Subject: let's discuss new Draw principles and problems...
Posted by fudadmin on Mon, 13 Feb 2006 12:22:50 GMT
View Forum Message <> Reply to Message

I'd like our cozy community to express their opinions about the features of the new Draw. Maybe that would help to make it really cool? What do think? What do you expect from it? What are the current problems?

Subject: Re: let's discuss new Draw principles and problems...
Posted by jadeite on Mon, 13 Feb 2006 12:35:19 GMT
View Forum Message <> Reply to Message

Firts, for those of us that are new, can you say -

- Is the 'new' Draw already implemented in current stable download, or is it only in uvs2/cvs?

- What is in the new Draw that is different from before?

What I would like:
- Fast, with anti-aliasing

- AGG support. Not necessarily as default, but optional. This is the way VCF does it. One line of code, and all of a sudden you are using AGG rendering for strokes and fills. It really is as simple as that. Nice for anti-aliasing, dashing, etc.

- Alpha blending/transparency in view area.

Subject: Re: let's discuss new Draw principles and problems...
Posted by mirek on Mon, 13 Feb 2006 12:41:27 GMT
View Forum Message <> Reply to Message

Just to kickstart this discussion, I would like to list problems that I hope to solve:

- we need to enforce RGBA format as the dominant Image format. Current draw does not support alpha at any level. Host platforms support alpha images for raster bitmaps (starting with Win98).

- new Image will need new Icon designer.

- Text rendering should be redesigned, separating font information from Draw. Implementation should be improved to synthetise characters missing in Linux fonts.

Now of course comes evergreen question about advanced rendering capabilities. IMHO, this is two-headed problem:

- host platform support part - while Cairo and GDI+ are of course very nice, they fail to provide "runs out of box" guarantee

- meanwhile, dominant number of applications does not need advanced rendering at all

Makes me think there are two possibilities of advanced rendering (in fact, they are not mutually exclusive, so both can be sooner or later implemented):

- client-side advanced rendering, using direct access to the new RGBA surfaces. This has disadvantage of being quite a lot of work to do and not using hardware acceleration (OTOH, software can be pretty fast there as well). Advantage is that it would not require anything from the host platform, it would even allow using Draw on GUI-less machines (good for web servers). In fact, using some existing advanced rendering library here looks like quite a good option too (AGG comes to mind).

- using advanced rendering of host platform. That would involve having separate "DrawEx" package that would provide, for applications that need so, interface for host platform advanced rendering ops (GDI+, Cairo).

Mirek

Subject: Re: let's discuss new Draw principles and problems...
Posted by fudadmin on Mon, 13 Feb 2006 13:30:28 GMT
View Forum Message <> Reply to Message

1. I'd like to have choices (like global settings) to use hardware acceleration or not etc.
2. Also, as many choices for configuration as possible.
3. Maybe intelligent (statistical?) self-reconfiguration.
4. What if to program some parts of it in pure assembler?

Subject: Re: let's discuss new Draw principles and problems...
Posted by fudadmin on Mon, 13 Feb 2006 13:46:20 GMT
View Forum Message <> Reply to Message

BTW, pixel manipulations can be done in memory but IMOH the bottleneck is to pump them from RAM memory to video card memory. Especially, in 32bit format. What tricks do you propose?

Subject: Re: let's discuss new Draw principles and problems...
Posted by mirek on Mon, 13 Feb 2006 15:12:48 GMT
View Forum Message <> Reply to Message

BTW, as tests clearly showed, for very complex scenes (like full screen of letters, each letter has

different color), overhead associated with pumping graphical commands through OS to videocard is quite high too... (even higher than pumping single bitmap)

So it all depends. That is why Draw is quite consrvative in what to expect from hardware/OS and I would like to keep it that way.

Mirek

Subject: Re: let's discuss new Draw principles and problems...
Posted by fudadmin on Mon, 13 Feb 2006 22:49:36 GMT
View Forum Message <> Reply to Message

Is semi-transparent menus considered advanced graphics?

