Subject: Painter 2.0
Posted by mirek on Thu, 12 Feb 2009 16:57:30 GMT
View Forum Message <> Reply to Message

I am happy to announce that I have finished the "Painter 2.0", basically a 2D rendering system with PDF/SVG strength, this time written almost from scratch (Painter1.0 was AGG based).

I have only recycled (and optimized) the basic polygon rasterizer from AGG (which, I believe, uses FreeType rasterizer code, which has roots in LibArt) - that algorithm is really unbeatable. Kudos to Raph Levien!

The result is about 10% faster, has about 60% of original code size and, most importantly, is internally much more flexible. I believe Maxim Shemanarev has done some inferior design decisions in AGG, that is now fixed for us

Mirek

Subject: Re: Painter 2.0
Posted by Zardos on Thu, 12 Feb 2009 19:04:56 GMT
View Forum Message <> Reply to Message

luzr wrote on Thu, 12 February 2009 17:57The result is about 10% faster, has about 60% of original code size and, most importantly, is internally much more flexible. I believe Maxim Shemanarev has done some inferior design decisions in AGG, that is now fixed for us

The code looks elegant, without any uneccesary complexity - as nearly everything in upp. A very, very valueable piece of work.

I was looking for such a thing a long time.

Thank you very much for "Painter" !

- Ralf

Subject: Re: Painter 2.0
Posted by cbpporter on Fri, 13 Feb 2009 09:03:09 GMT
View Forum Message <> Reply to Message

Hmmm, looks nice and it does not have that completely unusable interface of AGG. When I used AGG I had the feeling that my entire program consisted only out of typedefs.

I haven't tested it yet, but if you claim it is 10% faster, I believe you . The question is how much can it do compared to AGG? Does it do subpixel rendering and does the result have the same

quality as AGG version? And can it do more fancy stuff, like using a bitmap for a stroke pattern and delivering high quality path adaption while doing it?

I ask this out of curiosity. I don't really need software HQ rendering right now, but it's good to have it at you fingertips.

Subject: Re: Painter 2.0
Posted by mirek on Fri, 13 Feb 2009 10:00:18 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Fri, 13 February 2009 04:03Hmmm, looks nice and it does not have that completely unusable interface of AGG. When I used AGG I had the feeling that my entire program consisted only out of typedefs.

I haven't tested it yet, but if you claim it is 10% faster, I believe you . The question is how much can it do compared to AGG? Does it do subpixel rendering and does the result have the same quality as AGG version?

AGG does not do subpixel rendering (at least AGG2.4 regular version).

Subpixel rendering is using "RGB" subpixel of LCD screen.

What AGG and Painter2.0 do is "subpixel precision" and antialiasing.

I might add subpixel rendering later, it should be quite simple (just another rendering filter .

Quote:
And can it do more fancy stuff, like using a bitmap for a stroke pattern and delivering high quality path adaption while doing it?

It can use bitmap as stroke pattern, but not as stroke 'dash'.

The primary motivation was to match SVG/PDF and that is (more or less) achieved.

Quote:
I ask this out of curiosity. I don't really need software HQ rendering right now, but it's good to have it at you fingertips.

Yep. It will also solve a couple of issues with dependency on system GUI (which is not good for webservers).

Mirek

Subject: Re: Painter 2.0
Posted by cbpporter on Fri, 13 Feb 2009 10:13:34 GMT
View Forum Message <> Reply to Message

Quote:What AGG and Painter2.0 do is "subpixel precision" and antialiasing.
Yeah, that's what I meant.

Subject: Re: Painter 2.0
Posted by emr84 on Fri, 13 Feb 2009 23:31:22 GMT
View Forum Message <> Reply to Message

When I tried to compile PainterExamples (on Windows XP with mingw, Debug mode) in svn 861:

Quote:----- Painter (GUI GCC DEBUG DEBUG_FULL BLITZ WIN32) (2 / 10)
BLITZ: Painter.cpp PainterPath.cpp FontWin32.cpp FontX11.cpp DrawOp.cpp Painting.cpp Math.cpp Xform2D.cpp Approximate.c
 pp Stroker.cpp Dasher.cpp Transformer.cpp Interpolator.cpp Rasterizer.cpp RasterizerClip.cpp Path.cpp Context.cpp I
 mage.cpp Mask.cpp Gradient.cpp RadialGradient.cpp
In file included from C:/uppsvn/out/Painter/MINGW.Debug.Debug_full.Gui\$blitz.cpp: 27:
C:\uppsvn\uppsrc\Painter\Math.cpp: In function 'Upp::Pointf Upp::Mid(const Upp::Pointf&, const Upp::Pointf&)':
C:\uppsvn\uppsrc\Painter\Math.cpp:12: error: ISO C++ says that these are ambiguous, even though the worst conversion fo
 r the first is better than the worst conversion for the second:
C:\uppsvn\uppsrc/Core/Gtypes.h:148: note: candidate 1: Upp::Point_<double> Upp::operator/(Upp::Point_<double>, double)
C:\uppsvn\uppsrc/Core/Gtypes.h:148: note: candidate 2: Upp::Point_<int> Upp::operator/(Upp::Point_<int>, int)
In file included from C:/uppsvn/out/Painter/MINGW.Debug.Debug_full.Gui\$blitz.cpp: 31:
C:\uppsvn\uppsrc\Painter\Xform2D.cpp: In member function 'bool Upp::Xform2D::IsRegular() const':
C:\uppsvn\uppsrc\Painter\Xform2D.cpp:24: error: call of overloaded 'abs(double)' is ambiguous
C:\upp\mingw\include/stdlib.h:369: note: candidates are: int abs(int)
C:\uppsvn\uppsrc/Core/Core.h:484: note: Upp::int64 abs(Upp::int64)
C:\uppsvn\uppsrc\Painter\Xform2D.cpp:24: error: call of overloaded 'abs(const double&)' is ambiguous
C:\upp\mingw\include/stdlib.h:369: note: candidates are: int abs(int)
C:\uppsvn\uppsrc/Core/Core.h:484: note: Upp::int64 abs(Upp::int64)
In file included from C:/uppsvn/out/Painter/MINGW.Debug.Debug_full.Gui\$blitz.cpp: 35:
C:\uppsvn\uppsrc\Painter\Approximate.cpp: In function 'void Upp::ApproximateArc(Upp::LinearPathConsumer&, const Upp::Po
 intf&, const Upp::Pointf&, double, double, double)':
C:\uppsvn\uppsrc\Painter\Approximate.cpp:78: error: call of overloaded 'abs(double)' is ambiguous
C:\upp\mingw\include/stdlib.h:369: note: candidates are: int abs(int)
C:\uppsvn\uppsrc/Core/Core.h:484: note: Upp::int64 abs(Upp::int64)

```
In file included from  C:/uppsvn/out/Painter/MINGW.Debug.Debug_full.Gui\$blitz.cpp: 72:
C:\uppsvn\uppsrc\Painter\Path.cpp: In member function 'virtual void
Upp::BufferPainter::QuadraticOp(const Upp::Pointf&,
  bool)':
C:\uppsvn\uppsrc\Painter\Path.cpp:73: error: ISO C++ says that these are ambiguous, even
though the worst conversion fo
 r the first is better than the worst conversion for the second:
C:\uppsvn\uppsrc/Core/Gtypes.h:146: note: candidate 1: Upp::Point_<double>
Upp::operator*(double, Upp::Point_<double>)
C:\uppsvn\uppsrc/Core/Gtypes.h:146: note: candidate 2: Upp::Point_<int> Upp::operator*(int,
Upp::Point_<int>)
C:\uppsvn\uppsrc\Painter\Path.cpp: In member function 'virtual void
Upp::BufferPainter::CubicOp(const Upp::Pointf&, con
 st Upp::Pointf&, bool)':
C:\uppsvn\uppsrc\Painter\Path.cpp:87: error: ISO C++ says that these are ambiguous, even
though the worst conversion fo
 r the first is better than the worst conversion for the second:
C:\uppsvn\uppsrc/Core/Gtypes.h:146: note: candidate 1: Upp::Point_<double>
Upp::operator*(double, Upp::Point_<double>)
C:\uppsvn\uppsrc/Core/Gtypes.h:146: note: candidate 2: Upp::Point_<int> Upp::operator*(int,
Upp::Point_<int>)
Fillers.cpp
Render.cpp
PaintPainting.icpp
Painter: 24 file(s) built in (0:15.37), 640 msecs / file, duration = 16047 msecs

There were errors. (2:43.95)
```

Subject: Re: Painter 2.0
Posted by mirek on Sun, 15 Feb 2009 13:13:51 GMT
View Forum Message <> Reply to Message

Well, I have just spend a hour fixing Painter to work in Linux (well, rather in GCC).

Should be OK by now.

Mirek

Subject: Re: Painter 2.0
Posted by kodos on Sun, 15 Feb 2009 14:24:53 GMT
View Forum Message <> Reply to Message

I took a quick look at painter and I have to say: wow!
I think I have to ditch cairo, and use painter because it's really fast, and the code is quite small

compared to cairo

Good work!

Subject: Re: Painter 2.0
Posted by emr84 on Mon, 16 Feb 2009 22:34:10 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 15 February 2009 11:13Well, I have just spend a hour fixing Painter to work in Linux (well, rather in GCC).

Should be OK by now.

