Subject: Ideas and priorities for the next development cycle
Posted by mirek on Sat, 04 Aug 2012 09:13:27 GMT
View Forum Message <> Reply to Message

Hi everybody,

with Skylark being finialized (just a couple of documentation topics missing now), I would like to start a debate about things to be done next (and estimates in [d]ays/[w]eeks/[m]onth).

My favorite topics:

Platforms

- "rainbowization" of Font metrics (looks like there will be more options in the future than Win32 and fontconfig)
- MacOSX support (with fudadmin excellent contribution, it should now be relatively easy) 2m
- iOS support (well, I am not much into it, but having MacOSX, it perhaps could be easy) 2m
- Android support 4m
- Create U++ rainbow version using Qt as host API (in the same matter as we are using e.g. X11). This one in fact would make U++ apps to run on many many platforms. 2m
- Create U++ rainbow version using Gtk as host API. This would have the advantage of better integration of U++ with Linux, e.g. we should be able to use Gtk standard dialog. 2m
- WinRT?

IDE

- speedup of IDE startup and operations (got quite slow recently because we have now a lot more documentation) 2w
- new set of icons 1w
- finally adding Docking capability 1m
- improving Assist++ (add some heuristic support of macros) 2m

Feel free to add more suggestions.

Also note this previous thread:

 http://www.ultimatepp.org/forum/index.php?t=msg&th=6289& amp;start=0&

I guess some of topics should be brought back..

Mirek

Subject: Re: Ideas and priorities for the next development cycle
Posted by Mindtraveller on Sat, 04 Aug 2012 19:28:12 GMT
View Forum Message <> Reply to Message

Mirek, if you need it, I'll try to contribute icons for U++ IDE. Please contact me on details if you wish to.

Subject: Re: Ideas and priorities for the next development cycle
Posted by mdelfede on Mon, 06 Aug 2012 21:30:14 GMT
View Forum Message <> Reply to Message

A great small improvement on qtf... ability to have image/objects anchored to pages and text flowing over/around them.
It's the only thing I miss on it.

Max

Subject: Re: Ideas and priorities for the next development cycle
Posted by lectus on Mon, 03 Sep 2012 16:26:59 GMT
View Forum Message <> Reply to Message

Android support is a must have.

This platform is growing too quickly to be ignored.

Subject: Re: Ideas and priorities for the next development cycle
Posted by mirek on Mon, 03 Sep 2012 17:19:58 GMT
View Forum Message <> Reply to Message

lectus wrote on Mon, 03 September 2012 12:26Android support is a must have.

This platform is growing too quickly to be ignored.

I agree, unfortunately I have some capacity limits...

Surely, with Rainbow, it should be easy for somebody else to develop Android support...

An alternative and perhaps promising way is to simply use Qt as rainbow backend (same way we are using e.g. Win32) and let it handle platform specific issues. Also note that this is not mutually exclusive

Mirek

Subject: Re: Ideas and priorities for the next development cycle

Posted by kohait00 on Tue, 04 Sep 2012 08:28:55 GMT
View Forum Message <> Reply to Message

@android support: we should know where we want to go with android?

a) blank canvas with upp typical look and feel (from w32 i.e.), single touch support... should be posible over the NDK

b) android typical appearance in terms of control widgets (way more complicated, implies to properly integrate moultitouch in upp Ctrl.

c) limiting usage of upp to Core stuff, using NDK bridging gap of java to c++. native widgets from android.. not a 'real' integration.. that said: is a 'real' integration possible anyway?

apreciate your ideas..
cheers

Subject: Re: Ideas and priorities for the next development cycle
Posted by mirek on Tue, 04 Sep 2012 11:19:58 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Tue, 04 September 2012 04:28@android support: we should know where we want to go with android?

a) blank canvas with upp typical look and feel (from w32 i.e.), single touch support... should be posible over the NDK

b) android typical appearance in terms of control widgets (way more complicated, implies to properly integrate moultitouch in upp Ctrl.

c) limiting usage of upp to Core stuff, using NDK bridging gap of java to c++. native widgets from android.. not a 'real' integration.. that said: is a 'real' integration possible anyway?

apreciate your ideas..
cheers

Well, a and (then) b are "standard approach".

