Subject: Rainbow, first iteration
Posted by mirek on Mon, 13 Jun 2011 13:03:25 GMT
View Forum Message <> Reply to Message

I took time, especially considering it is mostly preprocessor compile time hack, but I believe that I have reached important milestone in Rainbow development, I believe it should be now possible to develop custom GUI for U++ without the need dir directly changing CtrlCore/CtrlLib (like MacOS or Android or OpenGL).

How is it supposed to work, at least, starting point:

The CtrlCore.h now begins with:

```
#include <guiplatform.h>

#ifndef GUIPLATFORM_INCLUDE

#ifdef PLATFORM_WIN32
#define GUIPLATFORM_INCLUDE "Win32Gui.h"
#endif

#ifdef PLATFORM_X11
#define GUIPLATFORM_INCLUDE "X11Gui.h"
#endif

#endif
```

guiplatform.h is in uppsrc root (not in package) and it is empty, so for normal operations "defaults" kick in.

If you are about to develop custom GUI backend, create a new nest, place guiplatform.h into it and using #define GUIPLATFORM_INCLUDE 'redirect' it to your own GUI specification header.

As for development itself, I believe that two 'default' backends give a good hint... I am using quite ugly combination of macros, includes and platform defined functions/methods; for now it seems to be the most cost efficient way. We will say how that works in practice...

Important notice: Whereas in the past we were using PLATFORM_WIN32 and PLATFORM_X11 in CtrlLib and other GUI code for #ifdefs to tell apart the GUI backend, this should be now replaced by 'GUI_WIN' and 'GUI_X11' (those are defined in GUIPLATFORM_INCLUDE), because it is no longer true that Windows backend has to used on Windows platform...

OK, this should work as first introduction. I am now plannig for actually developing generic framebuffer backend to tune this thing.

Perhaps Daniel could now also try to 'port' OpenGL U++ to rainbow too.

Mirek

Subject: Re: Rainbow, first iteration
Posted by mirek on Mon, 13 Jun 2011 19:24:19 GMT
View Forum Message <> Reply to Message

I have created 'rainbow' nest in svn for general rainbow development and as a proof of concept I have 'reimplemented' Win32 backend (simply by copying existing Win32 backend).

BTW, interesting stat: Win32 backend is about 6000 lines, chameleon excluded...

Subject: Re: Rainbow, first iteration
Posted by chickenk on Tue, 14 Jun 2011 08:36:05 GMT
View Forum Message <> Reply to Message

/home/lionel/upp/uppsrc/CtrlCore/CtrlCore.h:10:2: erreur: #error

I think the '#error' line is a garbage stuff, isn't it?

Subject: Re: Rainbow, first iteration
Posted by unodgs on Tue, 14 Jun 2011 12:06:52 GMT
View Forum Message <> Reply to Message

Excellent news.
Quote:
Perhaps Daniel could now also try to 'port' OpenGL U++ to rainbow too.

Yes, I'm gonna do that.

Subject: Re: Rainbow, first iteration
Posted by mirek on Tue, 14 Jun 2011 15:59:31 GMT
View Forum Message <> Reply to Message

chickenk wrote on Tue, 14 June 2011 04:36/home/lionel/upp/uppsrc/CtrlCore/CtrlCore.h:10:2: erreur: #error

I think the '#error' line is a garbage stuff, isn't it?

Yes, sorry about that...

Mirek

Subject: Re: Rainbow, first iteration
Posted by mirek on Wed, 15 Jun 2011 07:33:12 GMT
View Forum Message <> Reply to Message

There is now rainbow/Skeleton package - empty GUI backend, it only compiles and links and does nothing.

Subject: Re: Rainbow, first iteration
Posted by harmac on Wed, 15 Jun 2011 14:21:35 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 13 June 2011 15:03I am now plannig for actually developing generic framebuffer backend to tune this thing.

Lacking knowledge about U++ implementation and style and graphics programming in general so far, I don't know if the following resources are helpful in the endeavour, as at present I'm too much of a C/C++ newbie, but still:

There is portable framebuffer library pxCore that seems to have been ported already to Windows, PocketPC (ARM4), Linux(X11), OSX. Maybe you can recycle some code of it for U++ if it fits the need or maybe it suits as a generic backend already.

It may not be directly framebuffer related, but when I read about the Fog-Framework, its Fog-UI component seems to use an abstract interface for GUI descriptions that is later mapped to different native targets. It sounds somewhat like what Rainbow seems to be trying. I haven't looked at the implementation but conceptually it sounds cleaner than the current U++ approach, which you name ugly yourself. Coming from different languages, from what I understand about the C/C++ preprocessor, it is an ugly kludge in the first place, and maybe it is worth to consider not using it at all.

If not for Rainbow, looking at Fog might be worthy in its own right, as some of its explicit design notes (like for instance not using STL) seem to be in accord with U++ philosophy, so that it might possibly be interesting to recycle code from there or maybe even join some efforts, as some purposes of both projects seem to overlap.

I don't know in what state Fog currently is, though, as the author seems to be concerned with another project (AsmJit), part of which he but seems to plan to integrate into Fog in the form of BlitJit, and in a post on a blog about the Fog-Framework (which also has a post with a number of vector geometry resources, which are probably a bit over my mind at the moment but which more involved people or those interested in learning might find interesting) he assures that the project is not dead. One of the goals at least seems to be high performance with maximum compatibility and adaptation to the target.

Speaking of which, when somebody recently posted about jslinux, I also read in an article about Fabrice Bellard that he had implemented with TinyGL a small subset of OpenGL that can be used as a fallback for systems that normally don't have hardwaresupport for it. As it is very old and probably incomplete, I don't know how close it is to OpenGL ES, but maybe it might be interesting for implementers of U++ OpenGL backends to take a look at such a limited subset whose GL instruction implementation for some arbitrary target platform can be done with reasonable effort and use that subset as a portable target for translations from U++ GUI descriptions.

Finally, there is Agar, which I hadn't heard about before. It may not follow any U++ philosophy but also has different raw targets and might therefore be interesting for somebody interested in the topic to see how different folks are doing their implementations.

pxCore, Fog-Framework and Agar are BSD-licensed, TinyGL zlib-like. As also some external links on their respective websites might be interesting for one or another and some of the projects themselves might be interesting for the U++ folks, I thought that pointing to these projects might be worthwhile. Note that I currently lack the knowledge to decide whether or not they actually are useful.

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Wed, 22 Jun 2011 08:21:01 GMT
View Forum Message <> Reply to Message

this is really nice news..

MacOSX wont take long either with this helper. thanks.

as of android, it might be quite difficult to circumvent the whole java stuff. the only possibility will be the NDK.

concerning the 'uglyness' of rainbow:
as a program usually only uses one single graphics/ui backend, the compiled-in option is definitely the fastest way. of corse a clean HAL like GUI abstraction layer would be more clean but this al comes at expenses: complexity and speed. i think the current approach, though a bit 'ugly' and not transparent at first glance, is a moonolite solution. id definitely needs kind of a tutorial on how to start and what is what and what is expected to happen inside some functions. but this managable.

pxCore seems appealing. especially in terms of designing apps on PC framebuffer emulation whilst the still run on bare embedded frambuffer. that'd be a niiiiice advantage in embededded world. believe me one often breaks the neck there. recompiling untouched code is just heaven.

anyways, for the clean fb0 solution, i have found somewhere a test prog once, and tried to use it in u++.

find it attached.

@mirek: the Framebuffer / WinFB is a starter isn't it?

Framebuffer.h:60 lacks a ';'
do you already use the WinFB stuff? when i try to compile rainbow/Paint i get some errors.. have any working environment?

if you could provide a slight description on the files/some crucial functions i could try to make some steps.

File Attachments

1) fb0test.rar, downloaded 584 times

Subject: Re: Rainbow, first iteration
Posted by mirek on Wed, 22 Jun 2011 18:40:04 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Wed, 22 June 2011 04:21this is really nice news..

MacOSX wont take long either with this helper. thanks.

as of android, it might be quite difficult to circumvent the whole java stuff. the only possibility will be the NDK.

AFAIK, it is not longer needed. NDK is good enough now (I believe was ugraded at the end of last year).

Quote:
concerning the 'uglyness' of rainbow:

Well, as I am now working on FB backend, it is perhaps ugly by concept, but in "live" development it does not look so bad...

Quote:
and of a tutorial on how to start and what is what and what is expected to happen inside some functions. but this managable.

Actually, that is one purpose of FB backend...

Quote:
@mirek: the Framebuffer / WinFB is a starter isn't it?
Framebuffer.h:60 lacks a ';'
do you already use the WinFB stuff? when i try to compile rainbow/Paint i get some errors.. have any working environment?

It's under development. Frankly, it barely compiles now. A lot has to be done before it even paints something

Mirek

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Wed, 22 Jun 2011 22:02:23 GMT
View Forum Message <> Reply to Message

looking forward to have it
really havent found the time to actually fully get this stuff done, didnt get past the point of extracting/extending the interface just as you did for rainbow now, in a much cleaner way... but i'm still interested.

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Mon, 27 Jun 2011 07:52:40 GMT
View Forum Message <> Reply to Message

really nice progress in Frambuffer.

i tried to compile it in suse, making WinFB dependant of WIN32 flag. it depends on a the windows virtual key codes and the stuff from stdids.h.. the virtual key codes are in WinUser.h, how should we tread this stuff? can we pack them together?

