

## Re: Transparent bitmaps

**Source:**

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.compactframework/2004-02/0071.htm>

---

**From:** Elisa (*Elisa\_at\_nowhere.net*)

**Date:** 02/02/04

Date: Mon, 02 Feb 2004 13:36:21 GMT

Ed,

Thanks for your answer, I really appreciate someone standing up and defending their product. Sadly, your answer itself just enhances my feelings that you have at several points misjudged the requirements of the .NET Compact Framework development community:

Software development projects these days have ever-tighter deadlines to achieve, product development schedules are measured in days and weeks rather than months and years. (And I sometimes feel that project managers seem to think that because Mobile devices are smaller, writing software for them should take even less time).

And under such thight deadlines, you would want us to find a copy of the GIF89a spec, and develop a library that can interpret this type of file, and draw the relevant pixels on a custom control, and add the correct type of border around it, and show the button differently depending on the state of the button (normal, pressed, focused, enabled, disabled, ...)?

All this just to be able to show a button with an image on it, a feature available in any OS and any RAD tool that's on the market today (and for the last 10 years or so)?!

You don't call that being out-of-touch with the real world?!?

> *Unfortunately, some features just don't make the cut, whether for lack of demand, size, feasibility, or schedule.*

Q: Is there sufficient demand for an ImageButton?

A: Well, for one thing, it is listed on the F.A.Q. pages on the MS website, and F.A.Q. still stands for FREQUENTLY ASKED QUESTIONS. Also, the whole of Windows CE itself uses buttons with images on them all over the place, i.e. commandbar buttons, the input keyboard button, etc. Image buttons make an ideal widget in screenize-constraint devices, it's as simple as that.

Q: How much does it increase the size of the codebase?

A: Can't be a lot, because code to draw images is there already (e.g. ListView, TreeView, ...), code to draw buttons is there already, even code to draw images on buttons is there already (ToolBarButton). In any case, if everyone starts writing their own ImageButton component, surely more memory will be wasted than if an ImageButton component was part of the .NET Compact Framework itself.

Q: How difficult is it to write this feature?

A: No doubt, a fully working, state conveying ImageButton is not an easy thing to write, apparently it involves parsing GIF89a files ;-) But honestly, it is SOOOOO much easier for Microsoft to write it (hey, the code is in the full .NET framework!), then to force every Pocket PC developer to have a go at it themselves!

> *One of the nice things about the .NET Compact Framework is that it's  
> easy to drop in new components, such as the ImageButton provided on  
> GotDotNet*

I beg to differ! Firstly, the PicutreButton on the GotDotNet site is of no use as-is (doesn't convey state, doesn't support transparency, ...). More importantly, you can't just "drop" in a new component, you have to add the control to your form programmatically, because writing Pocket PC components with a design-time interface is extremely elaborate (and isn't even possible with VB.NET)!

> *ImageAttributes is the only feasible way to draw transparency in  
> managed code on the .NET Compact Framework. Otherwise you'd have  
> to go pixel by pixel and only draw the nontransparent ones, which  
> would be slow and painful*

Still, the ListView can do it, and it usually shows multiple images instead of just one. So the underlying code must be available in Windows CE somewhere (I actually know it is, as I used it in a previous eMbedded Visual C++ project).

I'm sorry to disappoint you, but had I known as much about the (lack of) functionality (and robustness) of in the .NET Compact Framework as I do now, I would recommended "pure" eMbedded Visual C++ over .NET anytime. I'm sure with time that will change, but as for now, it is several steps back instead of a leap forward.