Subject: Re: let's discuss new Draw principles and problems...
Posted by mirek on Mon, 13 Feb 2006 23:04:28 GMT
View Forum Message <> Reply to Message

No. Everything at "whole-window" level is already possible.

In fact, U++ already uses transparency (blending) for menu visual effects.

I have already experimented with semitransparent menus, it is no problem aside form the fact that it is not standard appearance in WinXP.

Mirek

Subject: Re: let's discuss new Draw principles and problems...
Posted by fudadmin on Mon, 13 Feb 2006 23:05:54 GMT
View Forum Message <> Reply to Message

what if to use regions with less bits as possible and an array of changing palettes (kind of compression and styles) and multi-buffering?

Subject: Re: let's discuss new Draw principles and problems...
Posted by fudadmin on Mon, 13 Feb 2006 23:09:48 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 13 February 2006 18:04

... it is not standard appearance in WinXP.

Mirek

But Windows Vista is comming... (BTW, very vector oriented...)

Subject: Re: let's discuss new Draw principles and problems...
Posted by mirek on Mon, 13 Feb 2006 23:17:09 GMT
View Forum Message <> Reply to Message

Then we have to hope we will better have "chameleon" ready by the time Vista comes

Well, you know, in fact this "advanced draw" pressure makes me uneasy. The real problem is that it would be very easy to use GDI+ and allow all kinds of visual eye-candy. However, it would at the same time mean bye bye to Linux/Win2k/Win98 compatibility, very likely also bye byte to printing, to export PDFs etc, etc...

Mirek

Subject: Re: let's discuss new Draw principles and problems...
Posted by fudadmin on Mon, 13 Feb 2006 23:22:07 GMT
View Forum Message <> Reply to Message

what if to make our own "GDI+" for all OS?

Subject: Re: let's discuss new Draw principles and problems...
Posted by mirek on Mon, 13 Feb 2006 23:25:25 GMT
View Forum Message <> Reply to Message

How?

The only possibility is client-side draw (manipulating pixels in memory). Nice idea, but for some reason, some computers are too poor when transporting bitmaps from memory to VGA (as was shown by our tests - while other are perfectly OK to do so)

OTOH, as I have explained in the beginning, this is still one of options - but to do that, we need RGBA-dominant image. Working on it....

Mirek

Subject: Re: let's discuss new Draw principles and problems...
Posted by fudadmin on Mon, 13 Feb 2006 23:30:16 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Mon, 13 February 2006 18:22what if to make our own "GDI+" for all OS? BTW, at the moment I'm experimenting with assembler... Also, maybe we can find some ready-to-use functions on the net?

Subject: Re: let's discuss new Draw principles and problems...
Posted by fudadmin on Mon, 13 Feb 2006 23:34:15 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 13 February 2006 18:25How?

The only possibility is client-side draw (manipulating pixels in memory). Nice idea, but for some reason, some computers are too poor when transporting bitmaps from memory to VGA (as was shown by our tests - while other are perfectly OK to do so)

OTOH, as I have explained in the beginning, this is still one of options - but to do that, we need RGBA-dominant image. Working on it....

Mirek

What list of problems are exactly?

Subject: Re: let's discuss new Draw principles and problems...
Posted by mirek on Tue, 14 Feb 2006 00:09:17 GMT
View Forum Message <> Reply to Message

Well, assembler is not a problem, but not a huge win either. You know, in initial experiments, the problem was not to render the image by software, but to put it to screen (pump it though the OS). Assembler is not gone help there.

Just interesting sidenote: Hand-made assembly did not gave us any significant improvements over C++ code, 10% was maximum I was able to achieve. Seems like modern CPUs and compilers are quiete optimal when dealing with C++ code...

Mirek

Subject: Re: let's discuss new Draw principles and problems...
Posted by fudadmin on Tue, 14 Feb 2006 08:26:35 GMT

View Forum Message <> Reply to Message

What if to use TV principle: use even and odd scanlines?

