 

'Option Explicit On

'Option Strict On

Public Class frmMain

Inherits System.Windows.Forms.Form

Private Declare Function GetPixel Lib "gdi32" Alias "GetPixel" (ByVal hdc As IntPtr, ByVal x As Int32, ByVal y As Int32) As Int32

'Following are the constants used for setting application priority

Private Const NORMAL\_PRIORITY\_CLASS As Short = &H20S

Private Const IDLE\_PRIORITY\_CLASS As Short = &H40S

Private Const HIGH\_PRIORITY\_CLASS As Short = &H80S

Private Const REALTIME\_PRIORITY\_CLASS As Short = &H100S

Private Const PROCESS\_DUP\_HANDLE As Short = &H40S

Private Const THREAD\_BASE\_PRIORITY\_MAX As Short = 2

'Following are the declaration for windows form and controls

Friend WithEvents HScrollBar1 As System.Windows.Forms.HScrollBar

Friend WithEvents HScrollBar2 As System.Windows.Forms.HScrollBar

Friend WithEvents HScrollBar3 As System.Windows.Forms.HScrollBar

Friend WithEvents Label1 As System.Windows.Forms.Label

Friend WithEvents Label2 As System.Windows.Forms.Label

Friend WithEvents Label3 As System.Windows.Forms.Label

Friend WithEvents MenuItem1 As System.Windows.Forms.MenuItem

Friend WithEvents ColorDialog1 As System.Windows.Forms.ColorDialog

'Declaration of different API used in the project just read it's name,

'U can search on google for details description

Declare Function SetThreadPriority Lib "kernel32"(ByValhThread As Integer, ByVal nPriority As Integer) As Integer

Declare Function SetPriorityClass Lib "kernel32" (ByVal hProcess As Integer, ByVal dwPriorityClass As Integer) As Integer

Declare Function GetCurrentThread Lib "kernel32" () As Integer

Declare Function GetCurrentProcess Lib "kernel32" () As Integer

Declare Function GetObject Lib "gdi32" Alias "GetObjectA" (ByVal hObject As Integer, ByVal nCount As Integer, ByRef lpObject As Bitmap) As Integer

Declare Function GetBitmapBits Lib "gdi32" (ByVal hBitmap As Integer, ByVal dwCount As Integer, ByRef lpBits As Object) As Integer

Declare Function SetBitmapBits Lib "gdi32" (ByVal hBitmap As Integer, ByVal dwCount As Integer, ByRef lpBits As Byte()) As Integer

Dim DPanel As Graphics

Dim up As Point

Dim down As Point

Friend WithEvents MenuItem3 As System.Windows.Forms.MenuItem

Friend WithEvents SaveFileDialog1 As System.Windows.Forms.SaveFileDialog

Friend WithEvents Label4 As System.Windows.Forms.Label

Friend WithEvents picImage As System.Windows.Forms.PictureBox

Dim r As Rectangle

Private myCam As iCam

Private Sub frmMain\_Load() Handles MyBase.Load

'this event will execute on form load / start

'as we need high speed processing so we need to set application priority to real time

'following two line will increase application execution priority

SetThreadPriority(GetCurrentThread, THREAD\_BASE\_PRIORITY\_MAX)

SetPriorityClass(GetCurrentProcess, REALTIME\_PRIORITY\_CLASS)

'set picture box size mode to streach image

Me.picOutput.SizeMode = PictureBoxSizeMode.StretchImage

myCam = New iCam

End Sub

Private Sub mnuExit\_Click() Handles mnuExit.Click

myCam.closeCam()

Application.DoEvents()

myCam = Nothing

End Sub

Private Sub mnuStartCam\_Click() Handles mnuStartCam.Click

myCam.initCam(Me.picOutput.Handle.ToInt32)

tmrUpdate.Enabled = True

End Sub

Private Sub mnuStopCam\_Click() Handles mnuStopCam.Click

tmrUpdate.Enabled = False

myCam.closeCam()

End Sub

Private Sub mnuResetCam\_Click() Handles mnuResetCam.Click

myCam.resetCam()

