Subject: Re: bug in ImageBuffer::Line() and operator[]
Posted by mirek on Wed, 12 Mar 2008 13:45:03 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Mon, 10 March 2008 06:19luzr wrote on Mon, 10 March 2008 09:36mdelfede wrote on Sun, 09 March 2008 17:46luzr wrote on Sun, 09 March 2008 22:28mdelfede wrote on Sun, 09 March 2008 16:21AFAIK ImageBuffer::Line[] (and consequently operator[]), which should return a writeable RGBA* do pick the ImageBuffer container
wiping its contents.
That makes Line() and operator[] quite useless....

Ciao

Max

I doubt it. Many things would not work if that would be the case...

Mirek

Uhmmm... I'll check it more in depth.... Somewhere my ImageBuffer is loosing its contents; it seemed to me that was just before getting RGBA pointer, but I may be wrong.

EDIT : Sorry for the wrong bug.... It was again the implied conversion between ImageBuffer and Image which wiped ImageBuffer.
I think we should really add Draw::DrawImage(ImageBuffer...) variants OR at least have some sort of message telling that
ImageBuffer was picked.

Max

I think the main problem here is that it is too easy to use ImageBuffer outside of its domain.

Orginally, this was only meant to be used as local variable to create or alter Image. In that scenario there are little problems... Conversion from/to Image is very fast. However, DrawImage for ImageBuffer would be quite slow.

Mirek

Why ? what I'd do is :

```
void Draw::DrawImage(ImageBuffer & buf.....)
{
  Image Img = buf;
  DrawImage(Img......);
  buf = Img;
```

}

That should work, just a few slower than original but would avoid such problems. IMHO ImageBuffer is very comfortable to create an image on the fly and draw it. Of course we can manually do what I coded above, but if you don't know the problem (or if you just don't think about it as in my case) you have a difficult to find error, as the stuff would work on first display cycle but hang on others.

Max

Well, maybe. I do not really like the idea of non-const ImageBuffer being passed in, but that could be hidden... The only problem I see that painting the same buffer several times would be much slower that doing the same with Image.

Anyway, again, I think the main problem really is that ImageBuffer should be used as local variable only....

Or maybe we could make the situation more clear by introducing

NewImageBuffer (to create a new Image)
ImageBufferOf (to alter existing Image)

and obsoleting the direct use of ImageBuffer.

Mirek