Anyway, I have to say that recently I have started thinking about c) option, which basically is "wxWidget way". Interesting part then is what makes U++ compeling to me and what of that could be preserved when going "native GUI".

IMO, Core is simple and clear, esp. on android.

Anything that can be compiled without CtrlCore is preservable as well - that includes Draw. It should also be relatively easy to integrate Draw with any native GUI.

Of course, CtrlCore would have to go, CtrlLib would have to be replaced by something. The main issue at that point is how to make CtrlLib cross-platform. Another issue is what are important "U++ style" features of CtrlLib and whether they could be preserved. I believe that those factors are important:

- widgets are independent from GUI. I guess this part might be quite hard to achive, but perhaps not impossible. (Just to make this point clear, it for example means that I can take ArrayCtrl, fill in values, THEN add it to dialog, and THEN open the dialog in GUI. Or in fact, I can use ArrayCtrl and never use it in GUI, just to store and retrieve values). Related issue is that widgets are represented by object variables, not pointers to objects.

- frame system; we have no view class, but we have powerfull ScrollBar that can be added as Frame to any Ctrl. Hopefully could be preserved in some form, but most likely not for native views...

- customization using Display/Convert

- layouts, but those are not as important...

Hard to say, perhaps we might try something like that... Maybe even use something like wxWidgets, encapsulate U++ way and let THEM to deal with platform independency

---

Subject: Re: Ideas and priorities for the next development cycle
Posted by mirek on Tue, 04 Sep 2012 13:22:20 GMT
View Forum Message <> Reply to Message

Recommended reading:

https://groups.google.com/forum/?fromgroups=#!topic/wx-dev/Y GcUkAjE5Es

---

Subject: Re: Ideas and priorities for the next development cycle
Posted by Tom1 on Tue, 04 Sep 2012 19:59:47 GMT
View Forum Message <> Reply to Message

Hi Mirek,

While reading the link and your thoughts on platform support, I once again started to wonder where is the world of GUI headed and how U++ is going to cover that. In order to better understand how you see the future of U++ on various platforms (existing and new) I would be very interested to see a platform by platform listing how Draw, CtrlCore and CtrlLib are going to be implemented. I mean, which APIs are going to be used on each platform.

I'm not a great fan of leaning on pre-existing libraries, since this is the highway to bloated software, difficult dependencies and bugs. In my book U++ project has a clean record in interfacing directly to platform APIs and implementing clean and efficient code for any required task. Using external dependencies between native APIs and U++ seems like a diversion from the well established path we have seen so far.

Best regards,

Tom

Subject: Re: Ideas and priorities for the next development cycle
Posted by mirek on Wed, 05 Sep 2012 08:05:40 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 04 September 2012 15:59Hi Mirek,

While reading the link and your thoughts on platform support, I once again started to wonder where is the world of GUI headed and how U++ is going to cover that. In order to better understand how you see the future of U++ on various platforms (existing and new) I would be very interested to see a platform by platform listing how Draw, CtrlCore and CtrlLib are going to be implemented. I mean, which APIs are going to be used on each platform.

Status quo is "bussines as usual" - attempting to take the lovest possible level, using rainbow. Which does not preclude using e.g. gtk instead of plain X11 - in fact, using gtk as platform API would at least had advantage of better integration, e.g. we would be able to use gnome fileselector as an option.

"option c" was, at the moment, just an idea.

That said, supporting two relatively friendly platforms (Win32, Posix/X11) is costly. Supporting 5 (adding MacOSX, iOS, and unfriendly Android) might prove too much...

Mirek

Subject: Re: Ideas and priorities for the next development cycle
Posted by kohait00 on Wed, 05 Sep 2012 08:49:50 GMT
View Forum Message <> Reply to Message

i must agree with mirek.. i kinda think android is too much different to be fully supported in upp. in fact, any of such touch orientated OSs would be, because they impose their entirely new user philosophie (i.e. view stack, back buttons, very context orientated menus, etc..). to comply with this, upp would need a fairly large amount of redesign from buttom up (as far as i can grasp upp as a whole, correct me if i am wrong). and if ever done, wouldn't we criple upp too much with it? touch orientated progs are too different anyway. probably, upp would need to develop a whole new set of controls anyway.

