

Robot Project

"RoboTruck BrainStem"

CLAUDIO CONCILIO
from ITALY

claudio.concilio@libero.it

INDEX

[INDEX](#)

[Something about myself and my robotics idea](#)

[My robot's evolution](#)

[The first Robot](#)

[The second Robot](#)

[The third Robot: Description of the RoboTruck Brainstem Robot Project](#)

[A constraint](#)

[What is the Robo Truck able to do ?](#)

[The Visual Basic 5 code examples](#)

[The Desk Laptop server software: The remote Pilot](#)

[The Robot client software: The RobTruck soft-Brain](#)

[The MS Winsock: Client and Server connection via IP protocol](#)

[The SERVER](#)

[The CLIENT](#)

[The MS Speech API \(SAPI\)](#)

[The MS DirectX8 Mouse detector and mapper](#)

[The MS Netmeeting](#)

[The Video Manager for the USB Web Cam](#)

[Vision Recognition and Object Following](#)

[The Brainstem drivers and routines](#)

Something about myself and my robotics idea

Since 4 years ago I had in my mind the idea of add some electronics to a PC in order to act something.

Since 1983, I remember my first Texas Instrument "Home computer" with the beautiful TMS9800 microprocessor. I opened it, the main board at open air and I added an external bus to the processor pins driving RAM banks, leds, switches etc...

The most interesting computing instruction of the first microprocessor equipments were the machine languages and the poke / pick statements of higher level languages.

Without any specialization in electronics or programming languages, my first ideas and experiments evolved requested every time more from the product available on the market.

Now we are since some years in the multimedia age: on the same pc we can have a lot: tv, video, radio, gsm, wireless and why not ... a fully working robot.

My robot's evolution

The possibility to build a Robot raised when I discovered on the Internet some components dedicated and cheap and now the present RoboTruck is my third Robot.

The previous ones were using the FerretTronics IC to drive servos and to receive bunching switch open/close signals via rs232.

It's one year that I discovered Acroname on the Internet (I cannot image to be without it now) and I thought to use the Brainstem instead of others.

My idea is still to use a little MSwindows laptop to drive slave electronics to do something.

No Linux, sorry, no C++, sorry again, simplifying my life: Visual Basic, SDK by Microsoft, OCX from the Internet: my developing platform.

A Casio Fiva MC101 Laptop Windows Millenium based, if closed of one half of an A4 paper: the real hard-brain, my programs the real soft-brain, the knowledge of the external world via sensors and it's more then enough.

The first Robot

My first Robot was FerretTronics driven: three wheels. One central wheel in front with two servos: pan and direct engine for locomotion.

Only dumpers on the four sides with reverse moving reactions.

It was really surprising! It worked, very slowly, but it worked.

The second Robot

An evolution of the same first project, but with a four wheels base extracted from an electric car toy. Better then the previous one, more stable.

Many software ideas were developed in this platform.

ü MSWindows based Speech Recognition and Text-to-Speech

- ü Infrared RX and TX using the electronics of the remote controllers and receivers of the televisions
- ü Ultrasonic RX and TX using a kit
- ü Parallel Port input/output drivers

The project became more impressive

The third Robot: Description of the RoboTruck Brainstem Robot Project

The final idea already tested was as follow:

- ü The big Laptop on the desk with a Joystick, a Wireless card and the right software in order to have on the display the command buttons, the messages and the map of the locomotion of the remote robot. Speech recognition to receive also live voice commands and Text To Speech to inform the remote pilot
- ü The small Laptop on board of a platform, a slave but stable electronics, a web cam with vision recognition, ir sensors at the four sides to measure the near obstacles a shorter range sensor to push and follow an object (an orange ball?), an ultrasonic ranger to detect around open doors or open direction to move to. A PC based autonomous Robot able to move around in stand alone way avoiding obstacles or driven by the remote pilot, but both ways sending messages and showing the locomotion map at the remote display. Speech recognition in order to accept live voice commands, too, text to speech to talk about ... itself and its decisions. Availability to be connected via a Wireless card or via a radio phone (gsm).
- ü Big autonomy: Acid-Lead batteries for the electronics, servos and the electric engine.

A big project, isn't it ?

Just yesterday, March 2 2003, it moves correctly without big mistakes. The project is quite at the end, still under improvements.

A constraint

The heavy (even if small) Laptop, the electronics and the Acid-Lead batteries imposed a strong wheeled frame.

I did choose a four rotating and wheel driven big electric Truck toy by Tamiya: out of production, the representative said.

"But, I have a used Monster Truck by Tamiya for you at half of its price: four wheel driven, only two rotating for changing direction, 10 pounds of load !!!"

Beautiful, I answered, and it was my Truck frame, with the modification of using an electronic speed regulator instead of its controller.

What is the Robo Truck able to do ?

Let me begin from running the software, after having switched on the two Brainstem cards and the electronics, the servos power supplier, the engine.

- ü Connecting to the Laptop on the desk using the MSWinsock: send / receive strings as commands, messages, mapping data, joystick positions etc...
- ü Connecting to the Laptop on the desk using the MSNetMeeting to send the images of the web cam
- ü Testing the Brainstem, the five Ir sensors, the Compass, the Ultrasonic Ranger, the

- engine, the wheel rotating servo
- ü Starting the locomotion with some logical decision on detecting the distances or proximity of obstacles
- ü On remote command switching off the NetMeeting conference (video and Audio) and loading the local video manager, for taking pictures or registering videos using the web cam with pan and tilt
- ü Recognizing a contrasted object like a ball and following it with the web cam acting locomotion to push or be closed to it
- ü Speaking and telling everything detected and taken decisions
- ü Being abandoned by the remote pilot and acting locomotion by itself

And many other features that the other application on the pc could permit. For examples receiving e-mails and reading them calling bthe Internet connection via wireless to the desk laptop, sending e-mails etc...

Instead if the Wireless card (I have only one PCMCIA slot) a GSM card could be inserted to activate a Remote Access service receiving direct calls from a remote modem.

Isn't it great ?

The Visual Basic 5 code examples

I will present examples about some peculiar features on robotics I have experienced with the RoboTruck and the Brainstem cards

- ü The Desk Laptop server software: The remote Pilot
- ü The Robot client software: The RobTruck soft-Brain
- ü The MS Winsock: Client and Server connection via IP protocol
- ü The MS Speech API (SAPI)
- ü The MS Windows Agents
- ü The MS DirectX8 Player for Sounds
- ü The MS DirectX8 Joystick manager
- ü The MS DirectX8 Mouse detector and mapper
- ü The MS Netmeeting
- ü The Video Manager for the USB Web Cam
- ü The Video Recognition and Object Following
- ü The Brainstem drivers and routines

The Desk Laptop server software: The remote Pilot

```
VERSION 5.00
Object = "{5CE55CD7-5179-11D2-931D-0000F875AE17}#1.1#0"; "CONF.EXE"
Object = "{248DD890-BB45-11CF-9ABC-0080C7E7B78D}#1.0#0"; "MSWINSCK.OCX"
Object = "{831FDD16-0C5C-11D2-A9FC-0000F8754DA1}#2.0#0"; "MSCOMCTL.OCX"
```

It's required to add to the VB project some components: in the order Netmeeting, Winsock, VB Comm Controls.

Sliders, Buttons, Checkers, Scroll Bars, Text Boxes, Pictures areas, Timers, shapes, labels are used art pleasure.

The Winsock component:

```
Begin MSWinsockLib.Winsock socket
...
End
```

The Netmeeting component:

```
Begin NetMeetingLibCtl.NetMeeting NetMeeting1
...
End
```

Some API and related Types:

```
Private Type RECT
    left As Long
    top As Long
    right As Long
    bottom As Long
End Type
Dim firsttime As Integer
Private Type POINT_TYPE
    x As Long
    y As Long
End Type
Dim coord As POINT_TYPE
Dim curcoord As POINT_TYPE
Private Declare Sub Sleep Lib "kernel32.dll" (ByVal dwMilliseconds As Long)
Private Declare Function MoveToEx Lib "gdi32.dll" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long, lpPoint As POINT_TYPE) As Long
Private Declare Function SetPixel Lib "gdi32.dll" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long, ByVal crColor As Long) As Long
Private Declare Function GetPixel Lib "gdi32.dll" (ByVal hdc As Long, ByVal nXPos As Long, ByVal nYPos As Long) As Long
Private Declare Function GetWindowRect Lib "user32.dll" (ByVal hwnd As Long, lpRect As RECT) As Long
Private Declare Function LineTo Lib "gdi32.dll" (ByVal hdc As Long, ByVal x As Long, ByVal y As Long) As Long
Private Declare Function SetCursorPos Lib "user32.dll" (ByVal x As Long, ByVal y As Long) As Long
Private Declare Function GetCursorPos Lib "user32.dll" (lpPoint As POINT_TYPE) As Long
' Joystick
```

The Robot client software: The RobTruck soft-Brain

```
VERSION 5.00
Object = "{5CE55CD7-5179-11D2-931D-0000F875AE17}#1.1#0"; "CONF.EXE"
Object = "{F5BE8BC2-7DE6-11D0-91FE-00C04FD701A5}#2.0#0"; "AGENTCTL.DLL"
Object = "{6B7E6392-850A-101B-AFC0-4210102A8DA7}#1.3#0"; "COMCTL32.OCX"
Object = "{648A5603-2C6E-101B-82B6-000000000014}#1.1#0"; "MSCOMM32.OCX"
Object = "{248DD890-BB45-11CF-9ABC-0080C7E7B78D}#1.0#0"; "MSWINSCK.OCX"
Object = "{A723F6EF-A3F6-11D2-905C-008029E9B3A6}#1.0#0"; "EZVIDCAP.OCX"
```

It's required to add to the VB project some components: in the order Netmeeting, the MS Agents, VB Comm Controls, MS Comm for the rs232, the Winsock, the Ocx for the video Capture (see dedicated chapter).

Sliders, Buttons, Checkers, Scroll Bars, Text Boxes, Pictures areas, Timers, shapes, labels are used art pleasure.

