Subject: Color and RGBA
Posted by unodgs on Wed, 22 Feb 2012 11:24:23 GMT
View Forum Message <> Reply to Message

I know this topic was discussed some (long) time ago, but do we really have those two different structures? It makes things more complicated especially if one uses rainbow renderer that supports transparency but due to draw interface he cannot easily use it.

For example I would like to paint transparent red rectangle. Normally I would do:

```
void Paint(Draw& w) {
   w.DrawRect(10, 10, 100, 100, Color(Red, 128));
}
```

Instead I need to cast draw to SystemDraw. In framebuffer it lookes like this:

```
void Paint(Draw& w) {

   SystemDraw& sw = (SystemDraw&) w;
   col.r = Red().GetR();
   col.g = Red().GetG();
   col.b = Red().GetB();
   col.a = 128;
   sw.RectPath(10, 10, 100, 100).Fill(col);
}
```

In WinGL

```
void Paint(Draw& w) {

   SystemDraw& sw = (SystemDraw&) w;
   sw.alpha = 128;
   sw.DrawRect(10, 10, 100, 100, Red);
}
```

Could not Color simply support transparency as RGBA? (in rainbow backend without transparency this part of information from Color would be simply ignored (not used))

PS: I also wonder if it wouldn't be better if Draw interface used doubles instead of integers as coordinates)

Subject: Re: Color and RGBA
Posted by mirek on Thu, 23 Feb 2012 19:52:12 GMT
View Forum Message <> Reply to Message

Well, the main reason for this is that Color supports "special values" while we still want it to be 32bit entity.

OTOH, now thinking about it, it is true that Null could be represented as RGBA(0, 0, 0, 0) and RGBA(x, 0, 0, 0) could in theory represent other special values, because it is invalid in premultiplied format.

We would still need separate RGBA and Color, because RGBA is POD, but I guess things would improve. Also we might have problems with "non-premultiplied" data. Anything else I miss?

Subject: Re: Color and RGBA
Posted by unodgs on Thu, 23 Feb 2012 21:40:15 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 23 February 2012 14:52Well, the main reason for this is that Color supports "special values" while we still want it to be 32bit entity.

OTOH, now thinking about it, it is true that Null could be represented as RGBA(0, 0, 0, 0) and RGBA(x, 0, 0, 0) could in theory represent other special values, because it is invalid in premultiplied format.

We would still need separate RGBA and Color, because RGBA is POD, but I guess things would improve. Also we might have problems with "non-premultiplied" data. Anything else I miss?

As I'm thinking about this I came to the conclusion that the most important is easy of use of the drawing API. What if Draw interface instead of Color class used RGBA structure (like in Painter). RGBA needs only a few useful constructors. According to the POD class definition (http://www.fnal.gov/docs/working-groups/fpcltf/Pkg/ISOcxx/do c/POD.html) structure with a constructor is still a POD. Color class have operator RGBA defined so it could still be easily used with Draw.
I'm not sure if that is a perfect solution but I also know that current one is not good if it comes to transparency support.

Subject: Re: Color and RGBA
Posted by Tom1 on Thu, 23 Feb 2012 21:49:36 GMT
View Forum Message <> Reply to Message

Hi,

I like double coordinates on Painter, but I would still like to preserve integers on Draw :

- Storing doubles takes twice as much memory.
- I have an impression (dating years back though) that computing with doubles is slower than with integers.

- GDI and X are mapped directly to integer

How about offering Painter interface to Ctrls as an option to the Draw version?:

void Paint(Painter &painter);

--

Slightly off-topic: I have recently found that the 2D performance offered by Windows GDI can be attained by software rendering. (Maybe because on Windows Vista/7 GDI is more or less software rendering anyway.) So I have started to render with BufferPainter + SetSurface(). Also having Wayland on its way to Linux, I expect applications will end up rendering display buffers on their own without much support from specific hardware. It seems to me Upp::Painter interface is a nice and clean way to do what is inevitable in not so far future, so why not having a "void Paint(Painter &painter);"?