OTOH, there is still a certain appeal in simple, unbloated, single touch orientated gui views (like with resistive touchscreens anyway), just on a very common multitouch platform, that for theese purposes simply hides it's presense (fullscreen view with rainbow)

so as a starter a) is a must anyway. from there on, we can see how far we can get.

what do you think?

Subject: Re: Ideas and priorities for the next development cycle
Posted by Mindtraveller on Fri, 07 Sep 2012 19:08:08 GMT
View Forum Message <> Reply to Message

I think it is good idea to avoid using CtrlCore/CtrlLib as something general enough to support different GUI paradigms such as desktop vs Android.

I think we do not need compatibility between classic and new (android-like also with iOS) interfaces. So we have to create different levels of abstraction for new interfaces above Rainbow (which is simply crossplatform GUI drawing interface, if I understand right). Going that way, we may create different GUI class hierarchy which supports tablet-like interfaces which require different controls, different events, different ideology.

Subject: Re: Ideas and priorities for the next development cycle
Posted by Tom1 on Sat, 08 Sep 2012 15:26:11 GMT
View Forum Message <> Reply to Message

Hi Pavel,

I think you've got a point there. However, it would still be beneficial to have the graphics rendering interface (Draw or rather something more advanced) compatible between the classic desktop and touch environments. Anyway, the split to desktop and touch would be something like this.

Desktop:

- Windows
- X11
- OSX
- (Wayland)

Touch:

- Android
- iOS
- Windows Metro

I'm personally not a great fan of touch interfaces, at least not yet, but I guess being prepared is not a bad policy.

Best regards,

Tom

Subject: Re: Ideas and priorities for the next development cycle
Posted by mirek on Sun, 09 Sep 2012 07:34:04 GMT
View Forum Message <> Reply to Message

One more thing about Android:

I am pretty sure that Firefox for Android is a C++ application.

More importantly, I am quite sure that Chrome is also C++.

Means it should be possible after all to achieve at least that level of integration... (But perhaps somebody more familiar with Android sources should check this...)

Mirek

Subject: Re: Ideas and priorities for the next development cycle
Posted by nlneilson on Mon, 10 Sep 2012 00:03:43 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Sat, 08 September 2012 08:26
I'm personally not a great fan of touch interfaces, ...

Same here! A small wireless mouse is what I prefer.

Porting between Python-Java-C++ is fairly easy once a person gets used to it.

Porting U++ to work with the Android OS shouldn't be that much of a problem if taken a step at a time.

Subject: Re: Ideas and priorities for the next development cycle
Posted by kohait00 on Mon, 10 Sep 2012 06:45:55 GMT

View Forum Message <> Reply to Message

[qoute]
I'm personally not a great fan of touch interfaces
[/quote]

they are probably not as precise enough as we are with the mouse.
nonetheless, touch interfaces will be the new holy grail for the next probably 5 to ten years at least (if they ever vanish at all)..so it's good to have the upp portability and ease of use in that area too..

we really might think of a new TouchCore and derived controls, to generate a whole set of ne controls. i think many developpers would prefer them in embedded systems to the current desktop like look, for a simplicity and information focused reason.

EDIT: mozilla sources

https://wiki.mozilla.org/Mobile/Fennec/Android

Subject: Re: Ideas and priorities for the next development cycle
Posted by drjo1952 on Mon, 10 Sep 2012 16:40:23 GMT