Subject: fb0 shows first Upp stuff
Posted by kohait00 on Mon, 27 Jun 2011 12:49:43 GMT
View Forum Message <> Reply to Message

hi mirek

i played around with rainbow a bit.

thanks to your work, i've been able to have fb0 show the fullscreen view of Paint package.
attached are the 2 patches.

patch0 has some fixes concerning PLATFORM_X11, which should be GUI_X11 or PLATFORM_POSIX now, please review it. there also arises a slight problem with Image, it still has some code PLATFORM_WIN32 and PLATFORM_X11 dependant. but due to elimination of PLATFORM_X11, which is available from Core level, GUI_X11 is available from CtrlCore level only, Image, which is included by CtrlCore does not know anything from GUI_X11. i dont have any solution for this. maybe the platform dependant stuff should be made a special class, which is

implemented platform dependantly.

patch1 has got the changes for the real framebuffer, it is veery basic only, just to make it show sth. what idea did you have concerning ProcessEvent / Eventloop? read /dev/input in another thread? and what about non MT environment? there is no way to make the ImageBuffer directly be based on framebuffer pointer? one could save the copying..

i also attached the sources directly. in case you arent able to aply the diff patch (created under suse 11.3 with git make patch)

i am getting really excited. upp becomes really an option for embedded stuff ..and a good option.

EDIT: for those of you trying it out: make sure to execute the app in a console, where fb is enabled. do not try to execute it in your GNOME or the like environment. it wont show the fb.

File Attachments

1) stuff.tar.gz, downloaded 575 times

Subject: Re: fb0 shows first Upp stuff
Posted by Sgifan on Mon, 27 Jun 2011 14:54:58 GMT
View Forum Message <> Reply to Message

Just for the pleasure of the eyes, could you please send a screen shot of this.

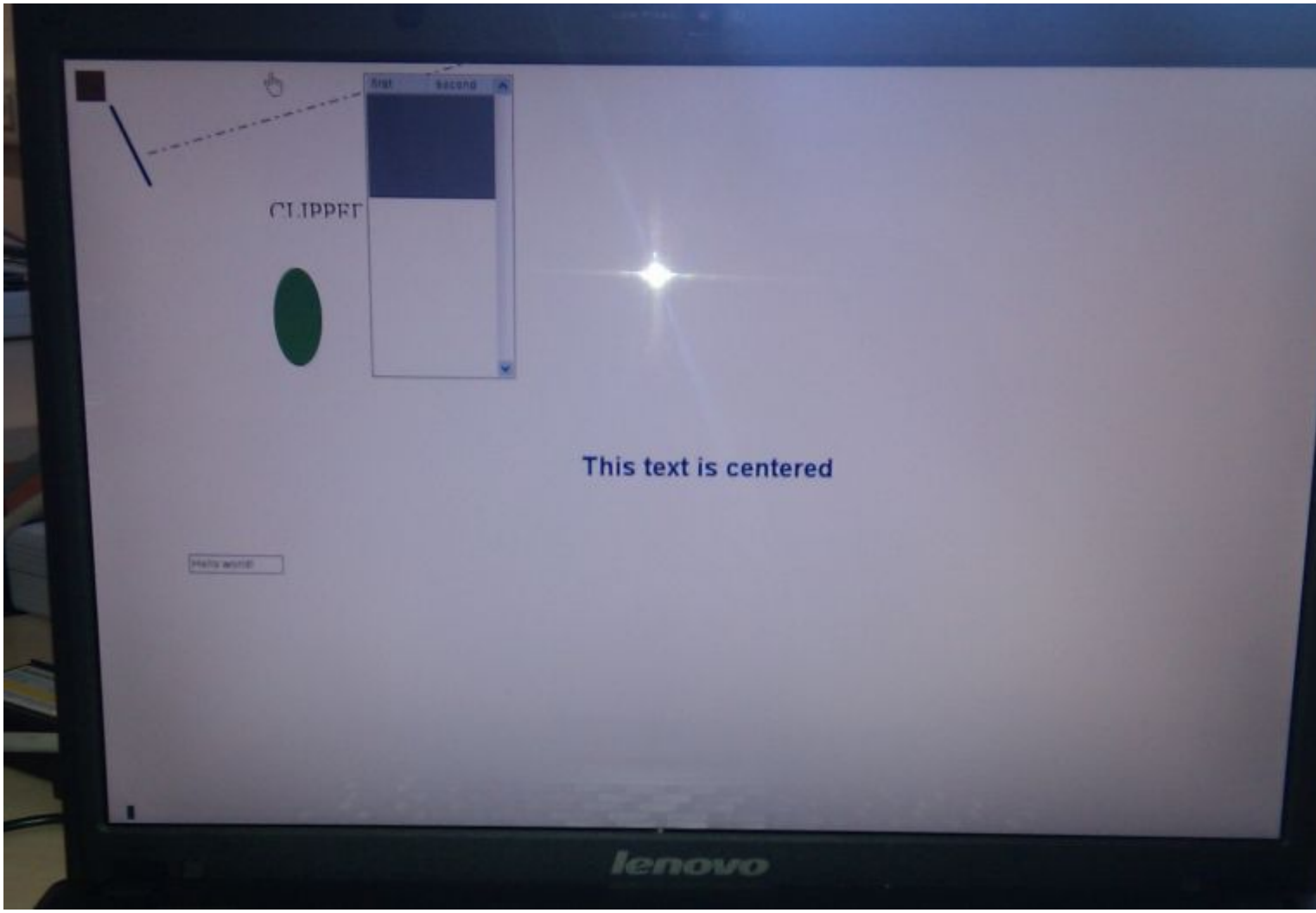
thxs

Subject: Re: fb0 shows first Upp stuff
Posted by kohait00 on Mon, 27 Jun 2011 15:14:57 GMT
View Forum Message <> Reply to Message

this is the quickest option i could find my phone
as mentioned, input handling is not implemented..so the cursor down left is there..
it's from my opensuse enterprise laptop, started from one of the virtual terminals..

File Attachments

1) IMG_20110627_171033.jpg, downloaded 2365 times



Subject: Re: Rainbow, first iteration
Posted by mirek on Mon, 27 Jun 2011 15:43:00 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Mon, 27 June 2011 03:52really nice progress in Frambuffer.

i tried to compile it in suse, making WinFB dependant of WIN32 flag. it depends on a the windows virtual key codes and the stuff from stdids.h.. the virtual key codes are in WinUser.h, how should we tread this stuff? can we pack them together?

Well, that is one of ToDo now... My current plan is to make keycodes defined in 'final' framebuffer backend, perhaps through rainbow #define.

Alternative approach would be to define framebuffer specific keycodes and translate (from Win or X11) to them. IMO, much more work...

Subject: Re: fb0 shows first Upp stuff
Posted by mirek on Mon, 27 Jun 2011 15:50:45 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Mon, 27 June 2011 08:49hi mirek

i played around with rainbow a bit.

thanks to your work, i've been able to have fb0 show the fullscreen view of Paint package.
attached are the 2 patches.

rainbow now has the same rights as bazaar - you are welcome to join the development and apply patches. Please commit linux fb as new backend if you wish so.

Just keep in mind please that the "Framebuffer" is meant to be generic, then there should be some kind of "final" backend, like WinFB.

Mirek

Subject: Re: Rainbow, first iteration
Posted by mirek on Mon, 27 Jun 2011 15:52:46 GMT
View Forum Message <> Reply to Message

P.S.: Give it an hour to rights become active...

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Mon, 27 Jun 2011 16:14:41 GMT
View Forum Message <> Reply to Message

i'll be checking it.. thanks for welcoming hope to help.

current name was RealFb which should probably be set to LinuxFb or sth. like PosixFB?

you are currently using the windows virtual key codes. why not keep them? stdids.h is lend from win32 as well and is part of CtrlCore (for X11) so why not? save some work actually. i admit i havent spend much on this isue anyhow..i just copied the VK codes from WinUser.h to make it run.

did you think about the GUI_X11 problem and how to solve it?
for best performance , ImageBuffer should be initializable from a given buffer as well. thus we can spare ourselves some copy time..
but it has Buffer<char> pixels inside..maybe to use sth different? maybe it will be difficult because of ref count Image data takeovers.. but think about it.

Subject: Re: Rainbow, first iteration
Posted by mirek on Mon, 27 Jun 2011 19:05:33 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Mon, 27 June 2011 12:14
you are currently using the windows virtual key codes. why not keep them? stdids.h is lend from win32 as well and is part of CtrlCore (for X11) so why not? save some work actually. i admit i havent spend much on this isue anyhow..i just copied the VK codes from WinUser.h to make it run.

I believe it will be easier to define a new set for particular framebuffer backend...

Actual revision moved keys.h to WinFB.

Quote:
ImageBuffer should be initializable from a given buffer as well. thus we can spare ourselves some copy time..

Well, I expect 'backpaint' operation to be default for framebuffer for now. It is easier to develop and perhaps the required mode anyway.

Quote:
but it has Buffer<char> pixels inside..maybe to use sth different? maybe it will be difficult because of ref count Image data takeovers.. but think about it.

Actually, I already did and in the future, I plan to change BufferPainter to use raw binary buffer too (instead of ImageBuffer).

Mirek

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Tue, 28 Jun 2011 10:39:04 GMT
View Forum Message <> Reply to Message

we actually dont need a common codes set, the backends normally provide them as well. i.e. linux input subsystem...

do you plan to detach the gui backends from CtrlCore completely?
i'd find it pretty nice to have the backends in seperate packages..
i noticed that WinAlt does not use cham. is this planned?

generally, from where do you plan to have the final GUI backend selection? from the top ackage like it is now or from the CtrlCore package? i mean where the guiplatform.h is finally situated. it could be 'overridden' in case of special specification. but all provided backends could live at some bottom level, to be selected with a compile flag like WINFB, LINUXFB, MACFB (those that are

known and already exist).

this would reliev the user to specify it's own guiplatform, instead he could only take the Framebuffer package and select the backend via compileflag.

BTW: i commited the state of yesterday. what is missing, are the patch0 changes.. i found a mean to convert git patches to tortoise patches. so here it comes again.

File Attachments

1) patch0.svn.patch, downloaded 601 times

Subject: Re: Rainbow, first iteration
Posted by mirek on Tue, 28 Jun 2011 11:32:01 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 28 June 2011 06:39we actually dont need a common codes set, the backends normally provide them as well. i.e. linux input subsystem...

do you plan to detach the gui backends from CtrlCore completely?