Mirek

Now it works!! Thanks!!

Subject: Re: Painter 2.0
Posted by Tom1 on Tue, 17 Feb 2009 09:35:15 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Painter 2.0 was a welcome enhancement. Thank you!

I recall you were earlier referring to a future CoreDraw and more. Could you outline the future you have planned for Draw and Painter interfaces?

As I understand it, currently the Draw interface can be used to render graphics on screen, images and printer using the hardware accelerated graphics of the system where available.

Now, the Painter interface can be used to render graphics on ImageBuffers without hardware acceleration.

Is there a plan to create a Painter interface for Windows GDI and X11 accelerated graphics? How about for printing on Linux or BSD?

// Tom

Subject: Re: Painter 2.0
Posted by mirek on Tue, 17 Feb 2009 11:06:27 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 17 February 2009 04:35Hi Mirek,

Painter 2.0 was a welcome enhancement. Thank you!

I recall you were earlier referring to a future CoreDraw and more. Could you outline the future you have planned for Draw and Painter interfaces?

Interfaces stay (with minor improvements / fixes).

The goal of CoreDraw is to separate Draw from sytem. The main target is webserver like apps (which need graphics, in form of e.g. .png, but cannot use X11).

Quote:
As I understand it, currently the Draw interface can be used to render graphics on screen, images and printer using the hardware accelerated graphics of the system where available.

Yes.

Quote:
Now, the Painter interface can be used to render graphics on ImageBuffers without hardware acceleration.

Yes, but I think people are generally puttin too much emphasis on "hardware acceleration". 2D works quite well (and in fact, in some cases even faster) without HW. MacOS X since recently was using only sw rendering of 2D graphics and nobody complained...

Quote:
Is there a plan to create a Painter interface for Windows GDI and X11 accelerated graphics?

No, does not make sense. WinGDI and X11 are not good enough.

In theory, we could consider OpenGL / DirectDraw...

Quote:
  How about for printing on Linux or BSD?
// Tom

AFAIK, works.

Mirek

Subject: Re: Painter 2.0
Posted by cbpporter on Tue, 17 Feb 2009 12:39:13 GMT
View Forum Message <> Reply to Message

And how do you intend to separate the use of Draw (with Win32/X11) from Painter. In some setups (servers) I would like to have completely controlled and self contained software rendering for handling graphics, but I would like the same graphics handled by the native platform API in other more desktop centric solutions.

On the other hand, some years ago I had a self written library for graphics quite similar to Painter (but without subpixel precision) which was used to solve the terrible inconsistencies between different versions of Windows, and also was considerably more powerful. This library was used extensively and while only doing software rendering it had very good performance (I could never replicate the same methods under U++ since single allocation bitmaps are so difficult to handle). So theoretically a Painter based solution could be usable in desktop environment. So maybe a single interface is in order, which handles the common denominator of both Draw and Painter and which can choose it's backend.

Subject: Re: Painter 2.0
Posted by mirek on Tue, 17 Feb 2009 13:42:06 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Tue, 17 February 2009 07:39
So maybe a single interface is in order, which handles the common denominator of both Draw and Painter and which can choose it's backend.

Painter implements Draw. That IMO is as far as you can get.

To make things clear: Draw has to be there because DrawText, DrawImage (single image many times) and DrawRect are HW accelerated. Also, they are enough to create GUI (e.g. CtrlLib only uses these 3 methods). Also worth noting is that DrawText is using system font renderer - rendering fonts any other way would make your app to look inconsistent with the rest of OS.

If you are about to create some complex graphics, use Painter and use Draw to put the resulting image on the screen. (And to paint to Painter, you can use any routine that accepts Draw&, e.g. RichText works in with Painter).

Mirek

Subject: Re: Painter 2.0
Posted by cbpporter on Wed, 18 Feb 2009 12:44:27 GMT
View Forum Message <> Reply to Message

Sorry if I'm missing something, but how can you use Painter when X libs are not available, if Painter inherits from Draw and Draw can't be built without those libs.

Or is this only a solution for HQ drawings, without solving the platform GUI lib dependency problem?

Subject: Re: Painter 2.0
Posted by mirek on Wed, 18 Feb 2009 13:47:34 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Wed, 18 February 2009 07:44Sorry if I'm missing something, but how can you use Painter when X libs are not available, if Painter inherits from Draw and Draw can't be built without those libs.

That is why we need DrawCore.... We need abstract "Draw" with no Xlib dependencies.

The, in "Draw" there will be platform Draw derived class that is then used in GUI widgets.

Mirek

Subject: Re: Painter 2.0
Posted by mirek on Wed, 18 Feb 2009 15:59:07 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 18 February 2009 08:47cbpporter wrote on Wed, 18 February 2009 07:44Sorry if I'm missing something, but how can you use Painter when X libs are not available, if Painter inherits from Draw and Draw can't be built without those libs.