View Forum Message <> Reply to Message

Mirek,

Ultimate++ is always my first 'go-to' tool in creating new tools for our lab work and for our clients. My usual approach in solving new coding chores is to try out Examples or Bazaar using code that has some possible code snippets that could be used. For example, I need to tie audio and video feedback elements in my current project; so I first tried the few media examples in bazaar. I am currently using Ubuntu 12.04 and first tried the SDL example (SDLCtrl_demo); I could only get this example to work by downloading the SDL project tarball and ./configure, make, and make install - there were missing header files otherwise. The MediaPlayer (bazaar/Media) example however does not work because of this: libavutil/colorspace.h not found. This raises a fundamental issue of testing - should we perhaps have a formal suite of examples that are automatically exercised in Ubuntu and Windows (at least) to correct configuration and missing libraries? I think Ultimate++ is a first-class tool and I don't want newbies becoming tainted by examples not working. I am not an expert in the internal ultimate++ build mechanics to pull this (automatic test suite) off by myself without some guidance but I believe we could garner significant benefits by the effort. I would like to vote for an automatic testing program as a critical next-step activity.

drjo1952

Subject: Re: Ideas and priorities for the next development cycle
Posted by Novo on Mon, 10 Sep 2012 16:52:44 GMT

View Forum Message <> Reply to Message

drjo1952 wrote on Mon, 10 September 2012 12:40Mirek,

I would like to vote for an automatic testing program as a critical next-step activity.

drjo1952

Buildbot is good for both automated building and testing.

Subject: Re: Ideas and priorities for the next development cycle
Posted by mirek on Mon, 10 Sep 2012 17:33:05 GMT
View Forum Message <> Reply to Message

Novo wrote on Mon, 10 September 2012 12:52drjo1952 wrote on Mon, 10 September 2012 12:40Mirek,

I would like to vote for an automatic testing program as a critical next-step activity.

drjo1952

Buildbot is good for both automated building and testing.

uppsrc/BuildAll builds all of reference and examples..

Original intention was to run it before each release

I agree that some automated testing system would be nice. In fact, I guess this is where dolik's experimental website might be useful

---

Subject: Re: Ideas and priorities for the next development cycle
Posted by drjo1952 on Wed, 12 Sep 2012 13:13:37 GMT
View Forum Message <> Reply to Message

Mirek,

I have tangentially been following Dolik's universal makefile concepts - My suggestion is a simple holistic question about actually doing testing suites now with the tools at hand! I will take you suggestions of buildroot and your BuildAll as a starting point to push the effort along. Does BuildAll also build the bazaar examples as well? This raises another important question that was in the background with my original proposal? Is bazaar supposed to be a working reference point for Ultimate++; ie. should our Ultimate++ community take the responsibility for insuring that all the examples in bazaar work? If so, this implies that we are choosing which tangential applications and libraries Ultimate will integrate with; a choice that should not be taken likely. I understand that this is implying a commitment for contributors to keep bazaar examples up to date if they choose to put their code there. Perhaps we need two bazaars, one for 'supported code' and one for experimental code. The supported bazaar would then come under our test suites and we would fix or otherwise address (perhaps remove) issues found with testing.

drjo1952

Subject: Re: Ideas and priorities for the next development cycle
Posted by koldo on Wed, 12 Sep 2012 16:02:21 GMT
View Forum Message <> Reply to Message

Hello all

I agree with drjo1952. Perhaps Bazaar items would fit in original BuildAll or in a separate BuildAllBazaar.

In any case the Bazaar contributors that would want their packages there would have to prepare the packages to fit in (if necessary) and indicate what are the packages to include.

Subject: Re: Ideas and priorities for the next development cycle
Posted by Novo on Thu, 13 Sep 2012 03:21:58 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 10 September 2012 13:33Novo wrote on Mon, 10 September 2012 12:52drjo1952 wrote on Mon, 10 September 2012 12:40Mirek,

I would like to vote for an automatic testing program as a critical next-step activity.

drjo1952

Buildbot is good for both automated building and testing.

uppsrc/BuildAll builds all of reference and examples..

Original intention was to run it before each release

I agree that some automated testing system would be nice. In fact, I guess this is where dolik's experimental website might be useful

Well, this one-loop app is far-far away in capabilities from buildbot, which is quite powerfull and easy to set up.
IMHO it would be better to spend time improving U++ itself instead of reimplementing already well working apps.

Subject: Re: Ideas and priorities for the next development cycle
Posted by Novo on Thu, 13 Sep 2012 03:49:29 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 04 August 2012 05:13Hi everybody,

with Skylark being finialized (just a couple of documentation topics missing now), I would like to start a debate about things to be done next (and estimates in [d]ays/[w]eeks/[m]onth).