The Winsock component:

```
Begin MSWinsockLib.Winsock Winsock1
...
End
```

The Netmeeting component:

```
Begin NetMeetingLibCtl.NetMeeting NetMeeting1
...
End
```

The Video Capture component (see dedicated chapter):

```
Begin vbVidCap.ezVidCap ezVidCap1
...
End
```

The MSComm:

```
Begin MSCommLib.MSComm MSComm1
...
End
```

Some API and related Types:

```
Private Declare Sub Sleep Lib "kernel32.dll" (ByVal dwMilliseconds As Long)
Private Type POINT_TYPE
    x As Long
    y As Long
End Type
Dim coord As POINT_TYPE
Private Declare Function GetCursorPos Lib "user32.dll" (lpPoint As POINT_TYPE) As Long
Private Declare Function SetCursorPos Lib "user32.dll" (ByVal x As Long, ByVal y As Long) As Long
Private Type LARGE_INTEGER
    lowpart As Long
    highpart As Long
End Type
',
Public gRestoreGrammar As Boolean
Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA"
(ByVal hWnd As Long, ByVal lpOperation As String, ByVal lpFile As String, ByVal lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd As Long) As
```

Long

```
Private Declare Function QueryPerformanceCounter Lib "kernel32"
```

```
(lpPerformanceCount As LARGE_INTEGER) As Long
```

```
Private Declare Function SetPixelV Lib "gdi32.dll" (ByVal hdc As Long, ByVal x As  
Long, ByVal y As Long, ByVal crColor As Long) As Long
```

```
Private Declare Function GetPixel Lib "gdi32.dll" (ByVal hdc As Long, ByVal nXPos  
As Long, ByVal nYPos As Long) As Long
```

The MS Winsock: Client and Server connection via IP protocol

The SERVER

The public variables for the Server Winsock:

```
'Server Winsock
Dim iSockets As Integer
Dim sRequestID As String
```

The Winsock Server routines:

Let put the server in listen mode:

```
Private Sub Winsock()
    On Error GoTo errorhandler
    socket(0).LocalPort = 1007
    ListMessage.AddItem "Listening to port: " & socket(0).LocalPort
    socket(0).Listen
End Sub
```

When a connection from the Robot Client is requested, automatically:

```
Private Sub socket_ConnectionRequest(Index As Integer, ByVal requestID As Long)
    If Index = 0 Then
        sRequestID = requestID
        iSockets = iSockets + 1
        Load socket(iSockets)
        socket(iSockets).LocalPort = 1007
        socket(iSockets).Accept requestID
    End If
End Sub
```

When a string data is received, then automatically it must be recognized. Here I put the examples of the three types of received messages: the speed, the position and compass data, the acknowledged locomotion and logic commands.

```
Private Sub socket_DataArrival(Index As Integer, ByVal bytesTotal As Long)
    Dim sItemdata As String
    Dim strData As String
    Dim vtdata As String
    Dim ID As Integer
    Dim fld As Variant
    Dim sdata As String
    Dim i As Integer
    Dim stringa As String
    socket(Index).GetData vtdata, vbString
    ListMessage.AddItem "Ricevuto: " & vtdata & " da " &
socket(Index).RemoteHostIP & " (" & sRequestID & ") "
    sdata = vtdata
    If vbString > 0 Then
        For i = 1 To Len(sdata) Step 4
            stringa = Mid(sdata, i, 4)
            txtReceive.Text = "RX: " + stringa
            If Mid(stringa, 1, 1) = "S" Then
                RecognitionSpeed stringa
            End If
            If Mid(stringa, 1, 1) = "A" Then
```

```

                RecognitionMap stringa, Dir_
            End If
            If Mid(stringa, 1, 1) = "0" Then
                Recognitioncmd stringa
            End If
        Next
    End If
End Sub

```

To send string data:

```

Private Sub cmdSend_click()
If socket(iSockets).State = sckConnected Then
    socket(iSockets).SendData txtTransmit.Text
End If
End Sub

```

To do something or to notice that the connection is closed:

```

Private Sub socket_Close(Index As Integer)
    socket(Index).Close
    Unload socket(Index)
    iSockets = iSockets - 1
End Sub

```

The CLIENT

The public variables for the Client Winsock:

```

'Client Winsock
Dim strOutData As String
Dim iSockets As Integer
Dim sRequestID As String

```

To connect to the server:

```

Private Sub cmdConnect()
    If Not Winsock1.State = sckConnected Then
        Winsock1.RemoteHost = "192.168.0.1" \ wireless IP address
        Winsock1.RemotePort = 1007
        Winsock1.Connect
    End If
End Sub

```

To send string data to the server:

```

Private Sub DataSend(stringa As String)
    If Winsock1.State = sckConnected Then
        Winsock1.SendData stringa
    End If
End Sub

```

When a string data is received, then automatically it must be recognized. Here I put the examples of the four digit string data to be recognized.

```

Private Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
On Error GoTo errorHandler
    Dim sItemdata As String
    Dim strData As String
    Dim vtdata As String
    Dim ID As Integer
    Dim fld As Variant

```

```
Dim sdata As String
Dim i As Integer
Dim stringa As String
Winsock1.GetData vtdata, vbString
sdata = vtdata
    If vbString > 0 Then
        For i = 1 To Len(sdata) Step 4
            stringa = Mid(sdata, i, 4)
            txtReceive.Text = "RX: " + stringa
            Command_Recognition stringa
        Next
    End If
End Sub
```

To do something or to notice that the connection is closed:

```
Private Sub cmdClose()
    If Winsock1.State = sckConnected Then
        Winsock1.Close
    End If
End Sub
```

The MS Speech API (SAPI)

The SAPI 4.0 I used, no migration has been yet provided to SAPI 5.0 or Speech.NET, the last version, and I believe that the Windows Millennium will not support it (or the opposite). The SAPI 4.0 is well working with full functions and I have discovered the same features in Windows XP, but it is another story.

The SAPI 4.0 are still available for downloading with VB documentation and examples at the following site:

www.microsoft.com/speech

An English language speech engine is available.

At IBM.com site, have a search for ViaVoice Runtime library, different languages are available with SAPI 4.0 and higher versions.

Add with it Speech recognition to your robots !!

The MS Windows Agents

Have a look at

www.microsoft.com/msagents

Test them, examples with VB or C++ are available.

Text-To-Speech engines are available for different and many languages.

Speech recognition languages are available at Microsoft.com or at ibm.com.

I used the agent to show on the Robot display the "MS Agent Robot" speaking, showing balloons, into Italian and English.

Enjoy the fun of an agent in your robot applications!!

The MS Directx8 Player for Sounds

For the Directx8 SDK, visit :

<http://www.microsoft.com/msdownload/platformsdk/sdkupdate/>

Public declarations:

```
' _____ DIRECTX8 Player _____  
Dim DirectX As New DirectX8  
'DirectSound Object  
Dim DS As DirectSound8  
'Primary Buffer  
Dim PBuffer As DirectSoundPrimaryBuffer8  
'Secondary Buffer, where the sound is loaded  
Dim DSBuffer As DirectSoundSecondaryBuffer8  
Dim DSBuffer1 As DirectSoundSecondaryBuffer8  
Dim DSBuffer2 As DirectSoundSecondaryBuffer8  
Dim DSBuffer3 As DirectSoundSecondaryBuffer8  
Dim DSBuffer4 As DirectSoundSecondaryBuffer8  
Dim DSBuffer5 As DirectSoundSecondaryBuffer8  
Dim DSBuffer6 As DirectSoundSecondaryBuffer8  
Dim DSBuffer7 As DirectSoundSecondaryBuffer8  
'Direct Sound Buffer Descriptor  
Dim DSDesc As DSBUFFERDESC
```

Initialize the DirectX8 for playing:

```
Private Sub Initialize_Directx()  
    Set DS = DirectX.DirectSoundCreate("")  
    If Err.Number <> 0 Then  
        ListMessage.AddItem "Error init DirectSound"  
    End If  
    DS.SetCooperativeLevel Me.hWnd, DSSCL_NORMAL  
    With DSDesc  
        .lBufferBytes = 0  
        .lFlags = DSBCAPS_PRIMARYBUFFER 'Is a Primary Buffer  
    End With  
    Set PBuffer = DS.CreatePrimarySoundBuffer(DSDesc)  
    DSDesc.lFlags = DSBCAPS_CTRLFREQUENCY _  
    Or DSBCAPS_CTRLPAN Or DSBCAPS_CTRLVOLUME _  
    Or DSBCAPS_STATIC  
End Sub
```

Examples to play an Alarm, stored in Alarm.wav:

```
Private Sub Alarm()  
    Set DSBuffer = DS.CreateSoundBufferFromFile(App.Path & "\Alarm.wav", DSDesc)  
    DSBuffer.Play DSBPLAY_DEFAULT  
    If Err.Number <> 0 Then  
        MsgBox "Error playing sound"  
    End If  
End Sub
```

Routine to close the Directx8:

```
Private Sub Close_Directx8()  
    DSBuffer.Stop  
    Set DSBuffer = Nothing  
    Set PBuffer = Nothing
```

```
Set DS = Nothing
Set DirectX = Nothing
End Sub
```

Add sounds to any action of your Robot !!!

Directx improve ram management and can play at the same time different sounds, also continuously.

The MS DirectX8 Joystick manager

For the DirectX8 SDK, visit :

<http://www.microsoft.com/msdownload/platformsdk/sdkupdate/>

Have a look at the DirectX8 SDK documentation of Microsoft: downloads at Microsoft.com.