Subject: Re: let's discuss new Draw principles and problems...
Posted by mirek on Tue, 14 Feb 2006 08:35:02 GMT
View Forum Message <> Reply to Message

To reduce required bandwith?

Well, interesting idea, but I doubt visual artifacts would be acceptable.

Frankly, what is bad about this:

- use basic Draw for most parts that do not require advanced rendering (e.g. editor in TheIDE)

- use client-side, double-buffered extended Draw (perhaps AGG based) for intensive graphical tasks

?

Mirek

Subject: Re: let's discuss new Draw principles and problems...
Posted by fudadmin on Tue, 14 Feb 2006 09:59:40 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 14 February 2006 03:35To reduce required bandwith?

to fool the eye and speed up.

Quote:
Well, interesting idea, but I doubt visual artifacts would be acceptable.

what exactly? I think opposite! It would give nice and smooth rendering. Solar OS use it as main one. And I guess even more applications... I suspect even Opera's menus ...

Quote:
Frankly, what is bad about this:

- use basic Draw for most parts that do not require advanced rendering (e.g. editor in TheIDE)

Why then do we need the new draw at all? Or let's make separate modules? Maybe we need a structured list with all the requirements?

Quote:
- use client-side, double-buffered extended Draw (perhaps AGG based) for intensive graphical tasks

I expect multi-buffered with "dirty" rectangles and/or regions...

Subject: Re: let's discuss new Draw principles and problems...
Posted by mirek on Tue, 14 Feb 2006 10:17:04 GMT
View Forum Message <> Reply to Message

Interlaced:

E.g. imagine holding page-down (with 30 fps autorepeat) in wordprocessor and interlaced rendering - you would see two pages at once.

Why new Draw:

New Draw infrastructure is needed to allow client-side advanced drawing. We just need to take this first step to get any further. How advanced draw will look like does not depend on this first step. Second reason is that current Draw raster image model does not support RGBA images and that is a serious flaw.

Mirek

Subject: Re: let's discuss new Draw principles and problems...
Posted by fudadmin on Tue, 14 Feb 2006 10:54:37 GMT
View Forum Message <> Reply to Message

Let's talk RGBA problem.
What do we need for RGBA exactly? Can we split this RGBA problem into 2 parts:
1 Part. Load, save, convert - format problems and manipulations in memory. (We need functions for them). What else?

2 Part. Display - "pumping" problems? But, AFAIK, 4 bytes is even better than 3 bytes for pumping. And let's say we can forget about hardware acceleration because as your tests show and materials on the net - there's no real advantage of it for 2D.What else?

Subject: Re: let's discuss new Draw principles and problems...
Posted by mirek on Tue, 14 Feb 2006 11:12:29 GMT
View Forum Message <> Reply to Message

- We need RGBA because we need RGBA images. RGBA is simply the most generic format (well, theoretically we could also think about 8 bytes RGBA - 2 bytes per channel).

- From pumping perspective, on modern GPU (by modern I mean anything past 1998) the source format does not matter, as conversions are performed by GPU. However, it makes _huge_ difference for sofware rendering operations. What is even more important, you have to implement rendering just for single format.

- It is true that for rendering itself, HW acceleration does not matter. However, pumping to video memory can be bottleneck for many machines.

Actually, the original idea (valid for last year's plans) was exactly that - do everything in sofware. That was before you have run the tests on your machine, showing amazing 10fps pumping performance (worst case since that was 40fps on unodgs Duron and even that was barely acceptable, many machines are capable of 200+ fps on that test).

Another possible disadvantage of software rendering is that you cannot use it for printer (it would be too slow).

Yet another disadvantage is that software rendering places high(er) load on CPU (means higher overal power consumption - bad for laptops).

I believe that what we are working on now will provide reasonable compromise (default normal rendering, advanced software rendering where needed).