My favorite topics:

Platforms

Another crazy idea: hardware-accelerated version of rainbow similar to FireMonkey.

Cross-platform compatibility
Vector drawn interface elements
Any visual component can be a child of any other visual component allowing for creation of hybrid components
Built-in styling support
Support for visual effects (such as Glow, Inner Glow, Blur for example) and animation of visual components

Subject: Re: Ideas and priorities for the next development cycle
Posted by mirek on Thu, 13 Sep 2012 08:41:31 GMT
View Forum Message <> Reply to Message

Novo wrote on Wed, 12 September 2012 23:49
Another crazy idea: hardware-accelerated version of rainbow similar to FireMonkey.

Cross-platform compatibility
Vector drawn interface elements
Any visual component can be a child of any other visual component allowing for creation of hybrid components
Built-in styling support
Support for visual effects (such as Glow, Inner Glow, Blur for example) and animation of visual components

I think you do not need to coordinate this with U++ release cycle, it should be possible to develop idependently...

Subject: Re: Ideas and priorities for the next development cycle
Posted by malinowskia on Thu, 27 Sep 2012 12:22:01 GMT
View Forum Message <> Reply to Message

What about porting U++ to touchpoints (tablets, smartphones)

I know that main purpose of this software is to use in desktop application but new devices are trying to replace standard desktop apraoch and will do in near future.

F have informed that use HTML5 was a mistake and will switch to native app.

Arek

Subject: Re: Ideas and priorities for the next development cycle
Posted by kohait00 on Thu, 27 Sep 2012 13:28:42 GMT
View Forum Message <> Reply to Message

what about a nicer, more win32/gtk independent beatyfied upp ctrls look? could be a upp own fingerprint /look and feel for upp-based software..we dont need software to look like OS related

those fancy controls from the touch gadgets are stealing the show... a bit.

win 8 will have touch/multitouch, we have to prepare to use that

---

Subject: Re: Ideas and priorities for the next development cycle
Posted by nlneilson on Thu, 27 Sep 2012 14:04:24 GMT
View Forum Message <> Reply to Message

Make the "Find" box display "No more found"
and an option to start at the top, edit text to search, ...

Instead of having it just disappear.

Subject: Re: Ideas and priorities for the next development cycle
Posted by nlneilson on Sun, 11 Nov 2012 12:50:26 GMT
View Forum Message <> Reply to Message

With a desktop or notebook with a keyboard Ctrl+F is fast and easy to pull up the 'Find' box.

On a tablet with a smaller screen without a keyboard clicking to bring up an on screen keyboard or

Edit->Find and Replace->Find to paste in selected text or just search again from the top or even another file.

I don't recall using any other IDE, text editor, etc. where the find box just vanishes.

Some users are more mouse oriented when debugging.
Especially when different computer keyboards are different for a 12, 15 or 17 inch computers.
And for those that do not have the Hot-Keys memorized.

One option may be to have Ctrl+F toggle it Off-ON.
Maybe even in setup have the option to toggle or the way it is now.

Subject: Re: Ideas and priorities for the next development cycle
Posted by victor on Sat, 01 Dec 2012 19:22:05 GMT
View Forum Message <> Reply to Message

All sounds great!
What is the status of UPP-MacOSX ? Where could I get the last version?
thanks,
w

Subject: Re: Ideas and priorities for the next development cycle
Posted by Cocos2d-x on Mon, 03 Dec 2012 09:14:43 GMT
View Forum Message <> Reply to Message

Hello All,

I know U++ can not be an all encompassing solution for every category of development.

That being said, I saw that someone proposed a FireMonkey (or WPF) like GUI system. I would like to second that.

I truly believe that more control over GUI and in particular adding animation will be mandatory really soon.

I don't know the easyest path to get this or even if core U++ developers have an interest to let this happen.

U++ is really great for productivity and it would be nice if it could also be used in future for more up to date GUIs.

Now a proposal. A full vectorial GUI is maybe too hard to achieve, but something more dynamic using OpenGl is feasible.

