //for video streams
    opts.fccType = (UInt32)Avi.mmioStringToFOURCC("vids", 0);
    opts.fccHandler = (UInt32)Avi.mmioStringToFOURCC("CVID", 0);

    //for audio streams
    //opts.fccType = (UInt32)Avi.mmioStringToFOURCC("auds", 0);
    //opts.fccHandler = (UInt32)Avi.mmioStringToFOURCC("CAUD", 0);

    //export the stream
    Avi.AVISaveV(fileName, 0, 0, 1, ref aviStream, ref opts);
}
```

Import sound from a Wave file

Now, we are able to build a video from bitmaps, and extract sound from it. And how does the sound get into the file? We could use AVISaveV again, to combine the video and audio streams in a new file - but we don't have to. The easiest way to add a new audio stream is to open the Wave file as an AVI file with only one stream, and then copy that stream:

C# Shrink ▲ Copy Code

```
public void AddAudioStream(String waveFileName){
    //open the wave file
    AviManager audioManager =
        new AviManager(waveFileName, true);
    //get the wave sound as an audio stream...
    AudioStream newStream = audioManager.GetWaveStream();
    //...and add it to the file
    AddAudioStream(newStream);
    audioManager.Close();
}

public void AddAudioStream(AudioStream newStream){
    Avi.AVISTREAMINFO streamInfo = new Avi.AVISTREAMINFO();
    Avi.PCMWAVEFORMAT streamFormat = new Avi.PCMWAVEFORMAT();
    int streamLength = 0;

    //read header info, format and length,
    //and get a pointer to the wave data
    IntPtr waveData = newStream.GetStreamData(
        ref streamInfo,
        ref streamFormat,
        ref streamLength);

    //create new stream
    IntPtr aviStream;
    Avi.AVIFileCreateStream(aviFile, out aviStream,
                                    ref streamInfo);

    //add format new stream
    Avi.AVIStreamSetFormat(
        aviStream, 0,
        ref streamFormat,
        Marshal.SizeOf(streamFormat));

    //copy the raw wave data into the new stream
    Avi.AVIStreamWrite(
        aviStream, 0,
        streamLength,
        waveData,
        streamLength,
        Avi.AVIIF_KEYFRAME, 0, 0);
```

```
    Avi.AVIStreamRelease(aviStream);
}
```

Copy a clip from video and sound

I have added this method, because many people asked me how this could be done. To copy a part of the video stream from second X to second Y, the indices of the first and last frames have to be calculated from the frame rate and second. For the Wave stream, we must calculate the byte offsets from samples per second, bits per sample, and the requested seconds. The rest is only *copy and paste*:

C# Shrink ▲ Copy Code

```
public AviManager CopyTo(String newFileName,
    int startAtSecond, int stopAtSecond) {
    AviManager newFile = new AviManager(newFileName, false);

    try {
        //copy video stream

        VideoStream videoStream = GetVideoStream();

        int startFrameIndex =
             videoStream.FrameRate * startAtSecond;
        int stopFrameIndex =
             videoStream.FrameRate * stopAtSecond;

        videoStream.GetFrameOpen();
        Bitmap bmp = videoStream.GetBitmap(startFrameIndex);

        VideoStream newStream = newFile.AddVideoStream(
             false,
             videoStream.FrameRate,
             bmp);

        for (int n = startFrameIndex + 1;
                  n <= stopFrameIndex; n++) {
            bmp = videoStream.GetBitmap(n);
            newStream.AddFrame(bmp);
        }
        videoStream.GetFrameClose();

        //copy audio stream

        AudioStream waveStream = GetWaveStream();

        Avi.AVISTREAMINFO streamInfo =
                        new Avi.AVISTREAMINFO();
        Avi.PCMWAVEFORMAT streamFormat =
                        new Avi.PCMWAVEFORMAT();
        int streamLength = 0;
        IntPtr ptrRawData = waveStream.GetStreamData(
            ref streamInfo,
            ref streamFormat,
            ref streamLength);

        int startByteIndex = waveStream.CountSamplesPerSecond
               * startAtSecond
               * waveStream.CountBitsPerSample / 8;

        int stopByteIndex = waveStream.CountSamplesPerSecond
               * stopAtSecond
               * waveStream.CountBitsPerSample / 8;

        ptrRawData =
          new IntPtr(ptrRawData.ToInt32() + startByteIndex);

        byte[] rawData =
```

```
            new byte[stopByteIndex - startByteIndex];
        Marshal.Copy(ptrRawData, rawData, 0, rawData.Length);

        streamInfo.dwLength = rawData.Length;
        streamInfo.dwStart = 0;

