

Forum:[Hack Forums](#)

Thread: [Drawing on Custom image control](#)

Doomsapien - July 13, 2013, 11:44 pm

So I'm using this custom image control which allows me to pan and zoom really large images easily (5000px * 5000px) but I can't draw on the image no matter what I do.

[Custom Image Control](#)

Somewhere in the comments the creator says to add:

Code Public Event PaintOverlay(ByVal g As Graphics)
and put this in the onPaint event

Code RaiseEvent PaintOverlay(e.Graphics)

Then add a handler to that to draw on. This doesn't work.

All I have is:

Code Private Sub Map_Paint(ByVal g As Graphics)

g.DrawString("Test string", Font_Calibri, Brush_Red, 50, 50)

End Sub I have tested Map_Paint is getting called, which it is, but no drawing :(

Heres the code for the entire control (plus a few edits by me):

Spoiler ([Click to View](#))

PHP Code

<?php

Imports System.Drawing.Imaging

<System.Runtime.InteropServices.ComVisible(False)> _

Friend Class DrawingBoard

'Public Events

Public Event SetScrollPositions()

Public Event PaintOverlay(ByVal g As Graphics)

'Member Variables

Private m_MouseButtons As System.Windows.Forms.MouseButtons = Windows.Forms.

MouseButtons.Left

Private m_OriginalImage As System.Drawing.Bitmap

Private m_StartPoint As System.Drawing.Point

Private m-Origin As New System.Drawing.Point(0, 0)

Private g As Graphics

Private SrcRect As System.Drawing.Rectangle

Private DestRect As System.Drawing.Rectangle

Private m_ZoomOnMouseWheel As Boolean = True

Private m_ZoomFactor As Double = 1.0

Private m_ApparentImageSize As New Size(0, 0)

```

Private m_DrawWidth As Integer
Private m_DrawHeight As Integer

Private m_centerpoint As Point

Private m_PanMode As Boolean = True
Private m_StretchImageToFit As Boolean = False

Private m_Select_Rect As Rectangle
Private m_Select_Pen As New Pen(Color.Blue, 2)
' Pen used to indicate a selection on the image (zoom window)

Private EndPoint As Point ' for pan and box-zoom

#Region "Public/Private Shadows"
Public Shadows Property Image() As System.Drawing.Image
Get
    Return m_OriginalImage
End Get
Set(ByVal Value As System.Drawing.Image)
    If Not m_OriginalImage Is Nothing Then
        m_OriginalImage.Dispose()
        m_Select_Rect = Nothing
        m-Origin = New Point(0, 0)
        m_ApparentImageSize = New Size(0, 0)
        m_ZoomFactor = 1
        GC.Collect()
    End If

    If Value Is Nothing Then
        m_OriginalImage = Nothing
        Me.Invalidate()
        Exit Property
    End If

    Dim r As New Rectangle(0, 0, Value.Width, Value.Height)
    m_OriginalImage = New Bitmap(Value)
    Dim bmpData As New BitmapData
    m_OriginalImage = DirectCast(m_OriginalImage.Clone(r, Imaging.PixelFormat.
Format32bppPArgb), Bitmap)

    'Force a paint
    Me.Invalidate()
End Set
End Property

Public Shadows Property initialimage() As System.Drawing.Image
Get
    Return m_OriginalImage
End Get
Set(ByVal value As System.Drawing.Image)
    Me.Image = value
    Me.ZoomFactor = 1

```

```

End Set
End Property

Public Shadows Property BackgroundImage() As System.Drawing.Image
Get
Return Nothing
End Get
Set(ByVal Value As System.Drawing.Image)
Me.Image = Value
Me.ZoomFactor = 1
End Set
End Property

```

#End Region

#Region "Protected Overrides"

```

Protected Overrides Sub OnPaint(ByVal e As PaintEventArgs)
e.Graphics.Clear(Me.BackColor)
DrawImage(e.Graphics)
RaiseEvent PaintOverlay(e.Graphics)
MyBase.OnPaint(e)
End Sub

Protected Overrides Sub OnSizeChanged(ByVal e As EventArgs)
DestRect = New System.Drawing.Rectangle(0, 0, ClientSize.Width, ClientSize.Height)
ComputeDrawingArea()
MyBase.OnSizeChanged(e)
End Sub