The public declarations:

```
' DIRECTX 8 JOYSTICK
Implements DirectXEvent8
Dim dx As New DirectX8
Dim di As DirectInput8
Dim diDev As DirectInputDevice8
Dim diDevEnum As DirectInputEnumDevices8
Dim EventHandle As Long
Dim joyCaps As DIDEVCAPS
Dim js As DIJOYSTATE
Dim DiProp_Dead As DIPROPLONG
Dim DiProp_Range As DIPROPRANGE
Dim DiProp_Saturation As DIPROPLONG
Dim AxisPresent(1 To 8) As Boolean
Dim running As Boolean
```

Loader:

```
Private Sub Load_DirectInput()
    running = True
    InitDirectInput
    Init_MSJoy
End Sub
```

Unloader:

```
Private Sub Unload_DirectInput()
On Local Error Resume Next
    If EventHandle <> 0 Then dx.DestroyEvent EventHandle
    running = False
    'Unacquire if we are holding a device
    If Not diDev Is Nothing Then
        diDev.Unacquire
    End If
    DoEvents
End Sub
```

Initialization:

```
Sub InitDirectInput()
    Set di = dx.DirectInputCreate()
    Set diDevEnum = di.GetDIDevices(DI8DEVCLASS_GAMECTRL, DIEDFL_ATTACHEDONLY)
    If diDevEnum.GetCount = 0 Then
        ListMessage.AddItem "No joystick attached."
        joyon = False
    Else
        ListMessage.AddItem "Joystick attached."
        MsgBox "Regolare SpeedSlider a 0!"
        joyon = True
    End If
    'Add attached joysticks to the listbox
```

```

    Dim i As Integer
    ' Get an event handle to associate with the device
    EventHandle = dx.CreateEvent(Me)
    Exit Sub
Error_Out:
    ListMessage.AddItem "Error initializing DirectInput."
    Unload Me
End Sub

Private Sub DirectXEvent8_DXCallback(ByVal eventid As Long)
' This is called whenever there's a change in the joystick state.
' We check the new state and update the display.
    Dim i As Integer
    Dim ListPos As Integer
    Dim S As String
    Dim str_value As String
    If diDev Is Nothing Then Exit Sub
    '' Get the device info
    On Local Error Resume Next
    diDev.GetDeviceStateJoystick js
    If Err.Number = DIERR_NOTACQUIRED Or Err.Number = DIERR_INPUTLOST Then
        diDev.Acquire
        Exit Sub
    End If
    On Error GoTo err_out
    ' Display axis coordinates
    ListPos = 0
    For i = 1 To 8
        If AxisPresent(i) Then
            Select Case i
                Case 1
                    ' DX SX
                    S = "X: " & js.x
                Case 2
                    ' Backward Forward
                    S = "Y: " & js.y
                ' others I did not used
                Case 3
                    S = "Z: " & js.z
                Case 4
                    S = "RX: " & js.rx
                Case 5
                    S = "RY: " & js.ry
                Case 6
                    S = "RZ: " & js.rz
                Case 7
                    ' Slider
                    S = "Slider0: " & js.slider(0)
                Case 8
                    S = "Slider1: " & js.slider(1)
            End Select
            ListPos = ListPos + 1
        End If
    Next
    ' Buttons I did not used
    If js.Buttons(0) = 128 Then
        If ChkJoy.Value = 1 Then
            ChkJoy.Value = 0
        Else
            ChkJoy.Value = 1
        End If
    End If
End Sub

```

```

        End If
    End If
    ' Hats
    Exit Sub
err_out:
    ListMessage.AddItem Err.Description & " : " & Err.Number + vbApplicationModal
    End
End Sub

```

```

Private Sub Init_MSJoy()
    On Local Error Resume Next
    'Unacquire the current device
    'if we are holding a device
    If Not diDev Is Nothing Then
        diDev.Unacquire
    End If
    'Create the joystick device
    Set diDev = Nothing
    Set diDev = di.CreateDevice(diDevEnum.GetItem(1).GetGuidInstance)
    diDev.SetCommonDataFormat DIFORMAT_JOYSTICK
    diDev.SetCooperativeLevel Me.hwnd, DISCL_BACKGROUND Or DISCL_NONEXCLUSIVE
    ' Find out what device objects it has
    diDev.GetCapabilities joyCaps
    Call IdentifyAxes(diDev)
    ' Ask for notification of events
    Call diDev.SetEventNotification(EventHandle)
    ' Set deadzone for X and Y axis to 10 percent of the range of travel
    With DiProp_Dead
        .lData = 1000
        .lHow = DIPH_BYOFFSET
        .lObj = DIJOFS_X
        diDev.SetProperty "DIPROP_DEADZONE", DiProp_Dead
        .lObj = DIJOFS_Y
        diDev.SetProperty "DIPROP_DEADZONE", DiProp_Dead
    End With
    ' Set saturation zones for X and Y axis to 5 percent of the range
    With DiProp_Saturation
        .lData = 9500
        .lHow = DIPH_BYOFFSET
        .lObj = DIJOFS_X
        diDev.SetProperty "DIPROP_SATURATION", DiProp_Saturation
        .lObj = DIJOFS_Y
        diDev.SetProperty "DIPROP_SATURATION", DiProp_Saturation
    End With
    SetPropRange
    diDev.Acquire
    Me.Caption = "Joystick Sample: Querying Properties"
    ' Get the list of current properties
    ' USB joysticks wont call this callback until you play with the joystick
    ' so we call the callback ourselves the first time
    DirectXEvent8_DXCallback 0
    ' Poll the device so that events are sure to be signaled.
    ' Usually this would be done in Sub Main or in the game rendering loop.
    While running = True
        DoEvents
        diDev.Poll
    Wend
End Sub

```

```

Sub SetPropRange()
' NOTE Some devices do not let you set the range
On Local Error Resume Next
' Set range for all axes
With DiProp_Range
    .lHow = DIPH_DEVICE
    .lMin = 0
    .lMax = 10000
End With
diDev.SetProperty "DIPROP_RANGE", DiProp_Range
End Sub

Sub IdentifyAxes(diDev As DirectInputDevice8)
' It's not enough to count axes; we need to know which in particular
' are present.
Dim didoEnum As DirectInputEnumDeviceObjects
Dim dido As DirectInputDeviceObjectInstance
Dim i As Integer
For i = 1 To 8
    AxisPresent(i) = False
Next
' Enumerate the axes
Set didoEnum = diDev.GetDeviceObjectsEnum(DIDFT_AXIS)
' Check data offset of each axis to learn what it is
Dim sGuid As String
For i = 1 To didoEnum.GetCount
    Set dido = didoEnum.GetItem(i)
    sGuid = dido.GetGuidType
    Select Case sGuid
        Case "GUID_XAxis"
            AxisPresent(1) = True
        Case "GUID_YAxis"
            AxisPresent(2) = True
        Case "GUID_ZAxis"
            AxisPresent(3) = True
        Case "GUID_RxAxis"
            AxisPresent(4) = True
        Case "GUID_RyAxis"
            AxisPresent(5) = True
        Case "GUID_RzAxis"
            AxisPresent(6) = True
        Case "GUID_Slider"
            AxisPresent(8) = True
            AxisPresent(7) = True
    End Select
Next
End Sub

```

The MS Directx8 Mouse detector and mapper

For the Directx8 SDK, visit :

<http://www.microsoft.com/msdownload/platformsdk/sdkupdate/>

Locomotion detection using a mouse which is crawling is the best and easy way to check move, detect speed and detect backward or forward directions.

With the Compass reading (sfr08) for the orientation, it is possible to map on the remote server the position of the robot.

Detecting also obstacles, the door is open to "learning" the environment for the right locomotion.

Isn't it great and easy?

Some public declarations:

```
' _____ DIRECTX Mouse x y Mapper _____  
Const INPUT_LEFTRIGHT_AXIS = 1  
Const INPUT_UPDOWN_AXIS = 2  
Const kMapGuid = "{20CAA014-60BC-4399-BDD3-84AD65A38A1C}"  
Const kUserName = "DInput 8 VB Sample User"  
Const KGenre = DIVIRTUAL_SPACESIM  
Dim m_mapper As New CInputMapper
```

A called routine:

```
Sub DefineActions()  
    m_mapper.ClearMap  
    With m_mapper  
        .AddAction INPUT_LEFTRIGHT_AXIS, DIMOUSE_XAXIS, 0, "Turn"  
        .AddAction INPUT_UPDOWN_AXIS, DIMOUSE_YAXIS, 0, "Move"  
    End With  
End Sub
```

Loading the mapper:

```
Private Sub Load_Mapper_CheckMotore()  
    DefineActions  
    Dim i As Long  
    If Not m_mapper.CreateDevicesFromMAP(Me.hWnd, kUserName, "Semantic Mapper VB  
Sample", kMapGuid, KGenre) Then  
        ListMessage.AddItem "unable to find any mappable input devices"  
    End If  
    Run_InputLoop  
End Sub
```

```
Private Sub Run_InputLoop ()  
    Do While InputLoop ()  
        DoEvents  
    Loop  
End Sub
```

```
Function InputLoop_CheckMotore() As Boolean  
    Dim didObjData As DIDEVICEOBJECTDATA  
    Dim strOut As String
```

```

On Local Error Resume Next
Static dwLRAxisData      As Long
Static dwUDAxisData     As Long
Dim nItems As Long
Dim i As Long, j As Long
Dim adod(10) As DIDEVICEOBJECTDATA
Static nItemsSave(20) As Long
Dim dev As DirectInputDevice8
    For i = 1 To m_mapper.GetNumDevices()
        nItems = 10
        Set dev = m_mapper.GetDevice(i)
        dev.Acquire
        If Err.Number <> 0 Then GoTo skipDevice
        dev.Poll
        If Err.Number <> 0 Then GoTo skipDevice
        nItems = 0
        nItems = dev.GetDeviceData(adod, 0)
        For j = 0 To nItems - 1
            If (adod(j).lUserData = INPUT_LEFTRIGHT_AXIS) Then
                dwLRAxisData = adod(j).lData
            ' one direction detected
            ElseIf (adod(j).lUserData = INPUT_UPDOWN_AXIS) Then
            ' the other direction detected
                dwUDAxisData = adod(j).lData
            End If
        Next
skipDevice:
        Err.Clear
    Next
    InputLoop = True
End Function

```

The addition of a .cls Class module is required:

```

VERSION 1.0 CLASS
BEGIN
    MultiUse = -1 'True
    Persistable = 0 'NotPersistable
    DataBindingBehavior = 0 'vbNone
    DataSourceBehavior = 0 'vbNone
    MTSTransactionMode = 0 'NotAnMTSObject
END
Attribute VB_Name = "CInputMapper"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
Option Explicit
Dim m_NumberOfSemantics As Long
Dim m_diaf As DIACTIONFORMAT
Dim m_DIEnum As DirectInputEnumDevices8
Dim m_Devices(100) As DirectInputDevice8
Dim m_DeviceTypes(100) As Long
Dim m_NumDevices As Long
Dim m_DI As DirectInput8
Dim m_bInit As Boolean
Dim m_hwnd As Long
Dim m_strUserName As String