I plan to keep Win32/X11 in CtrlCore. Most apps use just this and it would be 2 packages more in the list. In future, when MacOS and Android are available, I consider moving them there too.

Quote:
i noticed that WinAlt does not use cham. is this planned?

Unlike, WinAlt is only the first "proof of concept", nothing more.

Quote:
generally, from where do you plan to have the final GUI backend selection? from the top ackage like it is now or from the CtrlCore package? i mean where the guiplatform.h is finally situated. it could be 'overridden' in case of special specification. but all provided backends could live at some bottom level, to be selected with a compile flag like WINFB, LINUXFB, MACFB (those that are known and already exist).

I agree with compile flag and will be moving in that direction.

However, guiplatform.h placement is what it is supposed to be. I mean, first guiplatform.h in nest chain gets used. This is to allow everybody to groom his own backand withou fiddling with uppsrc code..

Quote:
this would reliev the user to specify it's own guiplatform, instead he could only take the Framebuffer package and select the backend via compileflag.

Actually, I believe we are heading right there.

BTW: i commited the state of yesterday. what is missing, are the patch0 changes.. i found a mean to convert git patches to tortoise patches. so here it comes again. [/quote]

I have found a problem there: In Image.h, GUI_X11 is not yet defined....

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Tue, 28 Jun 2011 11:37:06 GMT
View Forum Message <> Reply to Message

GUI_X11 not defined:

Quote:
patch0 has some fixes concerning PLATFORM_X11, which should be GUI_X11 or PLATFORM_POSIX now, please review it. there also arises a slight problem with Image, it still has some code PLATFORM_WIN32 and PLATFORM_X11 dependant. but due to elimination of PLATFORM_X11, which is available from Core level, GUI_X11 is available from CtrlCore level only, Image, which is included by CtrlCore does not know anything from GUI_X11. i dont have any solution for this. maybe the platform dependant stuff should be made a special class, which is implemented platform dependantly.

Subject: Re: Rainbow, first iteration
Posted by mirek on Tue, 28 Jun 2011 13:02:50 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 28 June 2011 07:37GUI_X11 not defined:

Quote:
patch0 has some fixes concerning PLATFORM_X11, which should be GUI_X11 or PLATFORM_POSIX now, please review it. there also arises a slight problem with Image, it still has some code PLATFORM_WIN32 and PLATFORM_X11 dependant. but due to elimination of PLATFORM_X11, which is available from Core level, GUI_X11 is available from CtrlCore level only, Image, which is included by CtrlCore does not know anything from GUI_X11. i dont have any solution for this. maybe the platform dependant stuff should be made a special class, which is implemented platform dependantly.

OK, sorry. Will deal with this soon...

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Wed, 29 Jun 2011 09:43:37 GMT
View Forum Message <> Reply to Message

i'm twiddleing a bit with SDL, it should be possible to treat it as Framebuffer backend, right ?

here, we got a problem with GUI_APP_MAIN define in After.h of Framebuffer. in PLATFORM_WIN32 environment, Framebuffer makes FBInit(hInstance). for SDL environment under win32 i need console app stuff

wouldn't it be good to just have an APP_MAIN and the rest via compile flags? as depricated means CONSOLE_APP_MAIN and GUI_APP_MAIN would redirect to APP_MAIN.. it's just an idea. it surely implies a lot more that i dont know of.

EDIT: tried to make APP_MAIN..see patch

the decision is made automatically, via flagGUI. the others redirect to APP_MAIN. i left DLL_APP_MAIN untouched.

File Attachments
1) patchappmain.svn.patch, downloaded 770 times

Subject: Re: Rainbow, first iteration
Posted by mirek on Wed, 29 Jun 2011 17:11:13 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Wed, 29 June 2011 05:43i'm twiddleing a bit with SDL, it should be possible to treat it as Framebuffer backend, right ?

here, we got a problem with GUI_APP_MAIN define in After.h of Framebuffer. in PLATFORM_WIN32 environment, Framebuffer makes FBInit(hInstance). for SDL environment under win32 i need console app stuff

wouldn't it be good to just have an APP_MAIN and the rest via compile flags? as depricated means CONSOLE_APP_MAIN and GUI_APP_MAIN would redirect to APP_MAIN.. it's just an idea. it surely implies a lot more that i dont know of.

EDIT: tried to make APP_MAIN..see patch

the decision is made automatically, via flagGUI. the others redirect to APP_MAIN. i left DLL_APP_MAIN untouched.

Hm, I guess that perhaps we should move GUI_APP_MAIN to final backend...

Subject: Re: Rainbow, first iteration
Posted by tojocky on Wed, 29 Jun 2011 18:47:34 GMT
View Forum Message <> Reply to Message

It seems that latest svn have not Framebuffer/FB.iml file
SVN version is: 3578.

Subject: Re: Rainbow, first iteration
Posted by cbpporter on Thu, 30 Jun 2011 06:33:27 GMT
View Forum Message <> Reply to Message

Just a quick though based on my experience with my Irrlicht back-ended GUI experimentation (without Rainbow).

What would be your recommendation on handling the lack of a windowing system? My solution is to use a "hidden" master window, which ownes everything, and the actual windows are actually fake windows, basically a slightly smarter StaticRect with frame.

Does rainbow have any support for defining an underlying windowing framework?

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Thu, 30 Jun 2011 07:04:25 GMT
View Forum Message <> Reply to Message

we recently thought about extending ownership behaviour to Ctrl itself as well, optionally. but it will bring a lot of problems, i.e. what to do if Remove() is called on an owned Ctrl. destroy it? what it program expects to keep it, but simply wants to hide.. it needs to be thought out well. a windowing framework also implies opening and closing of windows. how to do that in Upp? moving/resizing, layering is already well supported. but the other half (life cycle management) is needed to..

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Thu, 30 Jun 2011 09:14:42 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 29 June 2011 19:11Hm, I guess that perhaps we should move GUI_APP_MAIN to final backend...
i think so too. the APP_MAIN stuff posted above is running well, though. this would leave the user

with only one place to think about GUI switching, this beeing the flag.
a project would always compile with or without GUI flag, even if it uses CtrlCore stuff in a console app (might be a case someday), since CtrlCore/CtrlLib don't depend on GUI flag.

BTW: with that, i managed to override APP_MAIN in SDLFb.
so SDL is painting Upp now, too but it's all rudimentary. when it compiles and draws, one can relax and continue. for me to commit the starting point, i'd need to know which final direction this APP_MAIN stuff goes to.

i'd vote for this too, (but this is sort of taste dependant):
to move the specification of the implemented gui's, beeing native, to guiplatform.h
and have the defaulting take effect in CtrlCore.h
thus it uses the same means as upper guiplatform.h
which can #include "../uppsrc/guiplatform.h"
for backwards compatibility

CtrlCore.h:8

```
#ifndef GUIPLATFORM_INCLUDE

#ifdef PLATFORM_WIN32
#define flagWIN32GUI
#endif

#ifdef PLATFORM_POSIX
#define flagX11
#endif

//include again for defaults to activate
//guiplatform.h may not have ifndef/define protection
#include <guiplatform.h>
#endif
```

uppsrc/guiplatform.h

```
#ifdef flagWIN32GUI
#define GUIPLATFORM_INCLUDE <Ctrlcore/Win32Gui.h>
#endif

#ifdef flagX11
#define GUIPLATFORM_INCLUDE <Ctrlcore/X11Gui.h>
#endif
```

Subject: Re: Rainbow, first iteration
Posted by mirek on Mon, 04 Jul 2011 11:51:31 GMT
View Forum Message <> Reply to Message

Well, looks like it is getting somewhere:

A lot of issues to resolve in Framebuffer remaining, but I guess that Rainbow itself is now "proven".

File Attachments
1) Rainbow1.png, downloaded 1721 times



Subject: Re: Rainbow, first iteration
Posted by kohait00 on Mon, 04 Jul 2011 12:12:09 GMT
View Forum Message <> Reply to Message

looks niice..

the greatest issue for me currently (on SDL) is how to correctly process keys..

can you clarify a bit what K_DELTA and the others mean? K_ALT etc. are supposedly modifiers reported alongside the key codes.
when does Keyboard infrastructure expect ascii / unicode and when the +K_DELTA stuff. what is it for..

i think SDL will need to be a selfsustained (not Framebuffer dependant) thing. while it will loan most code from it.. but i hope to avoid it.. and keep it slim. it currently paints and processes mouse already.

BTW: is the APP_MAIN idea of any value? otherwise i'd drop my changes..

Subject: Re: Rainbow, first iteration
Posted by mirek on Mon, 04 Jul 2011 12:35:24 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Mon, 04 July 2011 08:12
BTW: is the APP_MAIN idea of any value? otherwise i'd drop my changes..

Well, for what is worth, I have moved GUI_APP_MAIN out of Framebuffer.

I do not see the reason for "nonempty" uppsrc/guiplatform.h at the moment (other than ortogonality, but I do not think it is that important here...)

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Mon, 04 Jul 2011 13:03:30 GMT
View Forum Message <> Reply to Message

no problem. was just a tweak.

do you have any K_DELTA infos ?

Subject: Re: Rainbow, first iteration
Posted by mirek on Mon, 04 Jul 2011 13:32:54 GMT
View Forum Message <> Reply to Message

K_DELTA is 65536, which > that all unicode characters (well, in theory it is not, but in practice, it is).

They idea is that Key is supposed to manage ALL key events - that includes (unicode) character events and other key events. Thus first 65536 values are reserved for character events and other events, like "K_UP" or "K_A" have K_DELTA (originally, in Win32, K_DELTA is added to noncharacter keycode).

Maybe we could figure out a better synonym But basically it means "this is not a character".

Mirek

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Mon, 04 Jul 2011 18:58:22 GMT
View Forum Message <> Reply to Message

did i understand you right, that key, first expects keydown/keyup events to be send, and then (after respective keyup) the corresponding char event? which windows is already doing natively, WM_KEYDOWN/UP and WM_CHAR.. (do you fake this for X11?)