Mirek

Subject: Re: let's discuss new Draw principles and problems...
Posted by fudadmin on Thu, 16 Feb 2006 07:22:16 GMT
View Forum Message <> Reply to Message

What do you think of CxImage library? Could we use it all or some parts of it? It has free zlib licence.
http://www.codeproject.com/bitmap/cximage.asp

Subject: Re: let's discuss new Draw principles and problems...
Posted by jadeite on Thu, 16 Feb 2006 12:43:11 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Thu, 16 February 2006 02:22What do you think of CxImage library?
I don't know anything about this lib except from what I read in the 1st paragragraph of the codeproject article. Although the 'core' of the lib might be platform independent, the author clearly claims that the lib as a whole is not. The question is, can the platform dependent part be factored out easily by the devs here? If not, is this really the road you plan to take, locking yourself into Windows?

Subject: Re: let's discuss new Draw principles and problems...
Posted by fudadmin on Thu, 16 Feb 2006 14:01:21 GMT
View Forum Message <> Reply to Message

I've downloaded the sources of CxImage and AGG. AGG is definetely more useful. And it's more elegant in terms of programming style. IMHO, Myrek should find a niche for AGG in U++...

Subject: Re: let's discuss new Draw principles and problems...
Posted by mirek on Thu, 16 Feb 2006 16:15:49 GMT
View Forum Message <> Reply to Message

I already did.

Whith new ARGB Image with direct surface access, you will be able to plugin similar libraries without too much hassle.

AGG is definitely the most interesting candidate.

Mirek

Subject: Xara Xtreme Engine
Posted by CyberEye on Wed, 07 Jun 2006 08:59:54 GMT
View Forum Message <> Reply to Message

Hello @ all!

This is my first posting here and I must say: Ultimate++ and TheIDE are very good! The concept is awesome! One little critic: The Ultimate++ Framework is this time less documented. But I know, that this task is work in progress.

But now to the topic:
I have read, that there is a problem to find a good platformindependent Draw-Engine.

What do you think about "Xara Xtreme"?
http://www.xaraxtreme.org

It is Open-Source for Linux. But I think, you can adept the engine and use it for platformindependent implementation.

Or do you search an Engine, to draw the GUI itself? Such as GTK, GDI, ...?

friendly greetings

CyberEye

PS.: I'm from Germany, so do not critic my English much, please.

Subject: Re: Xara Xtreme Engine
Posted by fudadmin on Wed, 07 Jun 2006 10:56:14 GMT
View Forum Message <> Reply to Message

CyberEye wrote on Wed, 07 June 2006 09:59Hello @ all!

This is my first posting here and I must say: Ultimate++ and TheIDE are very good! The concept is awesome! One little critic: The Ultimate++ Framework is this time less documented. But I know, that this task is work in progress.

But now to the topic:
I have read, that there is a problem to find a good platformindependent Draw-Engine.

What do you think about "Xara Xtreme"?
http://www.xaraxtreme.org

It is Open-Source for Linux. But I think, you can adept the engine and use it for platformindependent implementation.

Or do you search an Engine, to draw the GUI itself? Such as GTK, GDI, ...?

friendly greetings

CyberEye

PS.: I'm from Germany, so do not critic my English much, please.

GNU licence means Generally Not Usable...

Subject: Re: Xara Xtreme Engine
Posted by mirek on Wed, 07 Jun 2006 11:30:52 GMT
View Forum Message <> Reply to Message

CyberEye wrote on Wed, 07 June 2006 04:59Hello @ all!

This is my first posting here and I must say: Ultimate++ and TheIDE are very good! The concept is awesome! One little critic: The Ultimate++ Framework is this time less documented. But I know, that this task is work in progress.

But now to the topic:
I have read, that there is a problem to find a good platformindependent Draw-Engine.

Actually, not. We have agreed to

- provide means for integration of any software renderer

- start with AGG

Mirek