```

```

Dim m_cdParams As DICONFIGUREDEVICESPARAMS
Function GetDevice(i As Long) As DirectInputDevice8
    Set GetDevice = m_Devices(i)
End Function
Function GetNumDevices() As Long
    GetNumDevices = m_NumDevices
End Function
Function GetDInput() As DirectInput8
    Set GetDInput = m_DI
End Function
Function ConfigureDevices(Optional bAllowEdit = False)
    ReDim m_cdParams.ActionFormats(0)
    ReDim m_cdParams.UserNames(0)
    Dim i As Long
    m_cdParams.ActionFormats(0) = m_diaf
    m_cdParams.FormatCount = 1
    m_cdParams.UserCount = 1
    m_cdParams.UserNames(0) = m_strUserName
    If bAllowEdit Then
        m_DI.ConfigureDevices 0, m_cdParams, DICD_EDIT
    Else
        m_DI.ConfigureDevices 0, m_cdParams, DICD_DEFAULT
    End If
    m_diaf = m_cdParams.ActionFormats(0)
    For i = 1 To m_NumDevices
        If Not m_Devices(i) Is Nothing Then m_Devices(i).Unacquire
        Set m_Devices(i) = Nothing
    Next
    Set m_DIEnum = Nothing
    Dim ret As Long
    ret = CreateDevicesFromMAP(m_hwnd, m_strUserName, m_diaf.ActionMapName, _
        m_diaf.guidActionMap, m_diaf.lGenre, m_diaf.lBufferSize, _
        m_diaf.lAxisMin, m_diaf.lAxisMax)
End Function
Function CreateDevicesFromMAP(hWnd As Long, UserName As String, MapName As String,
MapGuid As String, Genre As CONST_DIGENRE, Optional buffersize = 16, Optional
AxisMin = -100, Optional AxisMax = 100) As Boolean
    On Local Error Resume Next
    Dim i As Long
    m_hwnd = hWnd
    m_strUserName = UserName
    m_diaf.guidActionMap = MapGuid
    m_diaf.lGenre = Genre
    m_diaf.lBufferSize = buffersize
    m_diaf.lAxisMax = AxisMax
    m_diaf.lAxisMin = AxisMin
    m_diaf.ActionMapName = MapName
    Dim dx As DirectX8
    Set dx = New DirectX8
    Set m_DI = dx.DirectInputCreate()
    Set dx = Nothing
    m_diaf.lActionCount = m_NumberofSemantics
    Set m_DIEnum = m_DI.GetDevicesBySemantics(m_strUserName, m_diaf, 0)
    If Err.Number <> 0 Then
        CreateDevicesFromMAP = False
        Exit Function
    End If
    Dim devinst As DirectInputDeviceInstance8
    For i = 1 To m_DIEnum.GetCount
        Set devinst = m_DIEnum.GetItem(i)

```

```

Set m_Devices(i) = m_DI.CreateDevice(devinst.GetGuidInstance)
m_DeviceTypes(i) = devinst.GetDevType
Set devinst = Nothing
If m_DeviceTypes(i) = DI8DEVTYPE_MOUSE Then
    Dim dipl As DIPROPLONG
    dipl.lHow = DIPH_DEVICE
    dipl.lData = DIPROPAXISMODE_REL
    m_Devices(i).SetProperty "DIPROP_AXISMODE", dipl
End If
m_Devices(i).BuildActionMap m_diaf, m_strUserName, 0
m_Devices(i).SetActionMap m_diaf, m_strUserName, 0
m_Devices(i).SetCooperativeLevel m_hwnd, DISCL_EXCLUSIVE Or
DISCL_FOREGROUND
Next
m_NumDevices = m_DIEnum.GetCount
m_bInit = True
CreateDevicesFromMAP = True
End Function
Sub ClearMap()
    On Local Error Resume Next
    Dim i As Long
    m_NumberofSemantics = 0
    ReDim m_diaf.ActionArray(0)
    m_bInit = False
    For i = 0 To m_NumDevices
        If Not m_Devices(i) Is Nothing Then
            m_Devices(i).Unacquire
            Set m_Devices(i) = Nothing
        End If
    Next
    Set m_DI = Nothing
    Set m_DIEnum = Nothing
End Sub
Sub AddAction(user As Long, semantic As Long, flags As Long, strName As String)
    If m_bInit Then
        Debug.Print "can not add actions after CreateDevicesFromMAP has been
called"
        Exit Sub
    End If
    ReDim Preserve m_diaf.ActionArray(m_NumberofSemantics)
    With m_diaf.ActionArray(m_NumberofSemantics)
        .ActionName = strName
        .lAppData = user
        .lFlags = flags
        .lSemantic = semantic
    End With
    m_NumberofSemantics = m_NumberofSemantics + 1
End Sub

```

The MS Netmeeting

Visit

www.microsoft.com/netmeeting

and download the sdk: you will find the conf.exe to add to your project to connect video and audio of your pc based robot to the server or as a Remote Access Service.

The Video Manager for the USB Web Cam

After some search on the Internet, I found a beautiful contribute for captuting and managing image and video using my USB Logitech web Cam.

The useful contribution was coming from:

<http://www.shrinkwrapvb.com/ezvidcap.htm>

DownLoad:

<http://www.shrinkwrapvb.com/ezvidcap60.zip>

for Visual Basic v. 6

<http://www.shrinkwrapvb.com/ezvidcap.zip>

for Visual Basic v. 5

Fantastic !!! I tested the Visual Basic 5 version and now I would like to approach the improvements on v6.

Here I present the version 5 examples.

The world of visioning and vision recognition opened at my mind.

Doing pictures on command.

Doing video on command.

But also, recognition of a contrasted object (an orange ball?) and react to it:

- ü Following by locomotion
- ü Following by Web Cam (the eye)
- ü Pushing
- ü Looking for
- ü Etc.....

Add the follwing mVidPro.BAS module to your Visual Basic project:

```
Attribute VB_Name = "mVidProcessing"
```

```
Option Explicit
```

```
Private Declare Sub CopyMem Lib "kernel32" Alias "RtlMoveMemory" (Destination As Any, Source As Any, ByVal Length As Long)
```

```
*****  
'TFrameBufferInfo based on Win32 VIDEOHDR STRUCT  
'By Ray Mercer Copyright (c) 1997  
*****  
'/* video data block header */  
'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;
Private Type TFrameBufferInfo
    lpData As Long        'locked pointer to frame buffer bits
    dwBufLen As Long
    dwBytesUsed As Long   ' size of locked pointer in bytes
    dwTimeStamp As Long
    dwUser As Long
    dwFlags As Long
    dwReserved(4) As Long
End Type
'//BITMAP DEFINES (from mmsystem.h)
'Public Type TBITMAPINFOHEADER
'    biSize As Long
'    biWidth As Long
'    biHeight As Long
'    biPlanes As Integer
'    biBitCount As Integer
'    biCompression As Long
'    biSizeImage As Long
'    biXPelsPerMeter As Long
'    biYPelsPerMeter As Long
'    biClrUsed As Long
'    biClrImportant As Long
'End Type
'
'Public Type TBITMAPINFO
'    bmiHeader As TBITMAPINFOHEADER
'    bmiColors() As Long 'array of RGBQUADS
'End Type
'
'Public Type TBITMAP
'    bmType As Long
'    bmWidth As Long
'    bmHeight As Long
'    bmWidthBytes As Long
'    bmPlanes As Integer
'    bmBitsPixel As Integer
'    bmBits As Long
'End Type

Public Sub MessWithVidBits(ByVal lpVHdr As Long)

    Static vidFrameBuffer As TFrameBufferInfo

    'copy the VideoHeader data into the VB UDT
    Call CopyMem(vidFrameBuffer, ByVal lpVHdr, Len(vidFrameBuffer))

    'Now we can access the members
    Debug.Print "Buffer Length: " & vidFrameBuffer.dwBufLen
    Debug.Print "Bytes Used: " & vidFrameBuffer.dwBytesUsed
    Debug.Print "Flags: " & vidFrameBuffer.dwFlags
    Debug.Print "TimeStamp: " & vidFrameBuffer.dwTimeStamp
    Debug.Print "User Data: " & vidFrameBuffer.dwUser
    Debug.Print "Pointer To Data: " & vidFrameBuffer.lpData

End Sub

```

Add also the following CmndDlg.BAS module:

```
Attribute VB_Name = "mCmndDlg"
'NOTE! - TO EZVIDCAP TESTERS!!!
'THIS FILE IS ONLY USED FOR SAVE FILE DIALOGS
'YOU DON'T NEED TO BETA-TEST THIS FILE, IT IS
'NOT PART OF ezVidCap.OCX
'IT'S JUST HERE SO THE TEST PROGRAM IS EASIER TO USE
'-RAY

'*****
'*  VB file:   CmndDlg.bas... VB32 wrapper for Win32 common dialog
'*              functions.
'*  created:   1997 by Ray Mercer
'*  modified:  8/98 by Ray Mercer (added browse for folders)
'*  modified:  10/21/98 by Ray Mercer (added comments)
'*  modified:  11/19/98 by Ray Mercer (major enhancements)
'*
'*
'*  original functions based on code found in Bruce McKinney's book
'*  "Hardcore Visual Basic"
'*  enhancements on 11/19/98 based on code by Brad Martinez (especially
'*  useful comments)
'*
'*  Copyright (c) 1997,1998 Ray Mercer.  All rights reserved.
'*****

Option Private Module
Option Explicit

Private Const MAX_PATH = 1024
Private Const MAX_FILE = 512

Private Type SHITEMID
    cb As Long      'Size of the ID (including cb itself)
    abID As Byte    'The item ID (variable length)
End Type
Private Type ITEMIDLIST
    mkid As SHITEMID
End Type

'most of these are also in
'HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell
Folders
Public Enum SPECIAL_FOLDERS
    'Windows desktop virtual folder at the root of the name space
    vbCSIDL_DESKTOP = &H0&    'File system directory that contains the
    'user's program groups (which are also file system directories)
    vbCSIDL_PROGRAMS = &H2&
    'Control Panel - virtual folder containing
    'icons for the control panel applications
    vbCSIDL_CONTROLS = &H3&
    'Printers folder - virtual folder containing installed printers.
    vbCSIDL_PRINTERS = &H4&    'File system directory that serves as a
    'common repository for documents (Documents folder)
```

```

vbCSIDL_PERSONAL = &H5&
'File system directory that contains the
'user's favorite Internet Explorer URLs
vbCSIDL_FAVORITES = &H6&
'File system directory that corresponds to the
'user's Startup program group
vbCSIDL_STARTUP = &H7&
'File system directory that contains the
'user's most recently used documents (Recent folder)
vbCSIDL_RECENT = &H8& 'File system directory that contains
'Send To menu items Public Const
vbCSIDL_SENDTO = &H9&
'Recycle bin file system directory containing file
'objects in the user's recycle bin. The location of
'this directory is not in the registry; it is marked
'with the hidden and system attributes to prevent the
'user from moving or deleting it.
vbCSIDL_BITBUCKET = &HA&
'File system directory containing Start menu items
vbCSIDL_STARTMENU = &HB&
'File system directory used to physically store
'file objects on the desktop (not to be confused
'with the desktop folder itself).
vbCSIDL_DESKTOPDIRECTORY = &H10&
'My Computer - virtual folder containing everything
'on the local computer: storage devices, printers,
'and Control Panel. The folder may also contain 'mapped network drives.
vbCSIDL_DRIVES = &H11&
'Network Neighborhood - virtual folder representing
'the top level of the network hierarchy
vbCSIDL_NETWORK = &H12&
'File system directory containing objects that
'appear in the network neighborhood
vbCSIDL_NETHOOD = &H13&
'Virtual folder containing fonts
vbCSIDL_FONTS = &H14&
'File system directory that serves as a
'common repository for document templates '(ShellNew folder.)
vbCSIDL_TEMPLATES = &H15&
End Enum
Private Declare Function SHGetSpecialFolderLocation Lib "shell32.dll" _
    (ByVal hwndOwner As Long, _
    ByVal nFolder As SPECIAL_FOLDERS,
    _
    pidl As ITEMIDLIST) As Long
'returns NOERROR on success
Public Const NOERROR As Long = 0

Private Type OPENFILENAME
    lStructSize As Long ' Filled with UDT size
    hwndOwner As Long ' Tied to Owner
    hInstance As Long ' Ignored (used only by templates)
    lpstrFilter As String ' Tied to Filter
    lpstrCustomFilter As String ' Ignored (exercise for reader)
    nMaxCustFilter As Long ' Ignored (exercise for reader)
    nFilterIndex As Long ' Tied to FilterIndex
    lpstrFile As String ' Tied to FileName
    nMaxFile As Long ' Handled internally
    lpstrFileTitle As String ' Tied to FileTitle
    nMaxFileTitle As Long ' Handled internally