Subject: Re: Rainbow, first iteration
Posted by mirek on Mon, 04 Jul 2011 20:57:41 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Mon, 04 July 2011 14:58did i understand you right, that key, first expects keydown/keyup events to be send, and then (after respective keyup) the corresponding char event?

Nope, characer is issued after keydown of course. For 'a' the sequence is K_A, 'a', K_A|K_KEYUP.

Quote:
which windows is already doing natively, WM_KEYDOWN/UP and WM_CHAR.. (do you fake this for X11?)

No need to fake, X11 is quite similiar, the only difference is that it only has KeyPress/KeyRelease and you convert to chars using some functions functions...

I guess these two levels naturally have to be on any system.

Mirek

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Mon, 04 Jul 2011 21:29:17 GMT
View Forum Message <> Reply to Message

OK.

i just commited a first shot SDLFb..
as i am not too experienced in SDL actually, there will be a lot to do yet.

we should think about grouping all the real Fb backends under Framebuffer. so the user actually does not need to add each and every backend at top level. Framebuffer could add them all and enable it with the right compile flag..

the speed of the stuff is actually really suboptimal due to full memcopy each repaint, i will try to fix this soon.

didn't watch the source development too carefully: have you added a BufferPainter creatable from an existing buffer yet? (specify the fb0 memmap directly as base, to speed up things).

Subject: Re: Rainbow, first iteration
Posted by mirek on Mon, 04 Jul 2011 21:44:05 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Mon, 04 July 2011 17:29
didn't watch the source development too carefully: have you added a BufferPainter creatable from an existing buffer yet? (specify the fb0 memmap directly as base, to speed up things).

Not yet.

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Tue, 05 Jul 2011 08:42:08 GMT
View Forum Message <> Reply to Message

during debug of sdl backend i noticed a bug:

Win32Wnd.cpp:857
should have parenthesis around (ctr ? .. : ..)
have the same line in SDLFb, and it properly quits only with paranthesis. so quite sure the compiler does not recognize the meaning properly.

```
 while(!EndSession() && !quit && (ctrl ? ctrl->IsOpen() && ctrl->InLoop() :
GetTopCtrls().GetCount()))
```

BTW
CtrlCore.h:1007

```
+ bool    IsIgnoreMouse() const                    { return ignoremouse; }
```

shouldn't
X11Proc.cpp:49 read this? similar to Win32Proc.. the K_ALT_KEY was missing

```
dword Ctrl::KEYtoK(dword key)
{
 if(key == K_ALT_KEY || key == K_CTRL_KEY || key == K_SHIFT_KEY)
  return key;
 if(GetCtrl()) key |= K_CTRL;
 if(GetAlt()) key |= K_ALT;
 if(GetShift()) key |= K_SHIFT;
 return key;
}
```

patch0 suggested PLATFORM_X11 changes for TDraw/util.cpp..
do you have other plans for them? i attach the this single patch again. i know of the PLATFORM_X11 legacy define..but uppsrc should be clean from it..uppsrc is not old legacy

EDIT: btw: is there a reason why the UWord doesnt work with WinGl? (asuming addint the right packeges). it crashes..

File Attachments

1) x11.patch, downloaded 527 times

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Tue, 05 Jul 2011 20:39:39 GMT
View Forum Message <> Reply to Message

this has gone recently must come back..

CtrlCore.h:1009

```
 bool    HasMouse() const;
```

Subject: Re: Rainbow, first iteration
Posted by mirek on Wed, 06 Jul 2011 07:44:31 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 05 July 2011 04:42during debug of sdl backend i noticed a bug:

Win32Wnd.cpp:857
should have parenthesis around (ctr ? .. : ..)
have the same line in SDLFb, and it properly quits only with paranthesis. so quite sure the compiler does not recognize the meaning properly.

```
 while(!EndSession() && !quit && (ctrl ? ctrl->IsOpen() && ctrl->InLoop() :
GetTopCtrls().GetCount()))
```

BTW
CtrlCore.h:1007

```
+ bool    IsIgnoreMouse() const                    { return ignoremouse; }
```

shouldn't
X11Proc.cpp:49 read this? similar to Win32Proc.. the K_ALT_KEY was missing

```
dword Ctrl::KEYtoK(dword key)
{
 if(key == K_ALT_KEY || key == K_CTRL_KEY || key == K_SHIFT_KEY)
  return key;
 if(GetCtrl()) key |= K_CTRL;
 if(GetAlt()) key |= K_ALT;
 if(GetShift()) key |= K_SHIFT;
 return key;
}
```

Applied.

Quote:
patch0 suggested PLATFORM_X11 changes for TDraw/util.cpp..
do you have other plans for them? i attach the this single patch again. i know of the

PLATFORM_X11 legacy define..but uppsrc should be clean from it..uppsrc is not old legacy

EDIT: btw: is there a reason why the UWord doesnt work with WinGl? (asuming addint the right packeges). it crashes..

TDraw is not canonical package. It is in the svn, but not in releaes. I will likely be moved to uppsrc2...

Mirek

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Fri, 08 Jul 2011 09:36:51 GMT
View Forum Message <> Reply to Message

didn't want to open another thread for that:

const correctness for a virtual
PusherCtrl.h & Button.cpp

String Pusher::GetDesc() const;

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Tue, 12 Jul 2011 09:31:33 GMT
View Forum Message <> Reply to Message

Win32Gui.h:324
X11Gui.h:227

void DrawDragRect(SystemDraw& w, const Rect& rect1, const Rect& rect2, const Rect& clip, int n,
Color color, uint64 pattern);

need to be added, otherwise it prevents rainbow/Paint from commpiling with GUI only. how should that be treaded generally? is this considered a 'to be implemented' function?

my goal is to have a bunch of backends to choose from, while developing can go GUI only, later one switches to GUI LINUXFB..

what about a meta package wihich incorporates all backends? so one only needs to add that one..it has the right flag switches set already..

@mirek: especcially for Framebuffer: could you summerize in short your thoughts/strategy on designning the FB* interface (FBInit, FBFlush.. etc)

Subject: Re: Rainbow, first iteration
Posted by mirek on Tue, 12 Jul 2011 13:04:35 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 12 July 2011 05:31Win32Gui.h:324
X11Gui.h:227

void DrawDragRect(SystemDraw& w, const Rect& rect1, const Rect& rect2, const Rect& clip, int n,
Color color, uint64 pattern);

need to be added, otherwise it prevents rainbow/Paint from commpiling with GUI only. how should that be treaded generally? is this considered a 'to be implemented' function?

...well, DrawDragRect is "under development" now, caused me some troubles (some indirectly). Interface is changing because of framebuffer, it shall be "to be implemented" function but with different signature.

Quote:
my goal is to have a bunch of backends to choose from, while developing can go GUI only, later one switches to GUI LINUXFB..

Yes.

Quote:
what about a meta package wihich incorporates all backends? so one only needs to add that one..it has the right flag switches set already..

Well, maybe. Do not consider this that important now. It is easy
to add required FB backends to the project.

Quote:
@mirek: especcially for Framebuffer: could you summerize in short your thoughts/strategy on designning the FB* interface (FBInit, FBFlush.. etc)

Anything specific?

I have to say some things are still changing..

Mirek

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Tue, 12 Jul 2011 13:46:17 GMT
View Forum Message <> Reply to Message

especially FBUpdate and FBFlush, when exactly are they called and what exactly are they supposed to do? i could guess from the names but it's not specific, i'll try to summerize in brief what i have found out so far:

Framebuffer is performing its drawing of the controls to a BufferPainter, whoose content can then be bitblitted to the real framebuffer. (what are the other ImageDraw, BackDraw etc.)

Framebuffer expects / calls some functions to help finish that process. it also expects the real backend to generate / derive the events/messages from your underlying hardware.

FBUpdate: reports the area that should directly be repainted / transfered to the underlying hardware framebuffer corresponding area, or should it schedule some kind of writeback..?

FBFlush: should it transfer everything (the entire area) from Ctrl::GetFrameBuffer to the underlying framebuffer section? is this not obsolete and could be done with FBUpdate(EntireSize)? Or does it work as a FBCommit after several FBUpdate calls? in this case, when FBUpdate directly memcpy's to the real framebuffer, no commit is needed, and FBFlush can remain {}

FBEndsession(): is this the means to signal to the Framebuffer package that the app wants to quit? (especially when there is one bare application, means no SDL or sth. there is no other means)? since SDL has the SDL_QUIT, which could map to the bool *quit flag from ProcessEvent.

FBSleep: ideally, this should sleep a fixed granularity of time, say 10ms, but be 'cancelable' or 'expireable' on arrival of new events to process.. if this is not possible, simply Sleep(10)?

FBIsWaitingEvent: should determine in a nonblocking manner, if there are messages or events to be processed. this is called in advance, prior to FBProcessEvent, which is called if messages/events to process really do exist. if there is no means to determine if events are there, simply return always true?

FBProcessEvent: here, *one single* backend message/event per call is dequeued and dispatched to upp understandable messages/events, using some custom translation mechanism..

but there are more things one needs to implement:

bool GetShift() { uint8* ka = SDL_GetKeyState(NULL); return ka[SDLK_LSHIFT] ||

```
ka[SDLK_RSHIFT]; }
bool GetCtrl()       { uint8* ka = SDL_GetKeyState(NULL); return ka[SDLK_LCTRL]  ||
ka[SDLK_RCTRL]; }
bool GetAlt()        { uint8* ka = SDL_GetKeyState(NULL); return ka[SDLK_LALT]   ||
ka[SDLK_RALT]; }
bool GetCapsLock()    { uint8* ka = SDL_GetKeyState(NULL); return ka[SDLK_CAPSLOCK]; }
bool GetMouseLeft()   { return (SDL_GetMouseState(NULL,NULL) &
SDL_BUTTON(SDL_BUTTON_LEFT)); }
bool GetMouseRight()  { return (SDL_GetMouseState(NULL,NULL) &
SDL_BUTTON(SDL_BUTTON_RIGHT)); }
bool GetMouseMiddle() { return (SDL_GetMouseState(NULL,NULL) &
SDL_BUTTON(SDL_BUTTON_MIDDLE)); }
```

the Keys.h assignments, for K_* of upp, caution, it uses some special structure, K_ALT and K_ALT_KEY are not the same..

one does normally NOT need to define the starting point of the application (done in Framebuffer), but can override it (see SDLFb)

```
#define GUI_APP_MAIN \
```

Subject: Re: Rainbow, first iteration
Posted by mirek on Tue, 12 Jul 2011 16:02:08 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 12 July 2011 09:46especially FBUpdate and FBFlush, when exactly are they called and what exactly are they supposed to do?