End Sub

Private Sub mnuSetFrame\_Click() Handles mnuSetFrame.Click

Dim myFrames As String

myFrames = InputBox("Enter Frames Per Second")

myCam.setFrameRate(CInt(myFrames))

End Sub

Private Sub tmrUpdate\_Tick() Handles tmrUpdate.Tick

'this timer code will be executed in loop to process camera view

Call ProcessPixel()

End Sub

Public Sub ProcessPixel()

'actual camera view processing code

Dim i As Integer ' variable for vertical scanning

Dim j As Integer ' variable for Horizontal scanning

Dim bm As Bitmap ' variable use to hold current frame in memory

'Show current status and frame per second rate

sBar.Text = ("Curent FPS = " & myCam.FPS & " " & "Running Status = " & myCam.iRunning)

'load current or copy current frame in BM variable in memory

bm = myCam.copyFrame(Me.picOutput, New RectangleF(0, 0, Me.picOutput.Width, Me.picOutput.Height))

Label4.ForeColor = Color.White 'set default color to normal status

Dim t As Boolean = False

'run loop to process current frame to get each pixel status

For i = 100 To picOutput.Width - 100 Step 1 'Loop to start vertical pixel scan

For j = 50 To picOutput.Height - 50 Step 1 'Loop to start horizontal pixel scan

'X is a color object where it can hold R,G,B for given pixel value

'after executing this line we get

'x.r as red

'x.g as green

'x.b as blue

Dim x As Color = bm.GetPixel(i, j)

'Now compare each value

'compare color with threash hold value for red

If x.R > HScrollBar1.Value And x.G < 75 And x.B < 75 Then

'If matched Then fill color with Red

t = True

Label4.ForeColor = Color.Red 'Show Status

Exit For

End If

'compare color with threash hold value for green

If x.R < 75 And x.G > HScrollBar2.Value And x.B < 75 Then

'If matched Then fill color with green

t = True

Label4.ForeColor = Color.Green 'Show Status

Exit For

End If

'compare color with threash hold value for blue

If x.R < 75 And x.G < 75 And x.B > HScrollBar3.Value Then

'If matched Then fill color with Blue

t = True

Label4.ForeColor = Color.Blue 'Show Status

Exit For

End If

Next

'if red point found exit frame process loop

If t = True Then Exit For

Next

End Sub

Private Sub MenuItem1\_Click() Handles MenuItem1.Click

'call user defined function to show camera source seletion window

Call myCam.ShowWebCamSource()

End Sub

Private Sub Button3\_Click(ByVal)

'clear picture box

DPanel.Clear(Color.White)

End Sub

Private Sub MenuItem3\_Click() Handles MenuItem3.Click

'this function will save current view as a imgae on disk

On Error Resume Next

'set file saving type filter

SaveFileDialog1.Filter = "All Files|\*.\*|JPG|\*.jpg"

'open save file dialog

SaveFileDialog1.ShowDialog()

'capture current view

picImage.Image = myCam.copyFrame(Me.picOutput, New RectangleF(0, 0, Me.picOutput.Width, Me.picOutput.Height))

'save image on disk

picImage.Image.Save(SaveFileDialog1.FileName)

End Sub

End Class

***User Define Module***

'This is user defined module which implements camera capturing and

'frame capturing functions / methods

Public Class iCam

'declaration of windows API function which are utilized in system

Private iDevice As String ' variable use to store driver ID

Private hHwnd As Integer ' variable use to store camera object handle

Private lwndC As Integer

Public iRunning As Boolean

Private CamFrameRate As Integer = 15

Private OutputHeight As Integer = 240

Private OutputWidth As Integer = 360

Public Sub resetCam()

'consider a condition user change the camera setting in that case we need to reset the camera

'so, following function will

'resets the camera after setting change

If iRunning Then

closeCam() ' User define function to close the camera view

Application.DoEvents() ' process request from O.S. If any (Not related to project but this will protect application from termination or crash)

If setCam() = False Then ' Check cam is running or not user defined toggle variable

MessageBox.Show("Errror Setting/Re-Setting Camera") 'show message box with error

End If

End If

End Sub

Public Sub initCam(ByVal parentH As Integer)

'once we got the driver details we need to create it's object so,

'This user defined function will get the camera object from O.S.