```

```

lpstrInitialDir As String      ' Tied to InitDir
lpstrTitle As String          ' Tied to DlgTitle
flags As Long                 ' Tied to Flags
nFileOffset As Integer       ' Ignored (exercise for reader)
nFileExtension As Integer    ' Ignored (exercise for reader)
lpstrDefExt As String        ' Tied to DefaultExt
lCustData As Long            ' Ignored (needed for hooks)
lpfnHook As Long             ' Ignored (good luck with hooks)
lpTemplateName As Long      ' Ignored (good luck with templates)
End Type

Private Declare Function GetOpenFileName Lib "COMDLG32" _
    Alias "GetOpenFileNameA" (filestruct As OPENFILENAME) As Long
Private Declare Function GetSaveFileName Lib "COMDLG32" _
    Alias "GetSaveFileNameA" (filestruct As OPENFILENAME) As Long
Private Declare Function GetFileTitle Lib "COMDLG32" _
    Alias "GetFileTitleA" (ByVal szFile As String, _
    ByVal szTitle As String, ByVal cbBuf As Integer) As Integer
'VFW "customized" File Dialogs
Private Declare Function GetOpenFileNamePreview Lib "MSVFW32" _
    Alias "GetOpenFileNamePreviewA" (filestruct As OPENFILENAME) As Long
Private Declare Function GetSaveFileNamePreview Lib "MSVFW32" _
    Alias "GetSaveFileNamePreviewA" (filestruct As OPENFILENAME) As Long

Public Enum EOpenFile
    OFN_READONLY = &H1
    OFN_OVERWRITEPROMPT = &H2
    OFN_HIDEREADONLY = &H4
    OFN_NOCHANGEDIR = &H8
    OFN_SHOWHELP = &H10
    OFN_ENABLEHOOK = &H20
    OFN_ENABLETEMPLATE = &H40
    OFN_ENABLETEMPLATEHANDLE = &H80
    OFN_NOVALIDATE = &H100
    OFN_ALLOWMULTISELECT = &H200
    OFN_EXTENSIONDIFFERENT = &H400
    OFN_PATHMUSTEXIST = &H800
    OFN_FILEMUSTEXIST = &H1000
    OFN_CREATEPROMPT = &H2000
    OFN_SHAREAWARE = &H4000
    OFN_NOREADONLYRETURN = &H8000
    OFN_NOTESTFILECREATE = &H10000
    OFN_NONETWORKBUTTON = &H20000
    OFN_NOLONGNAMES = &H40000
    OFN_EXPLORER = &H80000
    OFN_NODEREFERENCELINKS = &H100000
    OFN_LONGNAMES = &H200000
End Enum

Private Declare Function CommDlgExtendedError Lib "COMDLG32" () As Long

Public Enum EDialogError
    CDERR_DIALOGFAILURE = &HFFFF

    CDERR_GENERALCODES = &H0
    CDERR_STRUCTSIZE = &H1
    CDERR_INITIALIZATION = &H2
    CDERR_NOTEMPLATE = &H3
    CDERR_NOINSTANCE = &H4
    CDERR_LOADSTRFAILURE = &H5

```

```
CDERR_FINDRESFAILURE = &H6
CDERR_LOADRESFAILURE = &H7
CDERR_LOCKRESFAILURE = &H8
CDERR_MEMALLOCFAILURE = &H9
CDERR_MEMLOCKFAILURE = &HA
CDERR_NOHOOK = &HB
CDERR_REGISTERMSGFAIL = &HC
```

```
PDERR_PRINTERCODES = &H1000
PDERR_SETUPFAILURE = &H1001
PDERR_PARSEFAILURE = &H1002
PDERR_RETDEFFFAILURE = &H1003
PDERR_LOADDRVFAILURE = &H1004
PDERR_GETDEVMODEFAIL = &H1005
PDERR_INITFAILURE = &H1006
PDERR_NODEVICES = &H1007
PDERR_NODEFAULTPRN = &H1008
PDERR_DNDMMISMATCH = &H1009
PDERR_CREATEICFAILURE = &H100A
PDERR_PRINTERNOTFOUND = &H100B
PDERR_DEFAULTDIFFERENT = &H100C
```

```
CFERR_CHOOSEFONTCODES = &H2000
CFERR_NOFONTS = &H2001
CFERR_MAXLESSTHANMIN = &H2002
```

```
FNERR_FILENAMECODES = &H3000
FNERR_SUBCLASSFAILURE = &H3001
FNERR_INVALIDFILENAME = &H3002
FNERR_BUFFERTOOSMALL = &H3003
```

```
CCERR_CHOOSECOLORCODES = &H5000
```

End Enum

```
Private Declare Function SHBrowseForFolder Lib "Shell32" (lpbi As TBrowseInfo) As Long
```

```
Private Declare Function SHGetPathFromIDList Lib "Shell32" Alias "SHGetPathFromIDListA" _
```

```
(ByVal pidl As Long, _  
ByVal pszPath As String) As
```

```
Long 'C BOOL returns true on success
```

```
Public Declare Sub CoTaskMemFree Lib "ole32.dll" (ByVal pv As Long)
```

```
Private Type TBrowseInfo
```

```
'Handle of the owner window for the dialog box.
```

```
hwndOwner As Long
```

```
'Pointer to an item identifier list (an
```

```
'ITEMIDLIST structure) specifying the location of
```

```
'the "root" folder to browse from. Only the
```

```
'specified folder and its subfolders appear in the dialog box.
```

```
'This member can be NULL, and in that case, the namespace
```

```
'root (the desktop folder) is used.
```

```
pidlRoot As Long
```

```
'Pointer to a buffer that receives the display
```

```
'name of the folder selected by the user. The
```

```
'size of this buffer is assumed to be MAX_PATH bytes.
```

```
pidlRoot As Long
```

```
'Pointer to a buffer that receives the display
```

```
'name of the folder selected by the user. The
```

```
'size of this buffer is assumed to be MAX_PATH bytes.
```

```

pszDisplayName As String
'Pointer to a null-terminated string that is
'displayed above the tree view control in the
'dialog box.This string can be used to specify
'instructions to the user.
lpszTitle As String
'Value specifying the types of folders to be
'listed in the dialog box as well as other options.
'This member can include zero or more of
'the following values below.
ulFlags As Long
'Address an application-defined function that the
'dialog box calls when events occur. For more information,
'see the description of the BrowseCallbackProc function.
'This member can be NULL. (note: VB4 does not support
'callbacks, therefore this member is ignored.)
lpfn As Long
'Application-defined value that the dialog box
'passes to the callback function (if one is specified).
lParam As Long
'Variable that receives the image associated with
'the selected folder. The image is specified as an
'index to the system image list.
iImage As Long
End Type

Public Enum BROWSE_FLAGS
'default
vbBIF_NONE = &H0&
'Only returns file system directories. If the
'user selects folders that are not part of the
'file system, the OK button is grayed.
vbBIF_RETURNONLYFSDIRS = &H1&
'Does not include network folders below the
'domain level in the tree view control.
vbBIF_DONTGOBELOWDOMAIN = &H2&
'Includes a status area in the dialog box.
'The callback function can set the status
'text by sending messages to the dialog box.
vbBIF_STATUSTEXT = &H4&
'Only returns file system ancestors. If the
'user selects anything other than a file
'system ancestor, the OK button is grayed.
vbBIF_RETURNFNSANCESTORS = &H8&
'Only returns computers. If the user selects
'anything other than a computer, the OK
'button is grayed.
vbBIF_BROWSEFORCOMPUTER = &H1000&
'Only returns (network) printers. If the user
'selects anything other than a printer, the
'OK button is grayed.
vbBIF_BROWSEFORPRINTER = &H2000&
'// Browsing for Everything
vbBIF_BROWSEINCLUDEFILES = &H4000&
End Enum

Private Declare Function SendMessage Lib "user32" Alias "SendMessageA" _
    (ByVal hWnd As Long, _
    ByVal wParam As Long, _
    ByVal lParam As Long, _
    ByVal wMsg As Long) As Long