        IntPtr unmanagedRawData =
              Marshal.AllocHGlobal(rawData.Length);
        Marshal.Copy(rawData, 0, unmanagedRawData,
                                    rawData.Length);

        newFile.AddAudioStream(unmanagedRawData,
              streamInfo,
              streamFormat,
              rawData.Length);

    } catch (Exception ex) {
        newFile.Close();
        throw ex;
    }

    return newFile;
}
```

If you are still interested in AVI videos, download the wrapper library and the demo application. Finally, I dare to say: have fun with AVIFile!

Known issues

Adding frames to an existing stream does not work with all video codecs and/or bitmaps. You might get a `StackOverflowException` or broken frames. If you find out why this happens, please let me know.

History

- 9th July, 2004 - updated downloads.
- 3rd October, 2004 - new method `AviManager.CopyTo`.
- 18th December, 2004 - new classes `EditableVideoStream` and `AviPlayer`, and a few memory leaks fixed.
- 26th November, 2005 - removed performance reducers from `VideoStream.GetBitmap`.
- 01st January, 2006 - Corrections for invalid colour depth and interlaced video in `VideoStream.GetFrameOpen`, new property `VideoStream.FirstFrame`. Thanks a lot to Michael Covington!

License

Share

About the Author

Corinna John

Software Developer
Germany

Watch this Member

Corinna lives in Hanover/Germany and works as a C# developer.

Comments and Discussions

Add a Comment or Question

Email Alerts | Search Comments

First Prev Next

My vote of 5
abdellamk 7-Aug-21 12:01

can't open the video
Member 14273910 10-May-19 10:25

Exception in AVIFileOpen: -2147205009
Member 13187869 26-Jul-17 4:40

Compressing uncompressed video with multiple images is throwing an error
filtration 19-May-17 1:20

Error While AddVideoStream for Images to Video
Alican Akyol 25-Apr-17 17:05

Great work. Unfortunately a little outdated
Member @napoli 15-Nov-16 20:53

Exception : result is -2147205019 in method AddAudioStream
ChikaDen 11-Jul-16 16:25

Re: Exception : result is -2147205019 in method AddAudioStream
Unity Paradox 8-Jan-17 2:07

Error while trying stream.getframeopen()
Member 12569506 7-Jun-16 12:17

AVIStreamWrite File Size Limit
GSRId 12-Apr-16 15:09

Images to video
Member 11163981 12-Mar-16 8:02

Re: Images to video
Alican Akyol 25-Apr-17 16:56

Gut genug!
Christian Merritt 15-Feb-16 22:00

Crash on GetVideoStream()
Dmitri Nesteruk 16-Aug-15 19:50

Error While adding frame to video
Aqua Aqua 2-Apr-15 15:42

A generic error occurred in GDI+ EXCEPTION
Ladislav Kožej 23-Nov-14 21:11

A first chance exception of type 'System.AccessViolationException' occurred in AviFile.dll
Freddie Richard 12-Nov-14 20:54

Re: A first chance exception of type 'System.AccessViolationException' occurred in AviFile.dll
Denis Špelič 28-Nov-14 13:36

Re: A first chance exception of type 'System.AccessViolationException' occurred in AviFile.dll
Andriy Vasylyk 3-Jan-15 17:00

Re: A first chance exception of type 'System.AccessViolationException' occurred in AviFile.dll
putuyuwono 21-Sep-16 8:42

Re: A first chance exception of type 'System.AccessViolationException' occurred in AviFile.dll
Member @napoli 15-Nov-16 20:51

Re: A first chance exception of type 'System.AccessViolationException' occurred in AviFile.dll
MisterWayne10571973179 3-May-17 12:48

Re: A first chance exception of type 'System.AccessViolationException' occurred in AviFile.dll
Member 11211791 27-Aug-18 13:02

last 3 frames run fast
Member 11211791 6-Nov-14 13:32

Re: last 3 frames run fast
Member 11211791 27-Aug-18 13:34

Refresh 1 2 3 4 5 6 7 8 9 10 11 Next ▷

General News Suggestion Question Bug Answer Joke Praise Rant Admin

Use Ctrl+Left/Right to switch messages, Ctrl+Up/Down to switch threads, Ctrl+Shift+Left/Right to switch pages.