Subject: Mac OS X port architecture
Posted by daveremba on Sun, 17 Jul 2011 08:02:24 GMT
View Forum Message <> Reply to Message

Step A. Complete and test theide under X11 on MacOSX that is able to build itself.

Manual makefile exists now.
Need to try the makefile generation.
Building the examples have a visual problem:
flat buttons and garbage pixels under menus

PROPOSED:

Step B. Either use theide or Xcode on Mac to build up a Cocoa bridge into UPP framework.

n.b. Carbon lib with C++ direct linking still exists for 32 bit apps, but I think we want to support 64 bit arch, and this will require using Objective-C
and Cocoa.

For step B:

Here is how I think the Cocoa/MacOSX port would be architected, after I looked at wxWidgets and Firefox source:

1. For the Cocoa/MacOSX port, in CtrlCore, there will be new files like:

DrawCocoa.cpp
DrawWinCocoa.cpp
DrawTextCocoa.cpp
...

and
CocoaApp.cpp
CocoaProc.cpp
CocoaWnd.cpp
...

(as there is currently DrawX11... and DrawWin32... etc)

In particular: the top-level window is created through Cocoa, the App and Window are registered with MacOSX, events are captured from Cocoa and directed into UPP, drawing of specific buttons etc. are redirected out through Cocoa, font information obtained through Cocoa, etc.

2. Also in CtrlLib there will be new .m or .mm files for gcc to process; these are Objective-C source files that are part of the MacOS build that get linked into Theide. These files connect the DrawCocoa... files above into MacOSX Cocoa via the object messaging system.

n.b. It is likely that the new .cpp files will be small (or maybe not even needed) as most processing

could done inside the .m files. This is TBD.

The build step for #2 cannot be done on a Linux machine AFAIK, because there are Apple-specific headers and libs that are needed (hence 'legal').

n.b. Xcode doesn't seem to show the actual flags to gcc, but there should be a gcc command line equivalent that could be added to theide,
thus eliminating the need for Xcode to build the Objective-C and Cocoa parts. This is TBD.

3. The last step of generating an installable image is optional, and would give UPP a polished functionality so it installs like an .MSI or RPM; but is not essential. Step #3 not possible to generate on Linux AFAIK, because the .dmg disk
image file is MacOSX specific.

TBD: how to make installable disk image as a nightly build

4. These new files (in #1 and #2) will be switched on by a new flag, PLATFORM_MACOSX.

The existing flag PLATFORM_OSX11 will remain, and means to build a MacOSX app using the X11 emulator (this is the step A above).

n.b. Or PLATFORM_OSX11 could mean to use Cocoa, and the X11 version might be phased out, but I think it will have value for a while until the Cocoa port is completed, tested, accepted, etc.

Recommendations:

1. The Mac build of UPP does need to occur on a machine that runs true MacOSX (virtually or physically).

2. New files will need to be added to CtrlCore, as well as some minor patches to existing files.

note: this post was treated as a separate topic from a related post on the backend development process: architecture meaning what to do, process meaning how to do it.
But, I did mix both topics in this message.

-Dave