```

```

Private Const WM_USER As Long = &H400&
Private Const BFFM_INITIALIZED As Long = 1&
'Constants ending in 'A' are for Win95 ANSI
'calls; those ending in 'W' are the wide Unicode'calls for NT.
'Sets the status text to the null-terminated
'string specified by the lParam parameter.
'wParam is ignored and should be set to 0.
Private Const BFFM_SETSTATUSTEXTA As Long = (WM_USER + 100)
Private Const BFFM_SETSTATUSTEXTW As Long = (WM_USER + 104)
'If the lParam parameter is non-zero, enables the
'OK button, or disables it if lParam is zero.'(docs erroneously said wParam!)
'wParam is ignored and should be set to 0.
Private Const BFFM_ENABLEOK As Long = (WM_USER + 101)
'Selects the specified folder. If the wParam
'parameter is FALSE, the lParam parameter is the
'PIDL of the folder to select , or it is the path
'of the folder if wParam is the C value TRUE (or 1).
'Note that after this message is sent, the browse
'dialog receives a subsequent BFFM_SELECTIONCHANGED'message.
Private Const BFFM_SETSELECTIONA As Long = (WM_USER + 102)
Private Const BFFM_SETSELECTIONW As Long = (WM_USER + 103)

```

```
'mem functions
```

```

Private Const LMEM_FIXED As Long = &H0&
Private Const LMEM_ZEROINIT As Long = &H40&
Private Declare Function LocalAlloc Lib "kernel32" _
    (ByVal uFlags As Long, _
    ByVal uBytes As Long) As Long
Private Declare Function LocalFree Lib "kernel32" _
    (ByVal hMem As Long) As Long
Private Declare Sub MoveMemory Lib "kernel32" Alias "RtlMoveMemory" _
    (pDest As Any, _
    pSource As Any, _
    ByVal dwLength As Long)

```

```
Private Const sEmpty As String = ""
```

```

Public Function VBGetOpenFileName(filename As String, _
    Optional FileTitle As String, _
    Optional FileMustExist As Boolean = True, _
    Optional MultiSelect As Boolean = False, _
    Optional ReadOnly As Boolean = False, _
    Optional HideReadOnly As Boolean = False, _
    Optional filter As String = "All (*.*)| *.*", _
    Optional FilterIndex As Long = 1, _
    Optional InitDir As String = "", _
    Optional DlgTitle As String = "", _
    Optional DefaultExt As String = "", _
    Optional Owner As Long = -1, _
    Optional flags As Long = 0) As Boolean

```

```
Dim opfile As OPENFILENAME, s As String, afFlags As Long
```

```
With opfile
```

```
.lStructSize = Len(opfile)
```

```
' Add in specific flags and strip out non-VB flags
```

```
.flags = (-FileMustExist * OFN_FILEMUSTEXIST) Or _
    (-MultiSelect * OFN_ALLOWMULTISELECT) Or _
    (-ReadOnly * OFN_READONLY) Or _
```

```

        (-HideReadOnly * OFN_HIDEREADONLY) Or _
        (flags And CLng(Not (OFN_ENABLEHOOK Or _
                            OFN_ENABLETEMPLATE)))
' Owner can take handle of owning window
If Owner <> -1 Then .hwndOwner = Owner
' InitDir can take initial directory string
.lpstrInitialDir = InitDir
' DefaultExt can take default extension
.lpstrDefExt = DefaultExt
' DlgTitle can take dialog box title
.lpstrTitle = DlgTitle

' To make Windows-style filter, replace | and : with nulls
Dim ch As String, i As Integer
For i = 1 To Len(filter)
    ch = Mid$(filter, i, 1)
    If ch = "|" Or ch = ":" Then
        s = s & vbNullChar
    Else
        s = s & ch
    End If
Next
' Put double null at end
s = s & vbNullChar & vbNullChar
.lpstrFilter = s
.nFilterIndex = FilterIndex

' Pad file and file title buffers to maximum path
s = filename & String$(MAX_PATH - Len(filename), 0)
.lpstrFile = s
.nMaxFile = MAX_PATH
s = FileTitle & String$(MAX_FILE - Len(FileTitle), 0)
.lpstrFileTitle = s
.nMaxFileTitle = MAX_FILE
' All other fields set to zero

If GetOpenFileName(opfile) Then
    VBGetOpenFileName = True
    filename = Left$(.lpstrFile, InStr(.lpstrFile, vbNullChar) - 1)
    FileTitle = Left$(.lpstrFileTitle, InStr(.lpstrFileTitle, vbNullChar) - 1)
    flags = .flags
    ' Return the filter index
    FilterIndex = .nFilterIndex
    ' Look up the filter the user selected and return that
    filter = FilterLookup(.lpstrFilter, FilterIndex)
    If (.flags And OFN_READONLY) Then ReadOnly = True
Else
    VBGetOpenFileName = False
    filename = vbNullChar
    FileTitle = vbNullChar
    flags = 0
    FilterIndex = -1
    filter = vbNullChar
End If
End With
End Function

Public Function VBGetSaveFileName(filename As String, _
    Optional FileTitle As String, _
    Optional OverWritePrompt As Boolean = True, _

```

```

Optional filter As String = "All (*.*)| *.*", _
Optional FilterIndex As Long = 1, _
Optional InitDir As String = "", _
Optional DlgTitle As String = "", _
Optional DefaultExt As String = "", _
Optional Owner As Long = -1, _
Optional flags As Long) As Boolean

Dim opfile As OPENFILENAME, s As String
With opfile
.lStructSize = Len(opfile)

' Add in specific flags and strip out non-VB flags
.flags = (-OverWritePrompt * OFN_OVERWRITEPROMPT) Or _
        OFN_HIDEREADONLY Or _
        (flags And CLng(Not (OFN_ENABLEHOOK Or _
                            OFN_ENABLETEMPLATE)))
' Owner can take handle of owning window
If Owner <> -1 Then .hwndOwner = Owner
' InitDir can take initial directory string
.lpstrInitialDir = InitDir
' DefaultExt can take default extension
.lpstrDefExt = DefaultExt
' DlgTitle can take dialog box title
.lpstrTitle = DlgTitle

' Make new filter with bars (|) replacing nulls and double null at end
Dim ch As String, i As Integer
For i = 1 To Len(filter)
    ch = Mid$(filter, i, 1)
    If ch = "|" Or ch = ":" Then
        s = s & vbNullChar
    Else
        s = s & ch
    End If
Next
' Put double null at end
s = s & vbNullChar & vbNullChar
.lpstrFilter = s
.nFilterIndex = FilterIndex

' Pad file and file title buffers to maximum path
s = filename & String$(MAX_PATH - Len(filename), 0)
.lpstrFile = s
.nMaxFile = MAX_PATH
s = FileTitle & String$(MAX_FILE - Len(FileTitle), 0)
.lpstrFileTitle = s
.nMaxFileTitle = MAX_FILE
' All other fields zero

    VBGetSaveFileName = True

    filename = Day(Date) & Month(Date) & Year(Date) & Hour(Time) &
Minute(Time) & Second(Time) & ".bmp"
    FileTitle = filename

    flags = .flags
' Return the filter index
FilterIndex = .nFilterIndex
' Look up the filter the user selected and return that

```

```

        filter = FilterLookup(.lpstrFilter, FilterIndex)
End With
End Function
Public Function Vision_VBGetSaveFileName(filename As String, _
        Optional FileTitle As String, _
        Optional OverWritePrompt As Boolean = True, _
        Optional filter As String = "All (*.*)| *.*", _
        Optional FilterIndex As Long = 1, _
        Optional InitDir As String = "", _
        Optional DlgTitle As String = "", _
        Optional DefaultExt As String = "", _
        Optional Owner As Long = -1, _
        Optional flags As Long) As Boolean

    Dim opfile As OPENFILENAME, s As String
With opfile
    .lStructSize = Len(opfile)

    ' Add in specific flags and strip out non-VB flags
    .flags = (-OverWritePrompt * OFN_OVERWRITEPROMPT) Or _
        OFN_HIDEREADONLY Or _
        (flags And CLng(Not (OFN_ENABLEHOOK Or _
            OFN_ENABLETEMPLATE)))
    ' Owner can take handle of owning window
    If Owner <> -1 Then .hwndOwner = Owner
    ' InitDir can take initial directory string
    .lpstrInitialDir = InitDir
    ' DefaultExt can take default extension
    .lpstrDefExt = DefaultExt
    ' DlgTitle can take dialog box title
    .lpstrTitle = DlgTitle

    ' Make new filter with bars (|) replacing nulls and double null at end
    Dim ch As String, i As Integer
    For i = 1 To Len(filter)
        ch = Mid$(filter, i, 1)
        If ch = "|" Or ch = ":" Then
            s = s & vbNullChar
        Else
            s = s & ch
        End If
    Next
    ' Put double null at end
    s = s & vbNullChar & vbNullChar
    .lpstrFilter = s
    .nFilterIndex = FilterIndex

    ' Pad file and file title buffers to maximum path
    s = filename & String$(MAX_PATH - Len(filename), 0)
    .lpstrFile = s
    .nMaxFile = MAX_PATH
    s = FileTitle & String$(MAX_FILE - Len(FileTitle), 0)
    .lpstrFileTitle = s
    .nMaxFileTitle = MAX_FILE
    ' All other fields zero

    Vision_VBGetSaveFileName = True

    filename = "Vision.bmp"
    FileTitle = filename

```

```

        flags = .flags
        ' Return the filter index
        FilterIndex = .nFilterIndex
        ' Look up the filter the user selected and return that
        filter = FilterLookup(.lpstrFilter, FilterIndex)
End With
End Function

Private Function FilterLookup(ByVal sFilters As String, ByVal iCur As Long) As
String
    Dim iStart As Long, iEnd As Long, s As String
    iStart = 1
    If sFilters = vbNullChar Then Exit Function
    Do
        ' Cut out both parts marked by null character
        iEnd = InStr(iStart, sFilters, vbNullChar)
        If iEnd = 0 Then Exit Function
        iEnd = InStr(iEnd + 1, sFilters, vbNullChar)
        If iEnd Then
            s = Mid$(sFilters, iStart, iEnd - iStart)
        Else
            s = Mid$(sFilters, iStart)
        End If
        iStart = iEnd + 1
        If iCur = 1 Then
            FilterLookup = s
            Exit Function
        End If
        iCur = iCur - 1
    Loop While iCur
End Function

Public Function VBGetFileTitle(sFile As String) As String
    Dim sFileTitle As String, cFileTitle As Integer

    cFileTitle = MAX_PATH
    sFileTitle = String$(MAX_PATH, 0)
    cFileTitle = GetFileTitle(sFile, sFileTitle, MAX_PATH)
    If cFileTitle Then
        VBGetFileTitle = ""
    Else
        VBGetFileTitle = Left$(sFileTitle, InStr(sFileTitle, vbNullChar) - 1)
    End If
End Function

End Function

Public Function VBGetOpenFileNamePreview(filename As String, _
Optional FileTitle As String, _
Optional FileMustExist As Boolean = True, _
Optional MultiSelect As Boolean = False, _
Optional ReadOnly As Boolean = False, _
Optional HideReadOnly As Boolean = False, _
Optional filter As String = "All (*.*)| *.*", _
Optional FilterIndex As Long = 1, _
Optional InitDir As String, _
Optional DlgTitle As String, _
Optional DefaultExt As String, _
Optional Owner As Long = -1, _
Optional flags As Long = 0) As Boolean