```

#End Region

#Region "Public Properties"

```

Public Sub ZoomIn()
ZoomImage(True)
End Sub

Public Sub ZoomOut()
ZoomImage(False)
End Sub

Private Sub ZoomImage(ByVal ZoomIn As Boolean)
' Get center point
m_centerpoint.X = m_Origin.X + SrcRect.Width / 2
m_centerpoint.Y = m_Origin.Y + SrcRect.Height / 2

'set new zoomfactor
If ZoomIn Then
ZoomFactor = Math.Round(ZoomFactor * 1.1, 2)
Else
ZoomFactor = Math.Round(ZoomFactor * 0.9, 2)
End If

```

```
'Reset the origin to maintain center point
m_Origin.X = m_centerpoint.X - ClientSize.Width / m_ZoomFactor / 2
m_Origin.Y = m_centerpoint.Y - ClientSize.Height / m_ZoomFactor / 2
```

```
CheckBounds()
End Sub
```

```
Public Sub InvertColors()
Try
Cursor = Cursors.WaitCursor
If m_OriginalImage Is Nothing Then Exit Sub
' This is the color matrix to invert the image colors.
Dim cm As ColorMatrix = New ColorMatrix(New Single()() _
{New Single() {-1, 0, 0, 0, 0}, _
New Single() {0, -1, 0, 0, 0}, _
New Single() {0, 0, -1, 0, 0}, _
New Single() {0, 0, 0, 1, 0}, _
New Single() {1, 1, 1, 1, 1}})
```

```
Dim ImageAttributes As New ImageAttributes()
ImageAttributes.SetColorMatrix(cm)
```

```
Dim g As Graphics
g = Graphics.FromImage(m_OriginalImage)
```

```
Dim rc As New Rectangle(0, 0, m_OriginalImage.Width, m_OriginalImage.Height)
g.DrawImage(m_OriginalImage, rc, 0, 0, m_OriginalImage.Width, m_OriginalImage.Height, GraphicsUnit.Pixel, ImageAttributes)
```

```
Me.Invalidate()
Catch ex As Exception
Throw ex
Finally
Cursor = Cursors.Default
End Try
End Sub
```

```
Public Property PanButton() As System.Windows.Forms.MouseButtons
Get
Return m_MouseButtons
End Get
Set(ByVal value As System.Windows.Forms.MouseButtons)
m_MouseButtons = value
End Set
End Property
```

```
Public Property ZoomOnMouseWheel() As Boolean
Get
Return m_ZoomOnMouseWheel
End Get
Set(ByVal value As Boolean)
m_ZoomOnMouseWheel = value
End Set
```

End Property

Public Property ZoomFactor() As Double

Get

Return m_ZoomFactor

End Get

Set(ByVal value As Double)

m_ZoomFactor = value

If m_ZoomFactor > 15 Then m_ZoomFactor = 15

If m_ZoomFactor < 0.05 Then m_ZoomFactor = 0.05

If Not m_OriginalImage Is Nothing Then

m_ApparentImageSize.Height = m_OriginalImage.Height * m_ZoomFactor

m_ApparentImageSize.Width = m_OriginalImage.Width * m_ZoomFactor

ComputeDrawingArea()

CheckBounds()

End If

Me.Invalidate()

End Set

End Property

Public Property Origin() As System.Drawing.Point

Get

Return m_Origin

End Get

Set(ByVal value As System.Drawing.Point)

m_Origin = value

Me.Invalidate()

End Set

End Property

Public ReadOnly Property ApparentImageSize() As System.Drawing.Size

Get

Return m_ApparentImageSize

End Get

End Property

Public Property PanMode() As Boolean

Get

Return m_PanMode

End Get

Set(ByVal value As Boolean)

m_PanMode = value

End Set

End Property

Public Property StretchImageToFit() As Boolean

Get

Return m_StretchImageToFit

End Get

Set(ByVal value As Boolean)

m_StretchImageToFit = value

Me.Invalidate()

End Set

End Property

```
Public Sub fittoscreen()  
Me.StretchImageToFit = False  
Me.Origin = New Point(0, 0)  
If m_OriginalImage Is Nothing Then Exit Sub  
ZoomFactor = Math.Min(ClientSize.Width / m_OriginalImage.Width, ClientSize.Height / m_OriginalImage.Height)  
End Sub
```