Best regards,

Tom

Subject: Re: Color and RGBA
Posted by unodgs on Fri, 24 Feb 2012 07:06:26 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Thu, 23 February 2012 16:49Hi,
I like double coordinates on Painter, but I would still like to preserve integers on Draw :

- Storing doubles takes twice as much memory.
- I have an impression (dating years back though) that computing with doubles is slower than with integers.
- GDI and X are mapped directly to integer

I like integers too, but the common resolution for different "Painters" is double. But having doubles is not something I going to insist on.
Quote:
How about offering Painter interface to Ctrls as an option to the
Draw version?:
void Paint(Painter &painter);

And which one should be chosen by CtrlPaint ? (if CtrlPaint would be a part of rainbow then it could make sense) IMO there should be only one Paint to keep things simple. Besides I prefer Draw as its much simpler to implement.

Quote:
Slightly off-topic: I have recently found that the 2D performance offered by Windows GDI can be attained by software rendering. (Maybe because on Windows Vista/7 GDI is more or less

software rendering anyway.) So I have started to render with BufferPainter + SetSurface(). Also having Wayland on its way to Linux, I expect applications will end up rendering display buffers on their own without much support from specific hardware. It seems to me Upp::Painter interface is a nice and clean way to do what is inevitable in not so far future, so why not having a "void Paint(Painter &painter);"?

Mirek tried to replace GDI with software rendering some time ago but at that time WinXP was an os king and GDI was faster on many machines. But maybe it's a good time to look at it one more time. But as I said I'm not fun of Painter to be a common interface for every renderer (or maybe I need to take a closer look at it .

Subject: Re: Color and RGBA
Posted by Tom1 on Fri, 24 Feb 2012 12:32:02 GMT
View Forum Message <> Reply to Message

Still off-topic:

I don't know enough about CtrlPaint internals to comment on that one.

--

It seems, I have given you wrong/misleading information on the GDI performance. Sorry. Here is some more appropriate data on some raw rendering tests I did on Windows 7 x64. The results are in ms:

	thin 1px line	3px line	simple polygons
Draw(GDI)	0.892	10.563	7.02
Painter	36.75	41.200	27.405
GLCtrl	1.722	2.153	9.845
Direct2DAA	87.615	8.667	12.588
Direct2DNOAA	1.649	1.638	9.400
SimpleDraw	2.144	n/a	n/a

I'm quite a bit surprised myself since the difference was not this big in my application. This is probably because the application computes quite a bit when resolving the coordinates for viewing.

Anyway, the results are still in favor of GDI.

SimpleDraw is a simple aliased renderer I made that draws only narrow lines to ImageBuffer. (My application draws a lot of 1px narrow lines, so my point of view is clearly biased here.)

GLCtrl on my platform also draws aliased objects even if smoothing is enabled. Do not know why.

Painter does not improve much if it is turned into NOAA mode.

In my opinion these results do not justify switching away from Draw and GDI, but having SetSurface available and well optimized on any platform is important.

Best regards,

Tom

Subject: Re: Color and RGBA
Posted by mirek on Fri, 24 Feb 2012 13:03:22 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 24 February 2012 07:32

Painter does not improve much if it is turned into NOAA mode.

Well, it is likely because most of code still has to be there, NOAA switches off just small part of rendering.

Subject: Re: Color and RGBA
Posted by mirek on Fri, 24 Feb 2012 13:05:57 GMT
View Forum Message <> Reply to Message

BTW, final buffer rendering in Painter could be parallelized - we could then get quite close to GDI.

Another note: In painter, all lines are drawn as polygons. That is why numbers are so close...

Subject: Re: Color and RGBA
Posted by Tom1 on Fri, 24 Feb 2012 13:57:02 GMT
View Forum Message <> Reply to Message

Mirek,

Sorry, I did not mean to offend you or your Painter in any way. I am a fan of it and I have recently re-implemented a map view using Painter instead of Draw with beautiful results.

I have continued the rendering discussion in the coffee corner where it fits better:

http://www.ultimatepp.org/forum/index.php?t=msg&goto=355 15

Best regards,

Tom

Subject: Re: Color and RGBA
Posted by mirek on Fri, 24 Feb 2012 14:40:11 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 24 February 2012 08:57Mirek,

Sorry, I did not mean to offend you or your Painter in any way.

Oh, no offence taken, I was only thinking about numbers. Actually, they came out better than expected.

Mirek

Subject: Re: Color and RGBA
Posted by mirek on Sat, 03 Mar 2012 23:29:03 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 23 February 2012 14:52Well, the main reason for this is that Color supports "special values" while we still want it to be 32bit entity.

Hm, it actually looks like the only special value "remaining" is InvertColor. Now the question is whether we can affort to represent it as some bitpattern... (most likely A = 0 and nonzero in some of other channels).
