Subject: Re: Optimized storage of 1BPP images
Posted by mdelfede on Thu, 07 Feb 2008 13:22:59 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Thu, 07 February 2008 13:47
I had a similar problem with my image scrolling control. Performance was reasonablly good, but when scrolling, it felt like some weight was holding you back and it felt plain wrong to use.

well, it's not so strong as you tell here bu, yes, the feeling is of something holding back

Quote:
I fixed this by creating a permanent preallocated image, the size of the screen, and using it as a back buffer. When zooming for example, you only draw a zone zoomed on the backbuffer, and simply draw a portion as large as the parent control from the backbuffer.

That can't go for my case  I have *very* large images (fax tiffs, thousand's of pixels wide...). With point 2 on previous post I did something like you, just buffering the whole rescaled image. The big problem is that, having a multipage continuous scrolling (if you know evince program in Linux, the same stuff...) I must buffer the whole file in order to have smooth scroll. That works great with small scaled images (but then, it works also without buffering...); the performance on 100% scaled images is not bad at all, but the memory footprint starts to be very high... scaling to 150% it's no more acceptable, being better the performance with rescale-on-the-fly method.

Quote:
About number 2, I don't really know. U++ Images are in RGBA format, and it would be quite some effort to add support for other formats too.

I think so. The best would be bypass completely 'Image' format and to go directly from raster to a packed image format. But then I have to add this format to Rescale() function too, and to Draw() class too.... quite a big work.

Ciao

Max