Subject: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by kohait00 on Fri, 02 Jul 2010 08:19:05 GMT
View Forum Message <> Reply to Message

hi folks

i just took a brief look on sdl again, and in some other posts here the idea of porting upp to Mac world has stucked a bit. to push it a bit, here is an idea..

why not porting to sdl first, which would support MacOSX and many other platforms at once as well, and then, little by little...to platforms natively (even framebuffer, as stated in some of my other posts). SDL programs can be compiled in Upp already, but standalone ofcorse..

here is some quote again from libsdl.org

Quote:
SDL supports Linux, Windows, Windows CE, BeOS, MacOS, Mac OS X, FreeBSD, NetBSD, OpenBSD, BSD/OS, Solaris, IRIX, and QNX. The code contains support for AmigaOS, Dreamcast, Atari, AIX, OSF/Tru64, RISC OS, SymbianOS, and OS/2, but these are not officially supported.

SDL is written in C, but works with C++ natively, and has bindings to several other languages, including Ada, C#, D, Eiffel, Erlang, Euphoria, Guile, Haskell, Java, Lisp, Lua, ML, Objective C, Pascal, Perl, PHP, Pike, Pliant, Python, Ruby, Smalltalk, and Tcl.

SDL is distributed under GNU LGPL version 2.

RFC

PS: we at leas could take a look ad SDL in terms of how they bind with Objective C.. but i dont know exactly to *what* SDL bind, to the older Carbon API or the newer Cocoa API from MacOSX.

PSS: it would ne no great deal i think, becasue SDL offers just about same terms of interface, messageque and a blit surface, to which a SystemDraw could paint..

--> like always, RTFFAQ helps

http://www.libsdl.org/faq.php?action=listentries&categor y=7

so seams like SDL supports the Cocoa API already. best for us

will there be a licence issue if we get "inspiration only" from LGPL code related to MacOSX in SDL (LGPL). is that considered derived work?

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)

Posted by Mindtraveller on Fri, 02 Jul 2010 14:24:40 GMT
View Forum Message <> Reply to Message

I don't think it can be consdidered derived work as long as you don't take original LGPL-ed code as a basement for your code (it means copy-paste and then edit).

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by kohait00 on Fri, 02 Jul 2010 14:53:04 GMT
View Forum Message <> Reply to Message

no, its basicly to know what kind of functions are to be taken care of, so not to have to start from zero digging in man pages etc.

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by kohait00 on Thu, 18 Nov 2010 08:15:05 GMT
View Forum Message <> Reply to Message

i've seen koldo porting SDL to be a SDLCtrl.
but it seems not to be a 'true' sdl port, heavy relying on windows draw surface handles etc..

what was the intention for that?

btw:
Functions4U_Gui.h:7
needs to read like this to compile it successfully for GCC:
(making Image ref a const one)

inline const RGBA *GetPixel(const Image &img, int x, int y) {

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by koldo on Thu, 18 Nov 2010 10:13:15 GMT
View Forum Message <> Reply to Message

Hello kohait0
Quote:i've seen koldo porting SDL to be a SDLCtrl.
but it seems not to be a 'true' sdl port, heavy relying on windows draw surface handles etc..
what was the intention for that?

SDLCtrl is a control to handle SDL. It is designed for Linux and for Windows. It is going to evolve in few time but published code will remain rather stable.

It includes also a SDLCtrl_demo, based in unodgs SDL demo, but inside a Ctrl, so it is in a window, it moves and resizes and the creation and destruction is internally handled so the

programmer just have to program the video and audio. SDLCtrl supplies the remaining services.

The first intention is to be a base for a MediaCtrl to come soon. However SDLCtrl will include a C++ interface to many SDL possibilities.
Quote:Functions4U_Gui.h:7
needs to read like this to compile it successfully for GCC:
(making Image ref a const one)Sorry, I do not understand. Actually it compiles and runs without problem. However I will add the const to Image, it is more adequate.

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by kohait00 on Thu, 18 Nov 2010 10:19:06 GMT
View Forum Message <> Reply to Message

ah ok, now i get it..

SDLCtrl is meant to be just as any other Ctrl, beeing able to be included in a usual win/linux application.

so what still remains is to have a native rendering backend for Upp on solely SDL, no matter win or linux, to render SystemDraw on SDLSurface.

i think your example will be a great deal of help in doing just that.