That is why we need DrawCore.... We need abstract "Draw" with no Xlib dependencies.

The, in "Draw" there will be platform Draw derived class that is then used in GUI widgets.

Mirek

Thinking about the issue more, I guess the final solution will to move "GuiDraw" from Draw to CtrlCore, leaving Draw independent from X11 libs.

Of course, we will still have to handle text metrics - but fontconfig and freetype should fill the gap, I hope...

Mirek

Subject: Re: Painter 2.0
Posted by Mindtraveller on Tue, 03 Mar 2009 22:52:57 GMT
View Forum Message <> Reply to Message

Just a few words about Painter after a little using it.

First of all, big thanks to U++ authors. Antialising was jusr the thing I was looking for some applications. This makes visualization really eye-candy in a number of cases. Very useful thing, thanks!

Second, I`ve found very little bug in PainterExample when tested it with BackPaint:
```
//lines 84-85 @ main.cpp:
  for(int y = 0; y + 32 < sz.cy; y += 32)
   for(int x = 0; x + 32 < sz.cx; x += 32)

//should be changed to
  for(int y = 0; y < sz.cy; y += 32)
   for(int x = 0; x < sz.cx; x += 32)
```

Third, one little remark about Subpixel antialiasing mode. This makes lines look better but text looks too blurry, especially middle-sized. It`s bad for eyes IMO. So when one draws any complex things with good (subpixel) quality - it looks like one needs to split drawing in two stages. One for text (simple AA), and second for geometry (subpixel AA). This looks too complex.
These are just my thoughts and I`m not a professional in subpixel AA applied to fonts, so I may be wrong of course. A kind of user`s point of view.

Subject: Re: Painter 2.0
Posted by mirek on Fri, 06 Mar 2009 16:52:01 GMT
View Forum Message <> Reply to Message

Small update:

- I have optimized *generic* text drawing, average speedup about 50%

- I have added "BeginOnPath" command which transforms coordinate system to "lay" on current path. Can be used to e.g. render text on curve or to place arrow markers to lines.

Subject: Re: Painter 2.0
Posted by Mindtraveller on Wed, 18 Mar 2009 22:34:02 GMT
View Forum Message <> Reply to Message

As there`s no special "Painter" forum (hope we`ll have one), I post my little proposal here. It is adopted from one of my recent Painter etudes.

//Rectangle with rounded corners

```
Painter & Painter::Rectangle(double x, double y, double cx, double cy, double r)
{
 ASSERT(r >= 0.);
 if (cx < 0.) {x+=cx; cx=-cx;}
 if (cy < 0.) {y+=cy; cy=-cy;}

 Move(x+r,y)
 .Arc(x+r,y+r,r,r,-M_PI/2.,-M_PI/2.)
 .Line(x, y+cy-r)
 .Arc(x+r,y+cy-r,r,r,M_PI,-M_PI/2.)
 .Line(x+cx-r, y+cy)
 .Arc(x+cx-r,y+cy-r,r,r,M_PI/2,-M_PI/2.)
 .Line(x+cx, y+r)
 .Arc(x+cx-r,y+r,r,r,0.,-M_PI/2.)
 .Line(x+r, y);
}
```

Subject: Re: Painter 2.0
Posted by mirek on Thu, 19 Mar 2009 07:38:04 GMT
View Forum Message <> Reply to Message

Thanks, adopted as RoundedRectangle.

Mirek

Subject: Re: Painter 2.0
Posted by Mindtraveller on Fri, 20 Mar 2009 01:11:00 GMT
View Forum Message <> Reply to Message

I`ve met something like a little bug. I`m not sure if it is a bug at all, but this behaviour seems slightly surprising.
Imagine such a code:

```
//String s="something";
Painter p;
p.Rectangle(....).Stroke(1,Black());
p.Text(s,....).Fill(Blue());
```

It draws unfilled black rectangle with blue text inside of it. The surprise is that you will have black rectangle filled with blue when s is empty string.

Subject: Re: Painter 2.0

Posted by mirek on Fri, 20 Mar 2009 06:16:16 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Thu, 19 March 2009 21:11I`ve met something like a little bug. I`m not sure if it is a bug at all, but this behaviour seems slightly surprising.
Imagine such a code:
//String s="something";
Painter p;
p.Rectangle(....).Stroke(1,Black());
p.Text(s,....).Fill(Blue());
It draws unfilled black rectangle with blue text inside of it. The surprise is that you will have black rectangle filled with blue when s is empty string.

Nice - it is because no new path is defined, so previous path (recangle) is used. I will have to add some "clear path command" I guess...

Mirek