Well, unfortunately, meanings are not yet 100% fixed, as I am still solving some issues with mouse cursor fluidity, but actual broad meaning is:

FBUpdate - tell that some portion of framebuffer is to be moved to the screen. At the moment, it is not required nor prohibited that the update happens immediately.

FBFlush - at the end of function, all previous FBUpdate should be visible on the screen. If they are moved by FBUpdate, FBFlush can be NOP.

Quote:

Framebuffer is performing its drawing of the controls to a BufferPainter, whoose content can then be bitblitted to the real framebuffer.

Yes.

Quote:
(what are the other ImageDraw

Provides means for widget to draw on its view area directly (without Refresh/Paint).

Quote:
, BackDraw etc.)

Not used for Framebuffer; on system with direct painting, it is used when BackPaint mode is active (to provide backbuffer).

Quote:
Framebuffer expects / calls some functions to help finish that process.

Yes.

Quote:
it also expects the real backend to generate / derive the events/messages from your underlying hardware.

Yes, as part of event processing.

Quote:
FBEndsession(): is this the means to signal to the Framebuffer package that the app wants to quit?

No, it is basically means that GUI is to be shut down (e.g. computer switched off).

Of course, if we run "windowed fb", then equivalent is closing the host window.

BTW, dealing with end-session event is somewhat unfinished bussiness in U++...

Quote:
FBSleep: ideally, this should sleep a fixed granularity of time, say 10ms, but be 'cancelable' or 'expireable' on arrival of new events to process.. if this is not possible, simply Sleep(10)?

Yes.

Quote:
FBIsWaitingEvent: should determine in a nonblocking manner, if there are messages or events to be processed. this is called in advance, prior to FBProcessEvent, which is called if messages/events to process really do exist. if there is no means to determine if events are there, simply return always true?

Well, that would be bad - it has to return false sometimes, as some processing is tied to "empty input queue" condition, e.g. painting or timer.

Quote:
FBProcessEvent: here, *one single* backend message/event per call is dequeued and dispatched to upp understandable messages/events, using some custom translation mechanism..

Yep.

Quote:
but there are more things one needs to implement:

```
bool GetShift()       { uint8* ka = SDL_GetKeyState(NULL); return ka[SDLK_LSHIFT] ||
ka[SDLK_RSHIFT]; }
bool GetCtrl()        { uint8* ka = SDL_GetKeyState(NULL); return ka[SDLK_LCTRL]  ||
ka[SDLK_RCTRL]; }
bool GetAlt()         { uint8* ka = SDL_GetKeyState(NULL); return ka[SDLK_LALT]   ||
ka[SDLK_RALT]; }
bool GetCapsLock()    { uint8* ka = SDL_GetKeyState(NULL); return ka[SDLK_CAPSLOCK]; }
bool GetMouseLeft()   { return (SDL_GetMouseState(NULL,NULL) &
SDL_BUTTON(SDL_BUTTON_LEFT)); }
bool GetMouseRight()  { return (SDL_GetMouseState(NULL,NULL) &
SDL_BUTTON(SDL_BUTTON_RIGHT)); }
bool GetMouseMiddle() { return (SDL_GetMouseState(NULL,NULL) &
SDL_BUTTON(SDL_BUTTON_MIDDLE)); }
```

Careful here. The values MUST be frozen at the moment of input event. Same for GetMousePos.

Quote:
the Keys.h assignments, for K_* of upp, caution, it uses some special structure, K_ALT and K_ALT_KEY are not the same..

Yep.

Mirek

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Wed, 13 Jul 2011 08:46:47 GMT
View Forum Message <> Reply to Message

now thats sort of preliminary doc thanks for clearifying this..

another big question in embedded systems is, after i have talked again with my boss, support for 565 (16Bit) drawn backend, means BufferPainter with 16 bit support. now this is quite important, since this is the second most case in ES. with theese 2, we'd probably habve 95 % cases.

we could blit accordingly in FBUpdate, but this would really have a performance hit.

what means do we have to do that?

EDIT:
i know you wrote:
message 9781
Quote:

Well, our take (since the beginning) is that supporting other than 24 bit colors in the interface is not worth the trouble, especially supporting palettes is useless (the notable exception here is Image export/import infrastructure, where it is still required).

It was true in 1999 when we started and it is even more true now

That does not mean U++ apps would not work, they do, just look worse. U++ sets some default palette and uses it.

for ES this is not neccessarily well but i know, upp wasn't designed with ES in mind... so the question is, what to is the esiest path to have that? an own SystemDraw with a special BufferPainter?

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Wed, 13 Jul 2011 15:48:19 GMT
View Forum Message <> Reply to Message

LinuxFb: makes progress i've got mouse events processing working. it has some SDL GPLed code, which i need to rewrite first, before i can commit it. keyboard processing is on the way, but that will be bit more dificult i think..

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Thu, 14 Jul 2011 13:09:46 GMT
View Forum Message <> Reply to Message

the current Ctrl::EndSession uses the Ctrl::fbEndSession, which is not evaluated anywhere..

is it supposed to be evaluated in FBEndSession?

Subject: Re: Rainbow, first iteration
Posted by mirek on Fri, 15 Jul 2011 13:13:36 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Wed, 13 July 2011 04:46now thats sort of preliminary doc thanks for clearifying this..

another big question in embedded systems is, after i have talked again with my boss, support for 565 (16Bit) drawn backend, means BufferPainter with 16 bit support. now this is quite important, since this is the second most case in ES. with theese 2, we'd probably habve 95 % cases.

we could blit accordingly in FBUpdate, but this would really have a performance hit.

what means do we have to do that?

EDIT:
i know you wrote:
message 9781
Quote:

Well, our take (since the beginning) is that supporting other than 24 bit colors in the interface is not worth the trouble, especially supporting palettes is useless (the notable exception here is Image export/import infrastructure, where it is still required).

It was true in 1999 when we started and it is even more true now

That does not mean U++ apps would not work, they do, just look worse. U++ sets some default palette and uses it.

for ES this is not neccessarily well but i know, upp wasn't designed with ES in mind... so the question is, what to is the esiest path to have that? an own SystemDraw with a special BufferPainter?

I am afraid that even in this case, the best solution is still FBUpdate converting 24->16.

I believe that it should not be much really much slower.

That said, 16bit SystemDraw is still possible, but it is a big amount of work.

Mirek

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Fri, 15 Jul 2011 14:48:39 GMT

View Forum Message <> Reply to Message

what would cover the work nesseccary?

Painter implementation for 16 bit??

FBEndSession & EndSession: any hints here?

BTW: is there a uniform way to gracuefully stop the application, besides calling the current TopWindow::Close? i mean sth like EndSession would be really great..to ensure the GUI_APP_MAIN can finish its work, which would not be using exit(0).. i need sth without a TopWindow instance as base...

Subject: Re: Rainbow, first iteration
Posted by mirek on Fri, 15 Jul 2011 15:12:33 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Thu, 14 July 2011 09:09the current Ctrl::EndSession uses the Ctrl::fbEndSession, which is not evaluated anywhere..

is it supposed to be evaluated in FBEndSession?

Sorry, but I am yet not sure what is an overall correct behaviour w.r.t. system shutdown.

This thing was actually quite exposed by UWord with framebuffer, where it produces a lot of memory leaks on closing the host window (which is "system shutdown" for framebuffer).

I am considering add a new method, "Shutdown", to Ctrl interface. The behaviour in the case of system shutdown would then try to close all enabled TopWindows by:

- Calling Shutdown first
- If the window is not closed after Shutdown ends, "invoking close button". That should either close the window immediatelly, or invoke "Save? Yes No Cancel" sort of dialog (where "Cancel" does not make sense here, that is why there is the need for Shutdown).
- Repeat until there are any enabled windows that can be closed
- After that, leave all event loops (I mean all levels, so that we can properly exit thrugh leaving GUI_APP_MAIN)

Subject: Re: Rainbow, first iteration
Posted by mirek on Fri, 15 Jul 2011 16:17:05 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Fri, 15 July 2011 10:48what would cover the work nesseccary?

Painter implementation for 16 bit??

Yes, something like that.

Actually, now thinking about it, it would have to be that much hard. But I would rather try conversion at FBUpdate first anyway.

BTW, nice if not so much obvious trick during the conversion might be to test if the RGB source value is not the same as previous one (cache last value). Well, at least on CPU with branch prediction... I would say you would get 80% of "16-painter" performance this way.

Subject: Re: Rainbow, first iteration
Posted by nlneilson on Sat, 16 Jul 2011 22:38:09 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 15 July 2011 08:12Sorry, but I am yet not sure what is an overall correct behaviour w.r.t. system shutdown.

Here is something to consider for shutdown as I had problems with this in U++ (and Java).
http://www.ultimatepp.org/forum/index.php?t=msg&goto=300 02&#msg_30002

Shutting down each thread INSIDE the thread.

```
void endX(){end = true; Sleep(2000);}

GUI_APP_MAIN{
 GPSx2().Run();
 ...
 ...
 endX();

// and then in each thread
     Sleep(1000);
  while (CommPort.ReadDataWaiting() ) {
   if(end) break;
    try{
// with global
void endX();
bool end;
```

So if "end=true;" the timing is such that each thread can shut itself down.