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by koldo on Thu, 18 Nov 2010 10:23:42 GMT
View Forum Message <> Reply to Message

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by mirek on Sat, 27 Nov 2010 17:21:37 GMT
View Forum Message <> Reply to Message

I have came to conclusion that what we need to do first is total separation of host platforms from the code, so that anybody can start developing ports without the need to edit CtrlCore.

So I decided to start soon "project Rainbow", that will do just this. I hope to start in Jan and be done within 2-3 months.

BTW, my motivations is even more ambitious - this will allow me to experiment with "ultra-thin" web apps, where all processing is done by U++ and displayed by Java on the client (alt. Flash or maybe even Javascript) in a way similar to Terminal Services. So basically developing app with web interface should be principally the same as developing normal 'fat' U++ app...

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by dolik.rce on Sat, 27 Nov 2010 18:00:09 GMT
View Forum Message <> Reply to Message

luzr wrote on Sat, 27 November 2010 18:21BTW, my motivations is even more ambitious - this will allow me to experiment with "ultra-thin" web apps, where all processing is done by U++ and displayed by Java on the client (alt. Flash or maybe even Javascript) in a way similar to Terminal Services. So basically developing app with web interface should be principally the same as developing normal 'fat' U++ app...
Do you mean something like this: http://blogs.gnome.org/alexl/2010/11/23/gtk3-vs-html5/? (Just better, as it will be written in U++)

Anyway, such separation would be really great... It might also help a lot in (probably near) future when wayland server starts replacing Xorg.

Honza

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by mirek on Sat, 27 Nov 2010 18:51:36 GMT
View Forum Message <> Reply to Message

dolik.rce wrote on Sat, 27 November 2010 13:00luzr wrote on Sat, 27 November 2010 18:21BTW, my motivations is even more ambitious - this will allow me to experiment with "ultra-thin" web apps, where all processing is done by U++ and displayed by Java on the client (alt. Flash or maybe even Javascript) in a way similar to Terminal Services. So basically developing app with web interface should be principally the same as developing normal 'fat' U++ app...
Do you mean something like this: http://blogs.gnome.org/alexl/2010/11/23/gtk3-vs-html5/? (Just better, as it will be written in U++)

Yes. The question is at what level to do rendering. Above example simply passes changes in raster graphics. I believe that transfering "Draw stream" is more viable solution. I have done some preliminary experiments and have found that my current bussines application "full rePaint" produces Draw data that can be compressed to something like 5-8KB, maybe even much less (well, Image data are not in this, but after all, Images will be transfered just once...)

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by koldo on Sun, 28 Nov 2010 08:51:09 GMT
View Forum Message <> Reply to Message

Hello Mirek

So Rainbow package would be like Painter, but to be used in a web server to send Javascript to the client?. Would it include controls too?

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by mirek on Sun, 28 Nov 2010 11:21:18 GMT
View Forum Message <> Reply to Message

koldo wrote on Sun, 28 November 2010 03:51Hello Mirek

So Rainbow package would be like Painter, but to be used in a web server to send Javascript to the client?. Would it include controls too?

No.

Actually, anything web related is "to be decided later".

What I want to do as "Project Rainbow" is CtrlCore host platform separation. That is a good prerequisite for framebuffer, SDL, MacOS X, Android and maybe "web terminal".

The important thing is that after the separation, all these technologies could be developed without edits to CtrlCore...

In other words, it will be possible to work on all these projects simultanously.

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by unodgs on Sun, 28 Nov 2010 11:27:24 GMT
View Forum Message <> Reply to Message

luzr wrote on Sat, 27 November 2010 12:21I have came to conclusion that what we need to do first is total separation of host platforms from the code, so that anybody can start developing ports without the need to edit CtrlCore.

So I decided to start soon "project Rainbow", that will do just this. I hope to start in Jan and be done within 2-3 months.

BTW, my motivations is even more ambitious - this will allow me to experiment with "ultra-thin" web apps, where all processing is done by U++ and displayed by Java on the client (alt. Flash or maybe even Javascript) in a way similar to Terminal Services. So basically developing app with web interface should be principally the same as developing normal 'fat' U++ app...

That's a really really great news. Can't wait to try it in action. That would save me a lot of time. I could simply avoid creating web version of my app. Gtk guys use canvas and javascript - seems to be a good choice.

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by mirek on Sun, 28 Nov 2010 11:51:15 GMT
View Forum Message <> Reply to Message

unodgs wrote on Sun, 28 November 2010 06:27
That's a really really great news. Can't wait to try it in action. That would save me a lot of time. I could simply avoid creating web version of my app. Gtk guys use canvas and javascript - seems to be a good choice.