'now this object can be used further to start or stop camera CAPTURING

If Me.iRunning = True Then ' Check camera is already running or not

MessageBox.Show("Camera Is Already Running")

Exit Sub

Else

'Following line will get the camera object

'where we need to provide

'iDevice = Driver ID of camera driver

'mode of visibility

'co-ordinates

'height and width

hHwnd = capCreateCaptureWindowA(iDevice, WS\_VISIBLE Or WS\_CHILD, 0, 0, OutputWidth, CShort(OutputHeight), parentH, 0)

If setCam() = False Then ' Set camera initialized sucessfully

MessageBox.Show("Error setting Up Camera")

End If

End If

End Sub

Public Sub setFrameRate(ByVal iRate As Long)

'sets the frame rate of the camera

CamFrameRate = CInt(1000 / iRate)

resetCam()

End Sub

Private Function setCam() As Boolean

'This user defined function will actually start camera capturing not processing

'as we gathered camera object now by sending start signal / message to object

'we can start actual capturing

'send WM\_CAP\_DRIVER\_CONNECT signal to start capturing

If SendMessage(hHwnd, WM\_CAP\_DRIVER\_CONNECT, CShort(iDevice), CType(0, String)) = 1 Then

'Set preview rate that is frame per second

SendMessage(hHwnd, WM\_CAP\_SET\_PREVIEWRATE, CShort(CamFrameRate), CType(0, String))

'set priview mode ON

SendMessage(hHwnd, WM\_CAP\_SET\_PREVIEW, 1, CType(0, String))

Me.iRunning = True 'Toggle variable to set camera is running now

Return True

Else

Me.iRunning = False

Return False

End If

End Function

Public Sub ShowWebCamSize()

'this user defined function will open camera video format

'window where user can set video format

SendMessage(hHwnd, WM\_CAP\_DLG\_VIDEOFORMAT, 0, 0)

End Sub

Public Sub ShowWebCamSource()

'this user defined function will open camera video source

'where user can select camera to user for capturing

'as we can have more that one camera installed in the system

SendMessage(hHwnd, WM\_CAP\_DLG\_VIDEOSOURCE, 0, 0)

End Sub

Public Function closeCam() As Boolean

'Closes the camera means stop CAPTURING

If Me.iRunning Then

'send WM\_CAP\_DRIVER\_DISCONNECT signal to stop camera capturing

'means stop curent view

closeCam = CBool(SendMessage(hHwnd, WM\_CAP\_DRIVER\_DISCONNECT, 0, CType(0, String)))

Me.iRunning = False

End If

End Function

Public Function copyFrame(ByVal src As PictureBox, ByVal rect As RectangleF) As Bitmap

'as we know that camera view is streaming view means array of frame

'for processing the camera view we need to get current image or frame

'so this user defined funtion will used to extract the current frame from

'camera view

If iRunning Then

Dim srcPic As Graphics = src.CreateGraphics 'Create a area to hold graphics in memory

Dim srcBmp As New Bitmap(src.Width, src.Height, srcPic) 'Create a var to hold a current frame in memory

Dim srcMem As Graphics = Graphics.FromImage(srcBmp) 'Copy current frame to memory

Dim HDC1 As IntPtr = srcPic.GetHdc

Dim HDC2 As IntPtr = srcMem.GetHdc

BitBlt(HDC2, 0, 0, CInt(rect.Width), CInt(rect.Height), HDC1, CInt(rect.X), CInt(rect.Y), 13369376)

copyFrame = CType(srcBmp.Clone(), Bitmap)

'Clean Up

srcPic.ReleaseHdc(HDC1)

srcMem.ReleaseHdc(HDC2)

srcPic.Dispose()

srcMem.Dispose()

Else

'MessageBox.Show("Camera Is Not Running!")

End If

End Function

Public Function FPS() As Integer

Return CInt(1000 / (CamFrameRate))

End Function

End Class