In Java it was handled with this one line:
```
    af.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

Maybe something similar to this could be implemented in U++ as a way to shut down apps correctly.

Thread::ShutdownThreads(); Did not work for me!

Subject: Re: Rainbow, first iteration
Posted by daveremba on Mon, 18 Jul 2011 23:56:10 GMT
View Forum Message <> Reply to Message

For the MacOSX port, after looking at Rainbow,
I was thinking of using OpenGL for drawing,
and a small amount of Cocoa only for
window creation and events/input. I am not sure yet,
but you've already done the OpenGL work in Rainbow and
it ought to be portable. (so maybe much less need
for Cocoa code in Objective-C and mixed compiling
and linking)

Also, on SVN, what is uppsrc2, and uppdev?

Dave

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Tue, 19 Jul 2011 07:17:13 GMT
View Forum Message <> Reply to Message

sounds good. i don't mind to circumvent the macosx stuff as much as possible.. if we emulate the look and feel anyway and dont need the native means at all, that's more than perfect.

Subject: Re: Rainbow, first iteration
Posted by mirek on Tue, 19 Jul 2011 08:33:11 GMT
View Forum Message <> Reply to Message

daveremba wrote on Mon, 18 July 2011 19:56For the MacOSX port, after looking at Rainbow,
I was thinking of using OpenGL for drawing,
and a small amount of Cocoa only for
window creation and events/input. I am not sure yet,
but you've already done the OpenGL work in Rainbow and

it ought to be portable. (so maybe much less need
for Cocoa code in Objective-C and mixed compiling
and linking)

Not sure about OpenGL. I guess it should not be THAT hard to implement DrawText, DrawRect and DrawImage in Quartz2D...

Besides, the main problem is DrawText and font management - and there is no support for them in OpenGL.

Quote:
Also, on SVN, what is uppsrc2, and uppdev?

uppsrc2 is a place where obsolete packages are moved.

uppdev is a place for "development packages", either tests or new packages being developed.

In short: you can ignore both

Mirek

Subject: Re: Rainbow, first iteration
Posted by unodgs on Tue, 19 Jul 2011 08:42:15 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 19 July 2011 04:33
Besides, the main problem is DrawText and font management - and there is no support for them in OpenGL.

I'm trying to do something similar qt developers did (http://labs.qt.nokia.com/2011/07/15/text-rendering-in-the-qm l-scene-graph) I have a working shader, but the main problem is distance field generator (I'm trying to port one from qt5)

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Tue, 19 Jul 2011 13:01:29 GMT
View Forum Message <> Reply to Message

@unodogs: i am unable to find out what difference there is in PaintGl using WinGl and Paint using WinGl..the first compiles and runs, the latter compiles but crashes..

is there anything special about PaintGl? in theory, we should be able to enhance whichever

application by more graphical backends. that's why i felt bold enough to try to use Paint with WinGl as well. any help here?

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Tue, 19 Jul 2011 13:18:28 GMT
View Forum Message <> Reply to Message

@mirek:

the EndSession for Framebuffer works quite well now. but there is a incoherency currently.

Win32Wnd.cpp(95):GLOBAL_VAR(bool, Ctrl::EndSession)
yields bool& Ctrl::EndSession();
which is not compliant to void Ctrl::EndSession() current behaviour.
this is used in Win32 and WinAlt, but X11 backend doesnt have it at all.

i'd suggest to have a usual bool Ctrl::endSession here, and move over to have a static void EndSession() as a public interface part. so this would become portable means.

EDIT: i commited in WinAlt the respective changes. maybe this whole EndSession thing can even go to CtrlCore, not to have to repeat the mess for each backend..

Subject: Re: Rainbow, first iteration
Posted by mirek on Tue, 19 Jul 2011 19:29:13 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 19 July 2011 09:18@mirek:

the EndSession for Framebuffer works quite well now. but there is a incoherency currently.

Yes, I know, RM #86...

Mirek

Subject: Re: Rainbow, first iteration
Posted by unodgs on Tue, 19 Jul 2011 20:28:45 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 19 July 2011 09:01@unodogs: i am unable to find out what difference

there is in PaintGl using WinGl and Paint using WinGl..the first compiles and runs, the latter compiles but crashes..

is there anything special about PaintGl? in theory, we should be able to enhance whichever application by more graphical backends. that's why i felt bold enough to try to use Paint with WinGl as well. any help here?

PaintGl was simply a test application for WinGl backend. I will remove it soon. Could you pull the latest sources and tell me if it still crashes (and where)? I cannot reproduce it..

Ok I know what is your problem. Please copy tahoma.fnt and tahoma.png files from PaintGl to Paint folder.. (later basic fonts will be automatically linked or generated on the fly)

Subject: Re: Rainbow, first iteration
Posted by daveremba on Wed, 20 Jul 2011 04:17:46 GMT
View Forum Message <> Reply to Message

Are there more font capabilities needed beyond
the choices in the layout editor?

Quote:Not sure about OpenGL. I guess it should not be THAT hard to implement DrawText, DrawRect and DrawImage in Quartz2D...

Besides, the main problem is DrawText and font management - and there is no support for them in OpenGL.

There are some font support libraries for OpenGL:
basic support in GLUT, some newer libraries for the
gamers, and an older one for FreeType:
http://oglft.sourceforge.net/

--

I'm continuing to look also at Qt, wx, and Firefox.
They solve the problem of late-binding in Objective-C
and avoid the NIB file (static layout).

If anyone is curious, here is a link to an Apple man page explaining Cocoa
as it relates to the OS architecure.

As far as mixing Cocoa and C++ that is certainly do-able,
even from UPP theide by adding some options to gcc:

gcc main.m cocoa_test1AppDelegate.m -framework Foundation -framework Cocoa -o test

Cocoa apps also want to be wrapped in a folder called a
"bundle" with an info.plist XML file that defines it to
be an "app" to MacOSX. Xcode makes one when apps are built,
and bundle could also be built as a post-processing step in theide.

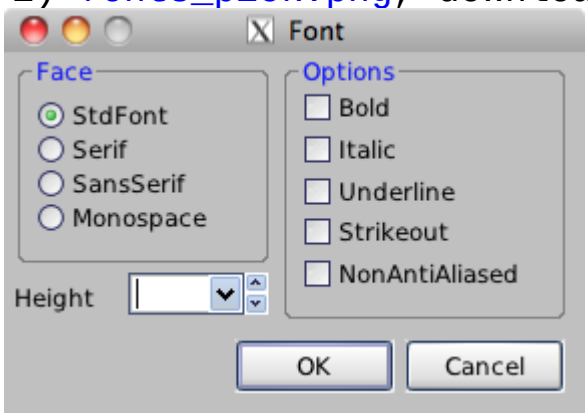
--

Lastly, what is chameleon? Theme-related? For example,
how does one change the background or default color of controls
in the layout?

Dave

File Attachments

1) fonts_pick.png, downloaded 1586 times



Subject: Re: Rainbow, first iteration
Posted by mirek on Wed, 20 Jul 2011 05:13:49 GMT
View Forum Message <> Reply to Message

daveremba wrote on Wed, 20 July 2011 00:17Are there more font capabilities needed beyond
the choices in the layout editor?

Yes.

Check Font class in Draw.