```

```

Dim opfile As OPENFILENAME, s As String, afFlags As Long
With opfile
.lStructSize = Len(opfile)

' Add in specific flags and strip out non-VB flags
.flags = (-FileMustExist * OFN_FILEMUSTEXIST) Or _
        (-MultiSelect * OFN_ALLOWMULTISELECT) Or _
        (-ReadOnly * OFN_READONLY) Or _
        (-HideReadOnly * OFN_HIDEREADONLY) Or _
        (flags And CLng(Not (OFN_ENABLEHOOK Or _
                            OFN_ENABLETEMPLATE)))

' Owner can take handle of owning window
If Owner <> -1 Then .hwndOwner = Owner
' InitDir can take initial directory string
.lpstrInitialDir = InitDir
' DefaultExt can take default extension
.lpstrDefExt = DefaultExt
' DlgTitle can take dialog box title
.lpstrTitle = DlgTitle

' To make Windows-style filter, replace | and : with nulls
Dim ch As String, i As Integer
For i = 1 To Len(filter)
    ch = Mid$(filter, i, 1)
    If ch = "|" Or ch = ":" Then
        s = s & vbNullChar
    Else
        s = s & ch
    End If
Next
' Put double null at end
s = s & vbNullChar & vbNullChar
.lpstrFilter = s
.nFilterIndex = FilterIndex

' Pad file and file title buffers to maximum path
s = filename & String$(MAX_PATH - Len(filename), 0)
.lpstrFile = s
.nMaxFile = MAX_PATH
s = FileTitle & String$(MAX_FILE - Len(FileTitle), 0)
.lpstrFileTitle = s
.nMaxFileTitle = MAX_FILE
' All other fields set to zero

    VBGetOpenFileNamePreview = True
,
    filename = Day(Date) & Month(Date) & Year(Date) & Hour(Time) &
Minute(Time) & Second(Time) & ".bmp"
    FileTitle = filename
    flags = .flags
    ' Return the filter index
    FilterIndex = .nFilterIndex
    ' Look up the filter the user selected and return that
    filter = FilterLookup(.lpstrFilter, FilterIndex)
    If (.flags And OFN_READONLY) Then ReadOnly = True
End With
End Function

Public Function VBGetSaveFileNamePreview(filename As String, _

```

```

Optional FileName As String = "", _
Optional FileMustExist As Boolean = True, _
Optional MultiSelect As Boolean = False, _
Optional ReadOnly As Boolean = False, _
Optional HideReadOnly As Boolean = False, _
Optional filter As String = "All (*.*)| *.*", _
Optional FilterIndex As Long = 1, _
Optional InitDir As String = "", _
Optional DlgTitle As String = "", _
Optional DefaultExt As String = "", _
Optional Owner As Long = -1, _
Optional flags As Long = 0) As Boolean

Dim opfile As OPENFILENAME, s As String, afFlags As Long
With opfile
.lStructSize = Len(opfile)

' Add in specific flags and strip out non-VB flags
.flags = (-FileMustExist * OFN_FILEMUSTEXIST) Or _
        (-MultiSelect * OFN_ALLOWMULTISELECT) Or _
        (-ReadOnly * OFN_READONLY) Or _
        (-HideReadOnly * OFN_HIDEREADONLY) Or _
        (flags And CLng(Not (OFN_ENABLEHOOK Or _
                            OFN_ENABLETEMPLATE)))

' Owner can take handle of owning window
If Owner <> -1 Then .hwndOwner = Owner
' InitDir can take initial directory string
.lpstrInitialDir = InitDir
' DefaultExt can take default extension
.lpstrDefExt = DefaultExt
' DlgTitle can take dialog box title
.lpstrTitle = DlgTitle

' To make Windows-style filter, replace | and : with nulls
Dim ch As String, i As Integer
For i = 1 To Len(filter)
    ch = Mid$(filter, i, 1)
    If ch = "|" Or ch = ":" Then
        s = s & vbNullChar
    Else
        s = s & ch
    End If
Next
' Put double null at end
s = s & vbNullChar & vbNullChar
.lpstrFilter = s
.nFilterIndex = FilterIndex

' Pad file and file title buffers to maximum path
s = filename & String$(MAX_PATH - Len(filename), 0)
.lpstrFile = s
.nMaxFile = MAX_PATH
s = FileName & String$(MAX_FILE - Len(FileName), 0)
.lpstrFileName = s
.nMaxFileName = MAX_FILE
' All other fields set to zero

VBGetSaveFileNamePreview = True

filename = Day(Date) & Month(Date) & Year(Date) & Hour(Time) &

```

```

Minute(Time) & Second(Time) & ".avi"
    FileName = filename

    flags = .flags
    ' Return the filter index
    FilterIndex = .nFilterIndex
    ' Look up the filter the user selected and return that
    filter = FilterLookup(.lpstrFilter, FilterIndex)
    If (.flags And OFN_READONLY) Then ReadOnly = True
End With
End Function

Public Function BrowseForFolder(ByVal hwndOwner As Long, _
    ByVal sTitle As String, _
    Optional ByVal initFolder As String = "", _
    Optional ByVal vRoot As SPECIAL_FOLDERS = _
vbCSIDL_DESKTOP, _
    Optional ByVal flags As BROWSE_FLAGS = vbBIF_NONE)
As String

    Dim BI As TBrowseInfo
    Dim lpsz As Long
    Dim pidl As Long
    Dim sPath As String

    BI.hwndOwner = hwndOwner
    BI.pszDisplayName = String$(MAX_PATH, 0)
    BI.lpszTitle = sTitle
    BI.pidlRoot = vRoot
    BI.ulFlags = flags
    BI.lpfncallback = vbGetProcAddress(AddressOf BrowseCallbackProc)
    If initFolder <> "" Then
        lpsz = LocalAlloc(LMEM_FIXED Or LMEM_ZEROINIT, Len(initFolder))
        Call MoveMemory(ByVal lpsz, ByVal initFolder, Len(initFolder))
        BI.lParam = lpsz
    End If
    'show dialog here
    pidl = SHBrowseForFolder(BI)
    sPath = String$(MAX_PATH, 0)
    Call SHGetPathFromIDList(pidl, sPath)
    If pidl <> 0 Then
        Call CoTaskMemFree(pidl)
        BrowseForFolder = StrZToStr(sPath)
    End If
    If lpsz <> 0 Then
        Call LocalFree(lpsz)
    End If

End Function

'*****
'StrZToStr()
'by Ray Mercer copyright (c) 1997
'converts a C string to a Visual Basic string
'based on a function from "VBPG by Dan Appleman"
'*****
Private Function StrZToStr(s As String) As String
    Dim startp As Integer, endp As Integer
    Dim newString As String

```

```

'
'   startp = 1
'   Do While (Asc(Mid$(s, startp, 1)) <> 0)
'       endp = InStr(startp, s, vbNullChar)
'       If endp = 0 Then StrZToStr = s: Exit Function 'handle VB strings
'       newString = newString & Mid$(s, startp, endp - startp)
'       startp = endp + 1
'   Loop
'   StrZToStr = newString
'different algorithm
Dim TempString As String
TempString = Left$(s, InStr(s, vbNullChar) - 1)
If TempString = "" Then
    'if VB string is accidentally passed in there will be no NULL
    'so just pass back the original string in that case
    StrZToStr = s
Else
    StrZToStr = TempString
End If
End Function

' Test file existence with error trapping
Private Function ExistFile(ByVal sSpec As String) As Boolean
    On Error Resume Next
    Call FileLen(sSpec)
    ExistFile = (Err = 0)
End Function
Private Function vbGetProcAddress(ByVal lpfunc As Long) As Long
    'indirection function for using AddressOf within VB
    vbGetProcAddress = lpfunc
End Function
Private Function BrowseCallbackProc(ByVal hWnd As Long, _
    ByVal uMsg As Long, _
    ByVal lParam As Long, _
    ByVal lpData As Long) As Long
'Callback for the Browse PIDL method.
'On initialization, set the dialog's
'pre-selected folder using the pidl
'set as the bi.lParam, and passed back
'to the callback as lpData param.
Select Case uMsg
    Case BFFM_INITIALIZED
        Call SendMessage(hWnd, BFFM_SETSELECTIONA, _
            1&, ByVal lpData)
    Case Else
        '
    End Select
End Function

```

I modified the previous module if compared to the original you can download, in order to set up the bmp or the avi file name as

ddmmyyyhhmmss.bmp or .avi

picking up the current Date and Time from the computer on the command.

And now the main procedures examples to manage the video with the USB web cam.