#End Region

```
Private Property Selected_Rectangle() As Rectangle  
Get  
Return m_Select_Rect  
End Get  
Set(ByVal Value As Rectangle)  
m_Select_Rect = Value  
Me.Invalidate()  
End Set  
End Property  
  
Private Sub Draw_Rectangle(ByVal e As System.Windows.Forms.MouseEventArgs)  
If (New Rectangle(0, 0, Me.Width, Me.Height)).Contains(PointToClient(Windows.Forms.Cursor.Position)) Then  
Dim Width As Integer = System.Math.Abs(m_StartPoint.X - e.X)  
Dim Height As Integer = System.Math.Abs(m_StartPoint.Y - e.Y)  
Dim UpperLeft As Point  
'need to determine the upper left corner of the rectangle regardless of whether it's  
'the start point or the end point, or other.  
UpperLeft = New Point(System.Math.Min(m_StartPoint.X, e.X), System.Math.Min(m_StartPoint.Y, e.Y))  
Selected_Rectangle = New Rectangle(UpperLeft.X, UpperLeft.Y, Width, Height)  
End If  
End Sub  
  
Private Sub DrawImage(ByRef g As Graphics)  
If m_OriginalImage Is Nothing Then Exit Sub  
  
g.PixelOffsetMode = Drawing2D.PixelOffsetMode.Half  
g.SmoothingMode = Drawing2D.SmoothingMode.None  
g.InterpolationMode = Drawing2D.InterpolationMode.NearestNeighbor  
  
If m_StretchImageToFit Then  
SrcRect = New System.Drawing.Rectangle(0, 0, m_OriginalImage.Width, m_OriginalImage.Height)  
Else  
SrcRect = New System.Drawing.Rectangle(m_Origin.X, m_Origin.Y, m_DrawWidth, m_DrawHeight)  
End If  
  
g.DrawImage(m_OriginalImage, DestRect, SrcRect, GraphicsUnit.Pixel)
```

```
If Not PanMode Then
    g.DrawRectangle(m_Select_Pen, Selected_Rectangle)
End If
```

```
RaiseEvent SetScrollPositions()
```

```
End Sub
```

```
Private Sub ComputeDrawingArea()
    m_DrawHeight = Me.Height / m_ZoomFactor
    m_DrawWidth = Me.Width / m_ZoomFactor
```

```
End Sub
```

```
Private Sub ImageViewer_MouseDown(ByVal sender As Object, ByVal e As System.Windows.
Forms.MouseEventArgs) Handles Me.MouseDown
    If m_OriginalImage Is Nothing Then Exit Sub
    EndPoint = Nothing
    Selected_Rectangle = Nothing
```

```
'Set the start point. This is used for panning and zooming so we always set it.
m_StartPoint = New Point(e.X, e.Y)
Me.Focus()
End Sub
```

```
Private Sub ImageViewer_MouseMove(ByVal sender As Object, ByVal e As System.Windows.Fo
rms.MouseEventArgs) Handles Me.MouseMove
    If m_OriginalImage Is Nothing Then Exit Sub
    If e.Button = m_MouseButtons Then
```

```
        Dim DeltaX As Integer = m_StartPoint.X - e.X
        Dim DeltaY As Integer = m_StartPoint.Y - e.Y
```

```
        If PanMode Then
            'Set the origin of the new image
```

```
            m-Origin.X = m-Origin.X + (DeltaX / m_ZoomFactor)
            m-Origin.Y = m-Origin.Y + (DeltaY / m_ZoomFactor)
```

```
        CheckBounds()
```

```
        'reset the startpoints
        m_StartPoint.X = e.X
        m_StartPoint.Y = e.Y
```

```
        'Force a paint
        Me.Invalidate()
    Else
        Draw_Rectangle(e)
    End If
End If
End Sub
```

```

Private Sub CheckBounds()
If m_OriginalImage Is Nothing Then Exit Sub
'Make sure we don't go out of bounds
'If m_Origin.X < 0 Then m_Origin.X = 0
'If m_Origin.Y < 0 Then m_Origin.Y = 0
'If m_Origin.X > m_OriginalImage.Width - (ClientSize.Width / m_ZoomFactor) Then
'    m_Origin.X = m_OriginalImage.Width - (ClientSize.Width / m_ZoomFactor)
'End If
'If m_Origin.Y > m_OriginalImage.Height - (ClientSize.Height / m_ZoomFactor) Then
'    m_Origin.Y = m_OriginalImage.Height - (ClientSize.Height / m_ZoomFactor)
'End If

'If m_Origin.X < 0 Then m_Origin.X = 0
'If m_Origin.Y < 0 Then m_Origin.Y = 0
End Sub

Private Sub DrawingBoard_MouseUp(ByVal sender As Object, ByVal e As System.Windows.
Forms.MouseEventHandler) Handles Me.MouseUp
If m_OriginalImage Is Nothing Then Exit Sub
If Not PanMode Then
    EndPoint = New Point(e.X, e.Y)
    If Selected_Rectangle = Nothing Then Exit Sub
    ZoomSelection()
End If
End Sub

Private Sub ZoomSelection()
If m_OriginalImage Is Nothing Then Exit Sub
Try

Dim NewOrigin As New Point(CInt(Me.Origin.X + (Selected_Rectangle.X / ZoomFactor)), _
Me.Origin.Y + (Selected_Rectangle.Y / ZoomFactor))

Dim NewFactor As Double
If Selected_Rectangle.Width > Selected_Rectangle.Height Then
NewFactor = (ClientSize.Width / (Selected_Rectangle.Width / ZoomFactor))
Else
NewFactor = (ClientSize.Height / (Selected_Rectangle.Height / ZoomFactor))
End If

Me.Origin = NewOrigin
Me.ZoomFactor = NewFactor

Catch ex As Exception
    Throw ex
End Try
Selected_Rectangle = Nothing
End Sub

Private Sub ImageViewer_MouseWheel(ByVal sender As Object, ByVal e As System.Windows.
Forms.MouseEventHandler) Handles Me.MouseWheel
If Not ZoomOnMouseWheel Then Exit Sub
'set new zoomfactor