We need the complete metrics of glyphs, capability of render unicode, the list of all fonts, and to

identify 3 basic fonts (serif, sans-serif, monospace).

Quote:
I'm continuing to look also at Qt, wx, and Firefox.
They solve the problem of late-binding in Objective-C
and avoid the NIB file (static layout).

Excelent.

Quote:
gcc main.m cocoa_test1AppDelegate.m -framework Foundation -framework Cocoa -o test

I suspect that "-framework" is link step option, correct?

Quote:
Cocoa apps also want to be wrapped in a folder called a
"bundle" with an info.plist XML file that defines it to
be an "app" to MacOSX. Xcode makes one when apps are built,
and bundle could also be built as a post-processing step in theide.

My bet is there will be some commandline way to make the bundle. Plus we will have to get theide to generate the .xml (perhaps replacing only the name of application).

Quote:

Lastly, what is chameleon? Theme-related? For example,
how does one change the background or default color of controls
in the layout?

Yes, it is skinning system. It is not so much about skinning individial widgets, but the whole GUI.

There is simple example covering basics:

http://www.ultimatepp.org/reference$Chameleon$en-us.html

(well, in this case, individual widgets are skinned, but that is not the common use).

Mirek

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Wed, 20 Jul 2011 08:28:10 GMT
View Forum Message <> Reply to Message

@unodogs:
Quote:
copy tahoma.fnt and tahoma.png files from PaintGl to Paint

that was the missing part, how could i be that blind. expected sth. in code.. runs like a charm

is it possible to move the fonts somehow in WinGl to relief the user from that?

Subject: Re: Rainbow, first iteration
Posted by unodgs on Wed, 20 Jul 2011 08:48:24 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Wed, 20 July 2011 04:28@unodogs:
Quote:
copy tahoma.fnt and tahoma.png files from PaintGl to Paint

is it possible to move the fonts somehow in WinGl to relief the user from that?
First I wanted to embed fonts in WinGl package using brc files. Unfortunately brc files cannot be properly linked in 64b mode. So for now I'll just use iml file to store font's texture and String variable to store font definition. I don't like it (it hard to replace and edit) but I don't see other solution right now. Like I said in one of my previous posts I want to use qt's technique to generate font definition on the fly, but I need some time to do that.

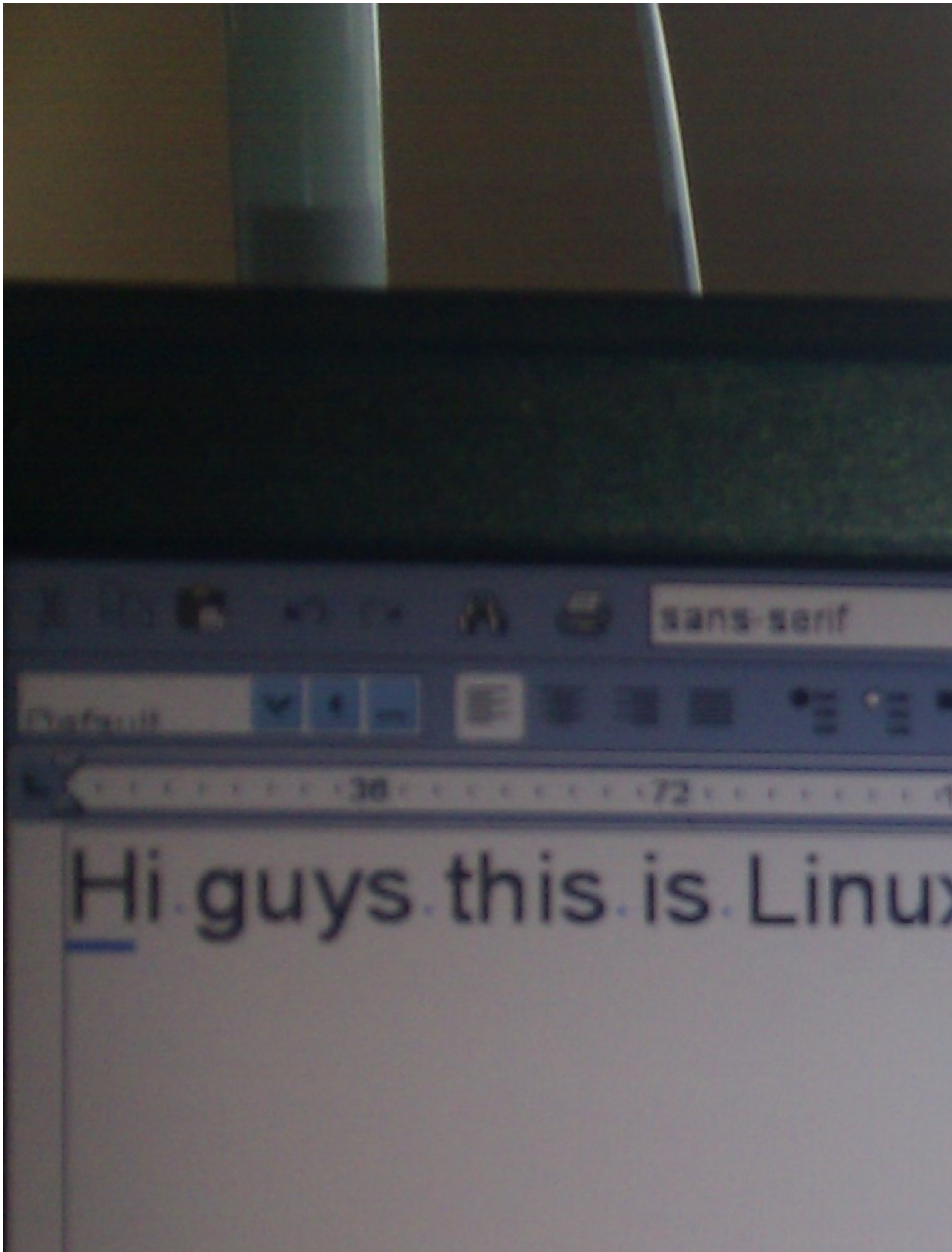
Subject: Re: Rainbow, first iteration
Posted by kohait00 on Wed, 20 Jul 2011 13:43:38 GMT
View Forum Message <> Reply to Message

LinuxFb works now in most parts. one thing still to manage is the view draw optimization..any idea on what could help here?

EDIT: there are some drawing problems with widgets that i can remember to already have seen..is it an issue with static controls initialized before the GUI?? no idea how to fix that..

File Attachments

1) IMG_20110720_155101.jpg, downloaded 2124 times


sans-serif
Default
36
72
Hi guys this is Linux

Subject: Re: Rainbow, first iteration
Posted by mirek on Wed, 20 Jul 2011 15:04:03 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Wed, 20 July 2011 09:43LinuxFb works now in most parts. one thing still to manage is the view draw optimization..any idea on what could help here?

What exactly is the problem?

Quote:
EDIT: there are some drawing problems with widgets that i can remember to already have seen..is it an issue with static controls initialized before the GUI?? no idea how to fix that..

It looks like metrics problem, most likely font metrics issue. Try DDUMP(GetStdFontCy()) (should be something between 10 - 20).

Other than that, congratulations. I guess we are heading in the right direction.

I plan to mostly finish framebuffer this weekend.

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Wed, 20 Jul 2011 22:10:01 GMT
View Forum Message <> Reply to Message

Thanks. Thema Problem With FBUpdate rect...IT curently copies Thema entire framebuffer...which slooooooows down. Same thing Tor sdlfb.

I will Check font metrics.thanks for the hint.

Subject: Re: Rainbow, first iteration
Posted by daveremba on Wed, 20 Jul 2011 22:37:30 GMT
View Forum Message <> Reply to Message

gcc main.m cocoa_test1AppDelegate.m -framework Foundation -framework Cocoa -o test
mirek wrote on Tue, 19 July 2011 22:13
I suspect that "-framework" is link step option, correctYes:-framework name[,suffix]
"This option tells the linker to search for `name.framework/name' the framework search path."
...
The default framework search path is /Library/Frameworks then /System/Library/FrameworksAn UPP installation for all users (treating UPP as any MacOSX product/application) can put its libraries there.mirek wrote on Tue, 19 July 2011 22:13My bet is there will be some commandline way to make the bundle. Plus we will have to get theide to generate the .xml (perhaps replacing

only the name of application).It isn't too complex and there is documentation.
I made a bundle of the X11 version of theide, it works
the same except that a terminal shell window is not
created (same as a "subsystem: windows" app).
--
mirek wrote on Tue, 19 July 2011 22:13We need the complete metrics of glyphs, capability of render unicode, the list of all fonts, and to identify 3 basic fonts (serif, sans-serif, monospace).about "render unicode" ...

I didn't see UTF-16/unicode support in UPP. Is that
something planned to be added?

Dave

Subject: Re: Rainbow, first iteration
Posted by mirek on Thu, 21 Jul 2011 09:28:04 GMT
View Forum Message <> Reply to Message

daveremba wrote on Wed, 20 July 2011 18:37
I didn't see UTF-16/unicode support in UPP. Is that
something planned to be added?

Dave

Well, perhaps it is because U++ is completely UTF-16 based?

For compatibility reason, you can set 8-bit encoding on app-wide basis, but recommended default there is UTF-8.

In any case, all fontmetrics info is UTF-16 and the basic Draw::DrawTextOp method expects UTF-16...

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Thu, 21 Jul 2011 10:08:15 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 20 July 2011 17:04
It looks like metrics problem, most likely font metrics issue. Try DDUMP(GetStdFontCy()) (should be something between 10 - 20).

you were right.. it reports 0.
where does the font stuff get initialized or how to fix it?

cheers

Subject: Re: Rainbow, first iteration
Posted by mirek on Thu, 21 Jul 2011 10:14:22 GMT
View Forum Message <> Reply to Message

void Font::InitStdFont()

Related files are Draw/Font.cpp, FontFc.cpp (for fontconfig/freetype).

Note: I am afraid that for the full Mac implementation, we will need "FontQuartz.cpp". But I guess fontconfig/freetype can be temporalily replacements there...

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Thu, 21 Jul 2011 14:16:59 GMT
View Forum Message <> Reply to Message

it worked out fine with