The Video Manager Loader:

```
Private Sub Call_videoManager_Load()  
On Error GoTo continue_error  
    If CheckSpeak.value = 1 Then  
        Agente.Speak "Riconoscimento visivo attivo."  
    End If  
    VideoManager_Load  
continue_error:  
End Sub
```

```
Private Sub VideoManager_Load()  
If VideoManagerOn = False Then  
    If Err.Number <> 0 Then  
        Exit Sub  
    End If  
    If Not 0 < ezVidCap1.NumCapDevs Then  
        ListMessage.AddItem "No Video Capture Device!" & vbInformation & App.Title  
    End If  
    ezVidCap1.DriverIndex = 2 ` ATTENTION: The device driver index on my pc;  
modify for yours  
    ezVidCap1.AutoSize = True  
    ezVidCap1.CenterVideo = True  
    ezVidCap1.StretchPreview = False  
    ezVidCap1.CaptureAudio = True  
    ezVidCap1.UsePreciseCaptureControls = False  
    ezVidCap1.MakeUserConfirmCapture = False  
    ezVidCap1.Preview = True  
    ezVidCap1.FrameEventEnabled = False  
    ezVidCap1.Visible = True  
    PicVideo.Visible = True  
    VideoManagerOn = True  
End If  
End Sub
```

Start Video Capturing:

```
Private Sub Inizio_Capture_Video()  
    cmdSTOP_Click  
    Stato_Stop_Stop  
    Click_play  
    If CheckSpeak.value = 1 Then  
        Agente.Speak "Inizio la registrazione del filmato."  
    End If  
    Call ezVidCap1.CaptureVideo  
End Sub
```

End Video Capturing and Saving file:

```
Private Sub Fine_Capture_Video()  
    End_Capture_and_SaveAVI_Video  
End Sub  
  
Private Sub End_Capture_and_SaveAVI_Video()
```

```

Dim filename As String
If mCmnDlg.VBGetSaveFileNamePreview(filename, _
    FileMustExist:=False, _
    filter:="AVI files (*.avi)|*.avi", _
    InitDir:=App.Path, _
    DlgTitle:="Save AVI File", _
    DefaultExt:="avi", _
    Owner:=Me.hWnd) _
    Then
    On Error Resume Next
    Call ezVidCap1.SaveAs(filename)
    If Err Then
        ListMessage.AddItem Err.Description + vbInformation + App.Title
    End If
End If
If CheckSpeak.value = 1 Then
    Agente.Speak "Registrazione terminata."
End If
Click_play
End Sub

```

Capture and saving a picture:

```

Private Sub Capture_SaveDIB_Immagine()
    Dim filename As String
    If CheckSpeak.value = 1 Then
        Agente.Speak "Scatto una fotografia."
    End If
    If mCmnDlg.VBGetSaveFileName(filename, _
        filter:="Bitmap files (*.bmp)|*.bmp", _
        InitDir:=App.Path, _
        DlgTitle:="Save Frame As Bitmap File", _
        DefaultExt:="bmp", _
        Owner:=Me.hWnd) _
        Then
        On Error Resume Next
        Call ezVidCap1.SaveDIB(filename)
        If Err Then
            ListMessage.AddItem Err.Description + vbInformation + App.Title
        End If
    End If
    PicVideo.Picture = LoadPicture(App.Path + "\" + filename)
End Sub

```

Video Management routines:

```

Private Sub ezVidCap1_StatusClear()
End Sub

Private Sub ezVidCap1_CaptureYield()
    'Setting Yield = True will allow this event to be generated
    'but will slow down performance
    'Debug.Print "yield"
    DoEvents
End Sub

Private Sub ezVidCap1_ErrorMessage(ByVal ErrCode As Long, ByVal ErrString As String)
    If ErrCode <> 0 Then
        'Debug.Print ErrString
    End If
End Sub

```

```

    End If
End Sub

Private Sub ezVidCap1_FrameCallback(ByVal lpVHdr As Long)
'Debug.Print "Video frame: " & lpVHdr
    Call MessWithVidBits(lpVHdr)
End Sub

Private Sub ezVidCap1_PreRollComplete()
    Dim userRet As Long

'    userRet = MsgBox("Using precise capture controls." & vbCrLf & _
                    "PreRoll complete - Click OK to start capture
immediately." _
                    , vbOKCancel, App.Title)

    If userRet = vbOK Then
        ezVidCap1.PreciseCaptureStart
    Else
        ezVidCap1.PreciseCaptureCancel
    End If
End Sub

Private Sub ezVidCap1_StatusMessage(ByVal StatCode As Long, ByVal StatString As
String)
ListMessage.AddItem "StatusCode: " & StatCode
If StatCode <> 0 Then
    ListMessage.AddItem StatString
End If
End Sub

Private Sub ezVidCap1_VideoStreamCallback(ByVal lpVHdr As Long)
    ListMessage.AddItem "Video stream: " & lpVHdr
End Sub

Private Sub ezVidCap1_WaveStreamCallback(ByVal lpWHdr As Long)
    ListMessage.AddItem "Wave stream: " & lpWHdr
End Sub

Private Sub VideoManager_Unload()
If VideoManagerOn = True Then
    If Err.Number <> 0 Then
        Exit Sub
    End If
    If Not 0 < ezVidCap1.NumCapDevs Then
        ListMessage.AddItem "No Video Capture Device!" & vbInformation & App.Title
    End If
    ezVidCap1.DriverIndex = 0
    ezVidCap1.AutoSize = True
    ezVidCap1.CenterVideo = True
    ezVidCap1.StretchPreview = False
    ezVidCap1.CaptureAudio = True
    ezVidCap1.UsePreciseCaptureControls = False
    ezVidCap1.MakeUserConfirmCapture = True
    ezVidCap1.Preview = True
    ezVidCap1.FrameEventEnabled = False
    ezVidCap1.Visible = False
    PicVideo.Visible = False
    VideoManagerOn = False
End If
End Sub

```

Vision Recognition and Object Following

Vision Recognition, taking a picture and saving on a file and showing it on the VB form of the application to be manipulated:

Capturing and saving the picture for recognition:

```
Private Sub Vision_Capture_SaveDIB_Imagine()  
    Dim filename As String  
    If CheckSpeak.value = 1 Then  
        Agente.Speak "Scatto una fotografia."  
    End If  
    If mCmnDlg.Vision_VBGetSaveFileName(filename, _  
        filter:="Bitmap files (*.bmp)|*.bmp", _  
        InitDir:=App.Path, _  
        DlgTitle:="Save Frame As Bitmap File", _  
        DefaultExt:".bmp", _  
        Owner:=Me.hWnd) _  
        Then  
        On Error Resume Next  
        Call ezVidCap1.SaveDIB(filename)  
        If Err Then  
            ListMessage.AddItem Err.Description + vbInformation + App.Title  
        End If  
    End If  
    PicVideo.Picture = LoadPicture(App.Path + "\" + filename)  
End Sub  
  
Private Sub cmdVISION_click()  
    Vision_On = True  
    Vision_Local  
End Sub  
  
Private Sub Vision_Local()  
If Vision_On = True Then  
    VideoManager_Load  
    Vision_Capture_SaveDIB_Imagine  
End If  
End Sub  
  
Private Sub Vision_Remote()  
If Vision_On = True Then  
    If NetMeeting1.IsInConference = 1 Then  
        NetMeeting1.LeaveConference  
        NetMeeting1.Visible = False  
        VideoManager_Load  
        Vision_Capture_SaveDIB_Imagine  
        TimerVideo.Interval = 40000  
        TimerVideo.Enabled = True  
        Vision_Recognition_Remote R_obj, G_obj, B_obj  
    Else  
        Vision_Capture_SaveDIB_Imagine  
        TimerVideo.Interval = 40000  
        TimerVideo.Enabled = True  
        Vision_Recognition_Remote R_obj, G_obj, B_obj  
    End If  
End If  
End Sub
```

```

Private Sub TimerVideo_timer()
    TimerVideo.Enabled = False
    Vision_On = False
End Sub

Private Sub Vision_Recognition_Local(R As Integer, G As Integer, B As Integer)
Dim i As Integer
Dim pixhdc As Long
Dim Newcolor As Long
Dim x As Long
Dim y As Long
Dim Rpic As Integer, Gpic As Integer, Bpic As Integer
Dim somma_x As Long, somma_y As Long, xmax As Long, xmin As Long, ymax As Long,
ymin As Long
Dim xc As Long, yc As Long, lx As Long, ly As Long
Dim n As Long
Dim ric_obj As Boolean
Do
    pixhdc = PicVideo.hdc
    DoEvents
    somma_x = 0
    somma_y = 0
    xmax = 0
    xmin = 0
    ymax = 0
    ymin = 0
    n = 0
    For x = 0 To PicVideo.ScaleWidth
        For y = 0 To PicVideo.ScaleHeight
            Newcolor = GetPixel(pixhdc, x, y)
            Rpic = (Newcolor Mod 256)
            Bpic = (Int(Newcolor \ 65536))
            Gpic = ((Newcolor - (Bpic * 65536) - Rpic) \ 256)
            If Rpic > R - 10 And Rpic < R + 10 And Gpic > G - 10 And Gpic < G
+ 10 And Bpic > B - 10 And Bpic < B + 10 Then
                textVision.Text = "Recognized Object"
                n = n + 1
                somma_x = somma_x + x
                somma_y = somma_y + y
                If x > xmax Then
                    xmax = x
                End If
                If x < xmin Then
                    xmin = x
                End If
                If y > ymax Then
                    ymax = y
                End If
                If y < ymin Then
                    ymin = y
                End If
            End If
        Next
    Next
    If n <> 0 Then
        \ Calculate the baricenter (xc and yc) of the shape
        \ Calculate the max length x and y of the shape
        xc = somma_x / n
        yc = somma_y / n
    End If
End Do

```

```

    lx = xmax - xmin
    ly = ymax - ymin
    If Primo_Scatto = False Then
        ric_obj = False
        If xc < PicVideo.ScaleWidth / 2 Then
            \ move camera to right
        End If
        If xc > PicVideo.ScaleWidth / 2 Then
            \ move camera to left
        End If
        If yc > PicVideo.ScaleHeight / 2 Then
            \ move camera down
        End If
        If yc < PicVideo.ScaleHeight / 2 Then
            \ move camera up
        End If
        If lx > lx_old Then
            \ move Robot backward (far)
        End If
        If lx < lx_old Then
            \ move Robot forward (near)
        End If
        If ric_obj = False Then
            \ lost object, do something to search it again
        End If
    End If
    End If
    xc_old = xc
    yc_old = yc
    lx_old = lx
    ly_old = ly
    Primo_Scatto = False
    Vision_Capture_SaveDIB_Imagine
    PicVideo.Refresh
Loop Until Vision_On = False
End Sub

```

```

Private Sub PicVideo_MouseDown(button As Integer, shift As Integer, x As Single, y
As Single)
Dim Newcolor As Long
    Newcolor = PicVideo.Point(x, y)
    Pixel_Get Newcolor
End Sub
Private Sub Pixel_Get(Newcolor As Long)
Dim R As Integer, G As Integer, B As Integer
    R = (Newcolor Mod 256)
    B = (Int(Newcolor \ 65536))
    G = ((Newcolor - (B * 65536) - R) \ 256)
    Rt.Text = R
    Gt.Text = G
    Bt.Text = B
    Primo_Scatto = True
    textVision.Text = "Start Visio Following of the object"
    TimerVideo.Interval = 40000
    TimerVideo.Enabled = True
    Vision_Recognition_Local R, G, B
End Sub

```


The Brainstem drivers and routines

As already done by Steve and Mark @ acronym

<http://www.acroname.com/brainstem/examples/vbtruck/vbtruck.html>