```

```

If e.Delta > 0 Then
    ZoomImage(True)
Elseif e.Delta < 0 Then
    ZoomImage(False)
End If
End Sub

```

```

Public Sub RotateFlip(ByVal RotateFlipType As System.Drawing.RotateFlipType)
If m_OriginalImage Is Nothing Then Exit Sub
m_OriginalImage.RotateFlip(RotateFlipType)
Me.Invalidate()
End Sub

```

```

Public Sub New()

```

```

' This call is required by the Windows Form Designer.
InitializeComponent()

```

```

' Add any initialization after the InitializeComponent() call.
Me.SetStyle(ControlStyles.AllPaintingInWmPaint, True)
Me.SetStyle(ControlStyles.DoubleBuffer, True)
End Sub

```

```

Private Sub DrawingBoard_Resize(ByVal sender As Object, ByVal e As System.EventArgs) Hand
les Me.Resize
Me.ComputeDrawingArea()
If Me.StretchImageToFit Then Me.Invalidate()
End Sub

```

```

End Class

```

More specifically this is the paint event in the control from above:

```

Code Protected Overrides Sub OnPaint(ByVal e As PaintEventArgs)
e.Graphics.Clear(Me.BackColor)
DrawImage(e.Graphics)
RaiseEvent PaintOverlay(e.Graphics)
MyBase.OnPaint(e)
End Sub

```

C3oI - July 14, 2013, 3:17 am

```

Code Dim img As Image = Image.FromFile(location)
Dim image1 As Image = New Bitmap(img.Width, img.Height)
Dim G As Graphics = Graphics.FromImage(image1)
G.DrawImage(img, New Rectangle(0, 0, image1.Width, image1.Height), 0, 0, image1.Width,
image1.Height, GraphicsUnit.Pixel)
img.Dispose()
G.DrawString(Str, New Font("Arial", 20, FontStyle.Bold, GraphicsUnit.Pixel), New
SolidBrush(Color.FromArgb(75, 0, 0, 0)), New Point(x, y))
G.Dispose()

```

Meerkat.? - July 14, 2013, 6:26 am

Perhaps you can just use the paint event instead of the custom one?

Doomsapien - July 14, 2013, 6:44 am

(Jul 14, 2013 06:26 AM)Mavamaarten~ Wrote: [\[CITE\]](#) Perhaps you can just use the paint event instead of the custom one? If you mean just adding a handler to the custom control for paint, that doesn't work. I tried it but I think (specifically because it says Protected Overrides) because of the one in the control, it screws it up.

If you mean putting my code inside the paint event in the control class, I would have to make about a billion variables and functions public which would cause all kinds of problems.