```
 Font::SetStdFont(ScreenSans(12));
```

but is there any possibility to have the font handling completely seperated as per backend basis? there is still code mixed in Draw/Font.cpp and so on..

Subject: Re: Rainbow, first iteration
Posted by mirek on Thu, 21 Jul 2011 16:40:27 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Thu, 21 July 2011 10:16it worked out fine with

```
 Font::SetStdFont(ScreenSans(12));
```

but is there any possibility to have the font handling completely seperated as per backend basis? there is still code mixed in Draw/Font.cpp and so on..

Well, I am considering it, but:

a) there are only 3 possible font metrics system we have to support
b) the code is relatively small without too many caveats (unlike GUI backend).

So I guess it is now worth it....

Subject: Re: Rainbow, first iteration
Posted by mirek on Thu, 21 Jul 2011 16:43:08 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Thu, 21 July 2011 10:16it worked out fine with

```
 Font::SetStdFont(ScreenSans(12));
```

..I guess this call should be placed somewhere in either Framebuffer or ChStd..

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Thu, 21 Jul 2011 16:48:24 GMT
View Forum Message <> Reply to Message

not neccessarily..
Font.cpp
InitStdFont() needs a restructuring, because, currently, win fonts are know, and can SyncStdFont, while i.e. X11 also does the
Font::SetStdFont(ScreenSans(12)); trick but later in init, which calls SyncStdFont as well. so it's kinda messy..

so best thing is to make InitStdFont or sth also backend dependant..

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Fri, 22 Jul 2011 09:42:58 GMT
View Forum Message <> Reply to Message

@mirek:
Quote:
So I guess it is now worth it....

... to change behavior or to leave it as is ?

meanwhile i will move the correspinding thing to InitFB..depending on POSIX...since WIN32 does init it..

Subject: Re: Rainbow, first iteration
Posted by cbpporter on Fri, 22 Jul 2011 11:23:18 GMT
View Forum Message <> Reply to Message

I was wondering what are required steps to get a new Rainbow backend started. Nothing too complicated, only something that can render the contents of a TopWindow (no border/sizing required) with a Paint method. I do not need fonts, Painter or other more advanced features.

Can you give me any instructions/hint/first steps? Do I need the rainbow from trunk or branches. I am guessing the skeleton package is a striped down starting point.

I want to try to investigate during weekend if my Irrlicht windowing system can be converted to a Rainbow backend.

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Fri, 22 Jul 2011 12:17:05 GMT
View Forum Message <> Reply to Message

@mirek:

I am experimenting with a ImageBuffer hack (including Buffer<>) to pass the fb surface directly as underlaying for Ctrl::framebuffer. the speed improvements for win/linux, fb0/SDL are considerable. especially in terms of responsiveness of the input handling.
while video handling under pc systems wont see much advantage, direct handling in embedded will really profit.

maybe we should really think about making upp directly drawable on fb surface, and making double buffering optional..

i will provide a patch to test it for your self. it is really quick.

@cbpporter: consider using WinAlt, in comparison with Skeleton, to see what exactly is needed. framebuffer is really a good choice as well, if irrlicht can offer a direct surface to draw to..fb based would not profit from optimized draw operations of irrlich though. see progress of MacOS port..it's getting exciting imagine upp running on all the backends..this is a huge advantage.

EDIT: the attached patch is a quick hack to enable the usage of a n arbitrary memory chunk under ImageBuffer. Buffer<> is extended with a ctor and ability to hold not owned memory. ImageBuffer is extended to be constructable from a supplied Buffer<RGBA> plus Size, Ctrl::framebuffer is made public, so it's Buffer can be replaced.

while Linux/SDL somewhat does not show improvements, win/SDL are really responsive. LinuxFb heavily profits from the speed. again, my vote is for making this possible, so upp can run on at least some embeddeds (we would love to use it in our company in such way)

File Attachments
1) patchdirect.svn.patch, downloaded 510 times

Subject: Re: Rainbow, first iteration
Posted by mirek on Fri, 22 Jul 2011 12:51:39 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Fri, 22 July 2011 07:23I was wondering what are required steps to get a new Rainbow backend started. Nothing too complicated, only something that can render the contents of a TopWindow (no border/sizing required) with a Paint method. I do not need fonts, Painter or other more advanced features.

Can you give me any instructions/hint/first steps? Do I need the rainbow from trunk or branches. I am guessing the skeleton package is a striped down starting point.

I want to try to investigate during weekend if my Irrlicht windowing system can be converted to a Rainbow backend.

This all is very much under intense development.

What is 'rainbow' is first attempts at different backends for U++. So in theory, you do not need rainbow nest to create GUI backend for U++. In practicem it is a good starting point....

The best start is thus to checkout 'rainbow' and try Paint or UWord 'examples'.

Mirek

Subject: Re: Rainbow, first iteration
Posted by mirek on Fri, 22 Jul 2011 13:02:24 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Fri, 22 July 2011 08:17@mirek:

I am experimenting with a ImageBuffer hack (including Buffer<>) to pass the fb surface directly as underlaying for Ctrl::framebuffer. the speed improvements for win/linux, fb0/SDL are considerable. especially in terms of responsiveness of the input handling.
while video handling under pc systems wont see much advantage, direct handling in embedded will really profit.

maybe we should really think about making upp directly drawable on fb surface, and making double buffering optional..

I am definitely NOT opposed to the idea; but let us say it is the 'next level'.

I also see two related issues there - first I believe ImageBuffer and Painter should have some way how to directly paint on external buffer. This part is relatively simple.

Then, completly different issue is Framebuffer with direct draw support - some things have to be significantly different there and more complicated (e.g. to have mouse cursor that is not flickering

is a little bit harder without backbuffer).

OTOH, while the first part (Buffer/Painter) has to be implemented in canonical packages by core developers, new Framebuffer could be developed now by anybody and in fact, it does NOT have to relly on ImageBuffer changes - it is not written in the stone that you have to use Painter...

That said, I also see big oportunity in DrawText optimization. Current Painter implementation is "pendantic" and always expects you want to do something different with each glyph than to fill it. Thus when doing DrawText, everything is rendered as quadratic curves and then filled. I think that some sort of bitmap caching for DrawText could bring dramatic improvement in FB gui responsivnes. If I ever get to it, this will be the first thing I will try to improve....

Mirek

i will provide a patch to test it for your self. it is really quick.

@unodogs: consider using WinAlt, in comparison with Skeleton, to see what exactly is needed. framebuffer is really a good choice as well, if irrlicht can offer a direct surface to draw to..fb based would not profit from optimized draw operations of irrlich though. see progress of MacOS port..it's getting exciting imagine upp running on all the backends..this is a huge advantage.[/quote]

Subject: Re: Rainbow, first iteration
Posted by kohait00 on Wed, 27 Jul 2011 13:49:11 GMT
View Forum Message <> Reply to Message

i'm definitely not into the whole Font's stuff, so i leave it for you but with drawing, i could try helping.

in fact, what is needed, is to at least have a 24 bit support (alpha-less). many current embedded framebuffers offer full 24-bit RGB backend, so firing up dma for the chunk is easy and you got a touchscreen working and a quite nice and responsive GUI.

couldnt the drawing backends be made sort of similar to rainbow? exchangeable? see, internally, upp could still use full RGBA/Color format. but depending of which format you chose for i.e. BufferPainter, a DrawLineOp or the like will call some coresponding helper functions, probably from the rasterizer. and as far as i can imagine, it would only affect the real local buffer painter. all the other painting backends will be connected to native OS means anyway, i.e WIN32 API, X11 drawing, GL rawing, MACOSx drawing. they are not buffer drawing

but i have to admit that i have not enough experience and in-depth knowledge of the whole drawing system, reading code is good, but a view phrases from the devs is best help available..

Subject: Re: Rainbow, first iteration
Posted by mirek on Thu, 28 Jul 2011 15:04:39 GMT

kohait00 wrote on Wed, 27 July 2011 09:49
couldnt the drawing backends be made sort of similar to rainbow? exchangeable? see, internally, upp could still use full RGBA/Color format. but depending of which format you chose for i.e. BufferPainter, a DrawLineOp or the like will call some coresponding helper functions, probably from the rasterizer. and as far as i can imagine, it would only affect the real local buffer painter.

Well, I guess something like this is quite possible to do... as in fact, there are only 2 "final" fillers in BufferPainter. Means RGBA is, I believe, only matter of a dozen of virtual functions.

Needless to say, however, that doing so we should also carefully consider refactoring Painter for multithreaded filling.

Another possible option is to write to backbuffer, but somehow remember "monocolor" areas, then render those with "memset" equivalent...

Subject: Re: Rainbow, first iteration
Posted by rett on Thu, 01 Sep 2011 07:54:07 GMT

Hello!

I traying compile UWord in rainbow package.
params:
GUI LINUXFB
GCC Optimal

after build a have errors:
----- CtrlLib (GUI LINUXFB GCC LINUX POSIX) (1 / 8)
----- RichEdit (GUI LINUXFB GCC LINUX POSIX) (2 / 8)
----- PdfDraw (GUI LINUXFB GCC LINUX POSIX) (3 / 8)
----- plugin/jpg (GUI LINUXFB GCC LINUX POSIX) (4 / 8)
----- LinuxFb (GUI LINUXFB GCC LINUX POSIX) (5 / 8)
----- Framebuffer (GUI LINUXFB GCC LINUX POSIX) (6 / 8)
----- Painter (GUI LINUXFB GCC LINUX POSIX) (7 / 8)
----- UWord (GUI LINUXFB MAIN GCC LINUX POSIX) (8 / 8)
Linking...
c++: /home/rett/upp/_out/rainbow/CtrlLib/GCC.Gui.Linuxfb/CtrlLib.a: No such file or directory
c++: /home/rett/upp/_out/rainbow/RichEdit/GCC.Gui.Linuxfb/RichEdit.a: No such file or directory
c++: /home/rett/upp/_out/rainbow/PdfDraw/GCC.Gui.Linuxfb/PdfDraw.a: No such file or directory
c++: /home/rett/upp/_out/rainbow/plugin/jpg/GCC.Gui.Linuxfb/jpg.a: No such file or directory
c++: /home/rett/upp/_out/rainbow/Painter/GCC.Gui.Linuxfb/Painter.a: No such file or directory

There were errors. (0:00.14)

how fix it?

Subject: Re: Rainbow, first iteration
Posted by mirek on Thu, 01 Sep 2011 19:16:24 GMT
View Forum Message <> Reply to Message

rett wrote on Thu, 01 September 2011 03:54Hello!

I traying compile UWord in rainbow package.
params:
GUI LINUXFB
GCC Optimal

after build a have errors:
----- CtrlLib (GUI LINUXFB GCC LINUX POSIX) (1 / 8)
----- RichEdit (GUI LINUXFB GCC LINUX POSIX) (2 / 8)
----- PdfDraw (GUI LINUXFB GCC LINUX POSIX) (3 / 8)
----- plugin/jpg (GUI LINUXFB GCC LINUX POSIX) (4 / 8)
----- LinuxFb (GUI LINUXFB GCC LINUX POSIX) (5 / 8)
----- Framebuffer (GUI LINUXFB GCC LINUX POSIX) (6 / 8)
----- Painter (GUI LINUXFB GCC LINUX POSIX) (7 / 8)
----- UWord (GUI LINUXFB MAIN GCC LINUX POSIX) (8 / 8)
Linking...
c++: /home/rett/upp/_out/rainbow/CtrlLib/GCC.Gui.Linuxfb/CtrlLib.a: No such file or directory
c++: /home/rett/upp/_out/rainbow/RichEdit/GCC.Gui.Linuxfb/RichEdit.a: No such file or directory
c++: /home/rett/upp/_out/rainbow/PdfDraw/GCC.Gui.Linuxfb/PdfDraw.a: No such file or directory
c++: /home/rett/upp/_out/rainbow/plugin/jpg/GCC.Gui.Linuxfb/jpg.a: No such file or directory
c++: /home/rett/upp/_out/rainbow/Painter/GCC.Gui.Linuxfb/Painter.a: No such file or directory

There were errors. (0:00.14)

how fix it?

I bet the problem is that uppsrc is not in the assembly.

Your assembly "package nests: should look like:

u:\upp.src\rainbow;u:\upp.src\uppsrc;

(u:\upp.src is specific to me, but I hope you got the idea...)

Mirek

Subject: Re: Rainbow, first iteration
Posted by rett on Fri, 02 Sep 2011 07:49:22 GMT
View Forum Message <> Reply to Message

Thank You, it fixed my problem.