I tried Cocos2d-X, which is a C++ port of Cocos2d.

Results are great and impressive, some GUI controls like sliders and potentiometers are available. For sure it is game oriented, but can be extended to be a full GUI. It is not vectorial, but highly multiplatform.

Cocos2D-x coding is kind of verbose. But if .LAY file could generate Cocos2d scenes it would be a true gift.

A mix of U++ and Cocos2d-X would give a productivity boost for game developpers and a animatable GUI to U++ developpers.

Sorry for this long post. But i'm really interested in it.

Best regards.

Subject: Re: Ideas and priorities for the next development cycle
Posted by nlneilson on Wed, 05 Dec 2012 01:27:16 GMT
View Forum Message <> Reply to Message

To stop the 'Find' box from disappearing this was done and maybe it will be useful to someone else when working on a tablet without a keyboard or get tired of using Ctrl+F.

Quote:To prevent Find box from vanishing:

C:\upp\uppsrc\CodeEditor\FindReplace.cpp
comment line 555 and add line with ;

compile C:\upp\uppsrc\ide in TheIDE
rename to theide.exe and paste into C:\upp

It will search to the bottom of a file then search from the top again. Other files can be searched also.

Subject: Re: Ideas and priorities for the next development cycle
Posted by nlneilson on Sat, 15 Dec 2012 15:50:54 GMT
View Forum Message <> Reply to Message

The above post worked OK for basics. When a Find-Replace in addition to the Find was opened there were problems.

Instead comment these lines

C:\upp\uppsrc\CodeEditor\CodeEditor.cpp(261):
C:\upp\uppsrc\CodeEditor\FindReplace.cpp(252):
C:\upp\uppsrc\RichEdit\Cursor.cpp(94):
C:\upp\uppsrc\RichEdit\Cursor.cpp(328):
C:\upp\uppsrc\RichEdit\Cursor.cpp(347):
C:\upp\uppsrc\RichEdit\Find.cpp(113):

I have not run into any problems yet.

edit: If the find box was not always on top and didn't jump to the bottom center it would be better, at least for some.

Subject: Re: Ideas and priorities for the next development cycle
Posted by Didier on Sat, 15 Dec 2012 16:23:49 GMT
View Forum Message <> Reply to Message

Hi all,

I have a very simple request/idea about the way you can use the help in theIde.

When editing some code, you can easily jump to the library source code, but when in the lib source all you can get is the help about a certain function OR edit the help for that lib (which is not what you want most of the time).

Idea:
When clicking on the 'blue square' a popup menu should appear. It would propose 2 options:
* edit the help
* view the help topic (and thus open the help viewer on the dedicated topic)

Subject: Re: Ideas and priorities for the next development cycle
Posted by nlneilson on Sat, 15 Dec 2012 16:34:57 GMT
View Forum Message <> Reply to Message

Or maybe like how it is done in Eclipse with a list of options or suggestions.

Subject: Re: Ideas and priorities for the next development cycle
Posted by lectus on Wed, 19 Dec 2012 14:47:51 GMT
View Forum Message <> Reply to Message

Android support is a must have, because it's the fastest growing mobile platform.

I believe with NDK you can use C++.

I think Mac OS X is also good to have, because it's part of the 3 big Desktop platforms: Windows, Linux and Mac OS X.

U++ is already full-featured as it is. It needs to concentrate on running on more platforms now (Skylark was a huge step in my opinion because web apps run everywhere if a browser is available).

Subject: Re: Ideas and priorities for the next development cycle
Posted by lectus on Wed, 19 Dec 2012 15:08:27 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 03 September 2012 13:19lectus wrote on Mon, 03 September 2012 12:26Android support is a must have.

This platform is growing too quickly to be ignored.

I agree, unfortunately I have some capacity limits...

Surely, with Rainbow, it should be easy for somebody else to develop Android support...

An alternative and perhaps promising way is to simply use Qt as rainbow backend (same way we are using e.g. Win32) and let it handle platform specific issues. Also note that this is not mutually exclusive

Mirek

There's now Necessitas (Qt for Android). So if you develop a Qt backend for U++ it will probably run on Android through Necessitas.