Finally, if you take a poll on the 10-ten features missing in the .NET Compact Framework, an ImageButton will be in the top three. Working non-full screen modal dialogs and Thread Synchronization with Time-outs will be in there as well ;-)

Regards,

Elisa

-----  
Ed Kaim [MSFT] wrote:

> I am disappointed by your post because it implies that Microsoft has done  
> something that is negligent or malicious without asking for the whole story.  
> Microsoft is probably the most in-touch company with respect to the  
> developers that build software on its platforms. I was the product manager  
> for the .NET Compact Framework and can assure you that we made some very,  
> very tough decisions on what had to be cut in order to maintain the minimal  
> deployment package with the most functionality. In fact, we had cut the  
> DataGrid before Beta 1, but there was so much developer feedback that we put  
> it back in before release. This was the result of hundreds of customer  
> meetings across the entire product team, as well as the many other feedback  
> channels we use during the development of all of our products.  
> Unfortunately, some features just don't make the cut, whether for lack of  
> demand, size, feasibility, or schedule. One of the nice things about the  
> .NET Compact Framework is that it's easy to drop in new components, such as  
> the ImageButton provided on GotDotNet, for functionality that wasn't  
> included.  
>  
> As for the GIF issue, if you go to  
> <http://www.w3.org/Graphics/GIF/spec-gif89a.txt>, you'll find the spec for  
> GIF89a. The GIF file format provides a list of all the colors each image  
> (GIFs can have multiple images in one file, such as for use in animation)  
> contains. Then, the actual pixels themselves are just index pointers into  
> the list, compressed using LZW. If you do a search within that page for  
> "Transparent Color Index", you'll find that each file also indicates which  
> color based on index, if any, is to be transparent in the the image when  
> drawn. The Microsoft implementation uses this standard, under license from  
> CompuServe, to draw GIF files appropriately. There is no secret here that  
> Microsoft is hiding from the developer community. Anyone can implement this,  
> given they do it under proper license from CompuServe.  
>  
> However, the issue you're raising is why Microsoft recommends the usage of  
> ImageAttributes in order to draw transparency, and why the top left pixel is  
> called out as a prime source. First of all, ImageAttributes is the only  
> feasible way to draw transparency in managed code on the .NET Compact  
> Framework. Otherwise you'd have to go pixel by pixel and only draw the  
> nontransparent ones, which would be slow and painful. The reason the top  
> left pixel is offered is because it's often transparent in images with  
> transparency. Sure, it's not *\*always\** transparent, but it's usually  
> transparent, due to its location in the image and the fact that most  
> pictures on buttons don't go all the way to the top left border. However,  
> you can decide that your standard chroma key color is bright pink (as many  
> do) and hardcode your ImageAttributes object to use that color for all  
> images, provided you do the work in the art to make sure the same exact  
> color is used on the transparent spots in your files, whatever format they  
> may be.  
>  
> I hope this is helpful. If you have any further concerns or questions,

Re: Transparent bitmaps

> please email me by taking the "online." out of edkaim@online.microsoft.com.

>

> "Elisa" <Elisa@nowhere.net> wrote in message

> news:ACoTb.3506\$Xk4.279600@phobos.telenet-ops.be...

>

>>Hi,

>>

>>If you look at the .NET Compact Framework FAQ, and the PictureBox

>>example on the GotDotNet website, Microsoft recommends using Image

>>Attributes to specify a transparent color. They suggest to assume that

>>the upper-left pixel defines which color should be interpreted as

>>transparent.

>>

>>But what makes me wonder is, surely Microsoft themselves know of a better

>>trick, because if you add normal GIF files to an ImageList, and use that

>>ImageList in a ListView, it correctly shows the GIF files without us

>>having to specify which color should be interpreted as transparent, and

>>without relying on the "upper-left pixel is transparent" trick. Thus

>>they must know how to read the transparent color from the GIF files

>>directly.

>>

>>So why won't they share their method with the rest of us?

>>

>>I constantly wonder how much Microsoft must be out of touch with its

>>developer base (=us) to have made such poor decisions regarding what

>>functionality should and shouldn't be dropped from the .NET Framework.

>>About 99% of us .NET Compact Framework developers' first task is to

>>write an ImageButton control, so why Microsoft insists it is "overhead"

>>is just beyond comprehension.

>>

>>

>>Regards,

>>

>>Elisa

>

>

>