Meerkat.? - July 14, 2013, 12:56 pm

(Jul 14, 2013 06:44 AM)The Inspector Wrote: [\[CITE\]](#) If you mean just adding a handler to the custom control for paint, that doesn't work. I tried it but I think (specifically because it says Protected Overrides) because of the one in the control, it screws it up.

If you mean putting my code inside the paint event in the control class, I would have to make about a billion variables and functions public which would cause all kinds of problems.

I meant just using the paint event like this:

It should work as the paint event (as shown above) is called after drawing everything inside the control.

Doomsapien - July 14, 2013, 6:26 pm

(Jul 14, 2013 12:56 PM)Mavamaarten~ Wrote: [\[CITE\]](#) I meant just using the paint event like this:

It should work as the paint event (as shown above) is called after drawing everything inside the control. I have it slightly different but it should be the same none the less.

I declare the custom image control in the code and initialize it that way instead of actually adding it on the form in the designer. Then I just add a handler to it for the paint event, but it doesn't work.

PHP Code

<?php

Public Map As New DrawingBoard

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)

Handles Me.Load

'Create Map

Map.Size = New Size(1280, 1280)

Map.Image = My.Resources.map

AddHandler Map.Paint, AddressOf Map_Paint

NsTheme1.Controls.Add(Map)

End Sub

Private Sub Map_Paint(ByVal g As Graphics)

g.DrawString("Test string", Font_Calibri, Brush_Red, 50, 50)

End Sub

That should be near identical to what your saying, except instead of having the custom image control in the designer, I'm adding it with code.

This doesn't work. In the comments of the control on the website, the guy who made it says its a problem he has to look into, but clearly never has.

<http://www.codeproject.com/Articles/15743/Pan-and-Zoom-Very-Large-Images?msg=2620311#xx2620311xx>

Ok I had to do this in a new post because everytime I tried to edit my previous message some cloud flare thing popped up telling me to enter a captcha that didn't exist.

I got it working anyway, no idea how. Now I just gotta figure out how to get the drawn images to stick to the main image when it pans and zooms around.

Think I'm just going have to throw this control out the door, seems the way it draws the image makes it impossible for me to draw anything on the image and keep it where I want it!

Meerkat.? - July 15, 2013, 2:48 pm

You can just draw on the Image that is inside the Image property? (From the top of my head:)

```
Code Dim G as graphics = Graphics.FromImage(PictureBox.Image)
G.DrawStuff
```

Doomsapien - July 15, 2013, 10:53 pm

Mava. What would I do without you, you magnificent beast. My only problem is that every thing that I draw looks like total shit.

vs Original

Two very different images. The original has transparency in it and I think thats getting destroyed. Not sure whats going on with the sizing though.

And I've just gone back to my old problem of the image drawing itself a million times because the Imagebox is constantly invalidated. Like the idea is the image is meant to move, but not create a million copies of itself.

Working with graphics is just not fun in anyway.

Meerkat.? - July 16, 2013, 2:29 am

(Jul 15, 2013 10:53 PM)The Inspector Wrote: [\[CITE\]](#) Mava. What would I do without you, you magnificent beast. My only problem is that every thing that I draw looks like total shit.

vs Original

Two very different images. The original has transparency in it and I think thats getting destroyed. Not sure whats going on with the sizing though.

And I've just gone back to my old problem of the image drawing itself a million times because the Imagebox is constantly invalidated. Like the idea is the image is meant to move, but not create a million copies of itself.

Working with graphics is just not fun in anyway.

Ah, I think you're drawing the image many times on top of itself. That's why it looks so weird.

Perhaps you can just "refresh" the image every time you draw on it? (G.Clear(),
G.DrawImage("Original image"), G.DrawImage("the transparent thingy"))

Doomsapian - July 16, 2013, 2:52 am

Using that method, works, but makes it very laggy and slow.

Memory usage goes up about 4x as well.

Its just never easy is it.

I'm thinking something like using the background image property to display my main image, then draw on the image property with my things, but have no idea how to implement this. I think I would have to rewrite some of that control.

Or what about 2 controls overlapping each other one with drawn stuff, the other with the image and they both pan and zoom at the same time.

Yeah I think I really need some sort of overlay layer I can draw to, rather than drawing on the big image.

I don't see a way to implement this though.

?eumonic - July 16, 2013, 3:18 am

Try double buffering your control and see if that helps at all.