Unfortunately, javascript appears to be to weak for RDP-like stream...

We would need at least Java to get what we want.

Plus, perhaps this technology will help in similar cases where Terminal services is usable, but would have big problem when your web latency is too high (just like with TS).

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by Tom1 on Mon, 29 Nov 2010 09:11:08 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Should I understand your 'project Rainbow' as 'GUI abstraction layer', a package that exposes the GUI API of each supported platform in a unified manner so that the future CtrlCore always sees this as the same regardless of the platform? (This includes the likes of SystemDraw, message queue handling, window management, etc.. or what?)

The GUI porting efforts to different platforms would then essentially focus on this GUI abstraction layer, right?

Best regards,

Tom

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by mr_ped on Mon, 29 Nov 2010 09:32:28 GMT
View Forum Message <> Reply to Message

yes, I think (actually I'm sure) this is what he meant.

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by mirek on Mon, 29 Nov 2010 09:50:01 GMT
View Forum Message <> Reply to Message

Yes. Basically, the goal is to to wipe out most of these #ifdef PLATFOM_ lines from CtrlCore. Except that I guess it still makes sense to have X11/Win32 implementations there...

Also, doing so, I think it even makes sense to make the system flexible so that more GUI modes can be available in single binary. I can imagine e.g. webserver that provides some diagnostics in Win32/X11 GUI...

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by Tom1 on Mon, 29 Nov 2010 11:56:14 GMT
View Forum Message <> Reply to Message

OK, I see.

Would it be possible to expose the platform graphics capabilities via Painter interface (possibly in addition to the conventional Draw interface) in this new 'GUI abstraction layer'? I mean in such a way that the Painter drawing primitives would be implemented using hardware acceleration if such is supported by the platform.

Best regards,

Tom

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by mirek on Mon, 29 Nov 2010 18:08:50 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 29 November 2010 06:56OK, I see.

Would it be possible to expose the platform graphics capabilities via Painter interface (possibly in addition to the conventional Draw interface) in this new 'GUI abstraction layer'?

No need. Painter interface already exists, you can already implement it in HW (other thing is that there is very likely no meaningful way of doing so, at least not with usual OpenGL - like graphics acceleration).

Mirek

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by zsolt on Fri, 17 Dec 2010 11:16:17 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 28 November 2010 12:51unodgs wrote on Sun, 28 November 2010 06:27
That's a really really great news. Can't wait to try it in action. That would save me a lot of time. I could simply avoid creating web version of my app. Gtk guys use canvas and javascript - seems

to be a good choice.

Unfortunately, javascript appears to be to weak for RDP-like stream...

We would need at least Java to get what we want.

Plus, perhaps this technology will help in similar cases where Terminal services is usable, but would have big problem when your web latency is too high (just like with TS).

What about HTML5 Canvas and the full duplex WebSocket API in HTML5 Javascript?

See ThinVNC as an example.

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by mirek on Sat, 18 Dec 2010 08:37:07 GMT
View Forum Message <> Reply to Message

zsolt wrote on Fri, 17 December 2010 06:16

What about HTML5 Canvas and the full duplex WebSocket API in HTML5 Javascript?

See ThinVNC as an example.

Yes, generally that is exactly I was planning for.

However I am a bit scared that Canvas is not supported in IE8. OTOH, it will take time to develop this, so when it is ready, there will be Canvas in most browsers anyway...

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by kohait00 on Tue, 05 Jul 2011 14:28:57 GMT
View Forum Message <> Reply to Message

SDLFb port (very alpha) is available through rainbow layer, thanks mirek.

what about MacOSX native port?

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by chickenk on Tue, 05 Jul 2011 16:14:13 GMT
View Forum Message <> Reply to Message

Hi,
kohait00 wrote on Tue, 05 July 2011 16:28SDLFb port (very alpha) is available through rainbow layer, thanks mirek.

I tried to compile rainbow packages with the SDLFb port using my upp-waf build system, and it works fine expect it showed me that SDLFb package should depend on CtrlCore (it includes CtrlCore.h somewhere). Can you add this dependency?

Congrats for the hard work to everyone involved! Those are really great news.

Lionel

Subject: Re: Porting Upp to SDL first ? (cause of MacOSX & framebuffer)
Posted by kohait00 on Tue, 05 Jul 2011 17:20:54 GMT
View Forum Message <> Reply to Message

There is quite a bit to still figure out how to do, also in terms of backend handling and dependencies. I will try to fix it soon. Next step will be to finish the native framebuffer port

---