Subject: bug in ImageBuffer::Line() and operator[]
Posted by mdelfede on Sun, 09 Mar 2008 20:21:09 GMT
View Forum Message <> Reply to Message

AFAIK 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

Subject: Re: bug in ImageBuffer::Line() and operator[]
Posted by mirek on Sun, 09 Mar 2008 21:28:23 GMT
View Forum Message <> Reply to Message

mdelfede 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

Subject: Re: bug in ImageBuffer::Line() and operator[]
Posted by mdelfede on Sun, 09 Mar 2008 21:46:27 GMT
View Forum Message <> Reply to Message

luzr 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

Subject: Re: bug in ImageBuffer::Line() and operator[]
Posted by mirek on Mon, 10 Mar 2008 08:36:37 GMT

View Forum Message <> Reply to Message

mdelfede 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

Subject: Re: bug in ImageBuffer::Line() and operator[]
Posted by mdelfede on Mon, 10 Mar 2008 10:19:03 GMT
View Forum Message <> Reply to Message

luzr 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

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

Subject: Re: bug in ImageBuffer::Line() and operator[]
Posted by nixnixnix on Wed, 16 Apr 2008 01:05:02 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Did something change recently in the behaviour of Image?

I have some code which used to work fine (I just checked with an old executable) but now its broken.

I'm getting an error that data=zero when I pass a pointer to an object containing an Image to a new Thread for processing. If I choose to run the static function in the same thread its fine (but obviously I dont want to do that as its clunky) but if I spawn a new thread I can access all other properties of my object except the Image member's data.

Nick

Subject: Re: bug in ImageBuffer::Line() and operator[]
Posted by mirek on Wed, 16 Apr 2008 11:29:16 GMT
View Forum Message <> Reply to Message

No, nothing important has not changed.

Mirek

Subject: Re: bug in ImageBuffer::Line() and operator[]
Posted by nixnixnix on Thu, 17 Apr 2008 19:40:05 GMT
View Forum Message <> Reply to Message

Is there any reason you can think of that the first line below would result in accessing an Image member with no data whilst the second call would work fine? (both versions did work at one point)

```
Thread().Run(callback4(&RasterLayer::MakeImageRasterAsThreadNoKey,pLayer,m_ptr,m_convertImage,0));

RasterLayer::MakeImageRasterAsThreadNoKey(pLayer,m_ptr,m_convertImage,0);
```

Nick

Subject: Re: bug in ImageBuffer::Line() and operator[]
Posted by mirek on Thu, 17 Apr 2008 19:44:44 GMT
View Forum Message <> Reply to Message

Some thread serialization problem perhaps? Hard to say from two lines, but generally, it can be tricky

Mirek

Subject: Re: bug in ImageBuffer::Line() and operator[]
Posted by nixnixnix on Thu, 17 Apr 2008 20:50:06 GMT
View Forum Message <> Reply to Message

Ok it certainly appears to be an issue with Image and threading.

I reverted to a previous version of my software in which the multithreading worked but even then, when I try to repaint the view while the second thread is accessing the Image, I get the same error. It looks as though only one thread is allowed access to the Image at any one time even though both threads are pointing to the same object.

The attached testcase is a bit of a disappointment in that it does not crash but perhaps there is something dodgy about how things are getting done in it so I included it anyway.

I find in my main app that an ASSERT is failing at Image.cpp line 160.

Nick

File Attachments
1) ImageMT.zip, downloaded 511 times

Subject: Re: bug in ImageBuffer::Line() and operator[]
Posted by mirek on Wed, 23 Apr 2008 08:25:34 GMT
View Forum Message <> Reply to Message

nixnixnix wrote on Tue, 15 April 2008 21:05Hi Mirek,

Did something change recently in the behaviour of Image?

I have some code which used to work fine (I just checked with an old executable) but now its broken.

I'm getting an error that data=zero when I pass a pointer to an object containing an Image to a new Thread for processing. If I choose to run the static function in the same thread its fine (but obviously I dont want to do that as its clunky) but if I spawn a new thread I can access all other properties of my object except the Image member's data.

Nick

Is it Win32 or Linux issue? (See my other reply/fix).

Mirek

Subject: Re: bug in ImageBuffer::Line() and operator[]
Posted by mirek on Wed, 23 Apr 2008 08:34:31 GMT

View Forum Message <> Reply to Message

```
void Layer::MakeLayerAsThread(Layer* pLayer,double op,void* pMain,int n)
{
 ImageMT* ptr = (ImageMT*)pMain;

 Layer* pNewLayer = new Layer(pLayer,op,pMain);

 ptr->AddLayer(pNewLayer);
 ptr->Refresh(); //<< !!! You can only do GUI in the main thread!!!
 AtomicDec(ptr->threads); // the last thing this thread does is clean up
}
```

Calls to anything defined in CtrlCore and beyond is only allowed in the main thread, with the exception of event queue calls.

Means that instead of calling Refresh directly, you must use event queue to pass this into the main thread. Best use

```
 void    KillSetTimeCallback(int delay_ms, Callback cb, int id);
```

variant (to remove any other identical Refresh from the queue).

Mirek

Subject: Re: bug in ImageBuffer::Line() and operator[]
Posted by nixnixnix on Wed, 23 Apr 2008 17:18:13 GMT
View Forum Message <> Reply to Message

I am getting the issue under Win32 and it doesn't appear to be related to calling Refresh as I get the same behaviour when I comment it out and I have many other functions which don't suffer from this bug although I will replace these calls with PostCallback() just to be sure.

Basically the Image object causes an ASSERT(data) even though the data object is perfectly valid.

Are threads other than the main thread allowed to make and refer to Image objects?

I don't really think I understand what you're telling me here. I am allowed to use PostCallback in threads other than the main right?

Nick

Subject: Re: bug in ImageBuffer::Line() and operator[]
Posted by mirek on Wed, 23 Apr 2008 19:04:01 GMT
View Forum Message <> Reply to Message

nixnixnix wrote on Wed, 23 April 2008 13:18I am getting the issue under Win32 and it doesn't appear to be related to calling Refresh as I get the same behaviour when I comment it out and I have many other functions which don't suffer from this bug although I will replace these calls with PostCallback() just to be sure.

Basically the Image object causes an ASSERT(data) even though the data object is perfectly valid.

Are threads other than the main thread allowed to make and refer to Image objects?

Yes. Anyway, as always with MT, if multiple threads work with single Image variable, you need to serialize access via Mutex.

Quote:
I don't really think I understand what you're telling me here. I am allowed to use PostCallback in threads other than the main right?

Yes, but ONLY PostCallback (and related event queue functions). Anything else in CtrlCore and beyond (-> CtrlLib, RichEdit... anything that includes CtrlCore) is disallowed.

(Draw, OTOH, you can use in multiple threads).

Mirek