TEnhAviCap component

Properties Methods Events

Unit
EnhAviCap

Description
The TEnhAviCap component handles video display and capture with video for windows devices.

Shareware
Please, note that this component is shareware. You can try it to see if it fits your needs but you can't sell nor use it in commercial programs. To make a long story short: you can't gain money through this component.
The component will work correctly while Delphi 2 is running.

The registration fee is, at present, 29 US$, but, for updated infos and special offers look on the web. Over the web, you can find how to register this component at **http://ascu.unian.it/~milani/delphi**
If the above link should ever be moved, try looking at the DSP (Delphi Super Page) **http://sunsite.icm.edu.pl/delphi**.

My e-mail address is **mc3078@mclink.it** or **milani@ascu.unian.it**.

Properties

CapDrivers	Fps	PreviewScaled
CaptureAudio	IsCapturing	VFWVersion
CapWnd	OkToCapture	
ConnectAs	Overlay	
Depth	Preview	

Events

OnFrame

Methods

CanOverlay	DlgCompression	OpenVideo
CaptureStart	DlgDisplay	ToClipboard
CaptureStop	DlgFormat	
CloseVideo	DlgSource	
ConnectDriver	IsOpen	

CapDrivers

Declaration

```
property CapDrivers: TStringList;
```

Description

Run-time and read-only. The CapDrivers property contains the names of the installed video for windows devices.

CaptureAudio

Declaration
```
property CaptureAudio: boolean;
```

Description
Selects if audio capture should be enabled when capturing to an AVI file.

CapWnd

Declaration
```
property CapWnd: HWnd;
```

Description
Run-time and read-only. The CapWnd property contains the handle to the opened AVIWnd.
Use it to access special features.
Usually you won't need this property.

ConnectAs

Declaration
```
property ConnectAs: TEACConnectAs;
```

Description
The ConnectAs property specifies how will be opened the device.
Possibile values are:
- caNone to use the default state for the device
- caPreview to open the device in the preview state
- caOverlay to open the device in the overlay state

Consider that some devices don't support the overlay or the preview state.

Depth

Declaration
```
property Depth: TEACVideoBits ;
```

Description
The depth property specifies how many bits per pixel will be in the captured image.
Possible values are:
- vd8bits to get 256 colors images
- vd16bits to get 65536 colors images
- vd24bits to get true-color images

Fps

Declaration
```
property Fps: integer;
```

Description
The fps property specifies how many frames per second will be shown or captured.

Consider that some devices will override this value.

IsCapturing

Declaration
```
property IsCapturing: boolean;
```

Description
Run-time and read-only. The IsCapture property lets you know whether the device is capturing or not.

OkToCapture

Declaration
```
property OkToCapture: boolean;
```

Description
The OkToCapture specifies if the user will be prompted to press the OK button when capture starts.

Overlay

Declaration
```
property overlay: boolean;
```

Description
Run-time. The overlay property can be used to set or read the overlay state for an opened device.

Preview

Declaration
```
property preview: boolean;
```

Description
Run-time. The preview property can be used to set or read the preview state for an opened device.

PreviewScaled

Declaration
```
property PreviewScaled: boolean;
```

Description
Run-time. The PreviewScaled property can be used to set or read the scaled state for an open device in the preview state.

Consider that some devices may not be able to handle scaled preview.

VFWVersion

Declaration
```
property VFWVersion: string;
```

Description
Run-time and read-only. The VFWVersion property returns the installed video for windows version. This is the version of the main core of video for windows. Informations about a specific device can be found in the CapDrivers property.

CanOverlay

Declaration
```
function CanOverlay: boolean;
```

Description
Returns true if the opened device is able to show video using overlay.

CaptureStart

Declaration
```
procedure CaptureStart(AFileName: string);
```

Description
Starts recording video to the specified AVI file.

CaptureStop

Declaration
```
procedure CaptureStop;
```

Description
Stops recording video to an AVI file.

CloseVideo

Declaration
```
procedure CloseVideo;
```

Description
Closes the video device previously opened with OpenVideo.

ConnectDriver

Declaration

```
function ConnectDriver(DriverID: integer): boolean;
```

Description

Changes the device presently connected with an open video window by connecting it with another device.

DlgCompression

Declaration
```
function DlgCompression: boolean;
```

Description
Shows the compression dialog owned by the active device and returns true if successfull.

Consider that not all devices support this dialog.

DlgDisplay

Declaration
```
function DlgDisplay: boolean;
```

Description
Shows the display dialog owned by the active device and returns true if successfull.

Consider that not all devices support this dialog.

DlgFormat

Declaration
```
function DlgFormat: boolean;
```

Description
Shows the format dialog owned by the active device and returns true if successfull.

Consider that not all devices support this dialog.

DlgSource

Declaration
```
function DlgSource: boolean;
```

Description
Shows the source dialog owned by the active device and returns true if successfull.

Consider that not all devices support this dialog.

IsOpen

Declaration
```
function IsOpen: boolean;
```

Description
Returns true if a device is currently open.

Consider that it may happen (though I never saw it happening...) that a failed call to ConnectDriver lead to an ambiguous state in which IsOpen returns true, but no device is actually connected.

OpenVideo

Declaration
```
procedure OpenVideo(DriverID: integer);
```

Description
Opens a new window connecting it to the specified driver. You can open any device whose number lies between 0 and CapDrivers.count-1. The window is opened with the size (width and height) of the component and, therefore, you have to size correctly the component BEFORE opening it.

ToClipboard

Declaration
```
function ToClipboard: boolean;
```

Description
Copies to the clipboard the frame that is being shown.

OnFrame

Example

Declaration

```
TOnFrameEvent = procedure(Sender: TObject;h: THandle;VHdr: PVideoHdr) of
object;
property OnFrame: TOnFrameEvent;
```

Description

This event handler receives a handle H to a DIB that contains the frame to be shown.
The TEnhAviCap component does its most to handle compressed and encoded frames and will always return a correct DIB even if the device uses a hardware compressor.
The VHdr structure points to raw data as it is passed back from the device itself.

Use great care with this event as several devices behave differently.

The TVideoHdr structure defines the header used to identify a video-data buffer.

```
PVideoHdr=^TVideoHdr;
TVideoHdr=packed record
            lpData        :pointer;   { pointer to locked data buffer }
            dwBufferLength:longint;   { Length of data buffer }
            dwBytesUsed   :longint;   { Bytes actually used }
            dwTimeCaptured:longint;   { Milliseconds from start of stream }
            dwUser        :longint;   { for client's use }
            dwFlags       :longint;   { assorted flags (see defines) }
            dwReserved    :array[0..3] of longint;  { reserved for driver }
end;  { TVideoHdr }
```

lpData
Specifies a far pointer to the video-data buffer.

dwBufferLength
Specifies the length of the data buffer.

dwBytesUsed
Specifies the number of bytes used in the data buffer.

dwTimeCaptured
Specifies the time (in milliseconds) when the frame was captured relative to the first frame in the stream.

```
procedure TForm1.VideoFrame(Sender: TObject; h: Integer; VHdr: PVideoHdr);
var punt:pointer;
begin
label1.caption:=format('%d %d',[VHdr.dwBufferLength,VHdr.dwBytesUsed]);
punt:=GlobalLock(h);
SetDIBits(img.canvas.handle,img.picture.bitmap.handle,
          0,PBitmapInfoHeader(punt).biHeight,
          pointer(longint(punt)+PBitmapInfoHeader(punt).biSize),
          PBitmapInfo(punt)^,
          DIB_RGB_COLORS);
GlobalUnlock(h);
img.repaint;
end;  { Tform1.VideoFrame }
```