Subject: best way to draw text/fonts on MacOS from C/C++
Posted by daveremba on Sun, 24 Jul 2011 06:37:09 GMT
View Forum Message <> Reply to Message

Apparently not all graphics calls need to be
done using Objective-C/C++ on Apple.

Here are some relevant calls for fonts, in C/C++:

CTFontCreateWithName
CTFontGetGlyphsForCharacters
CTFontGetAdvancesForGlyphs

then, to draw, one uses the Core Graphics library,
also callable in C/C++ (drawing the line writes the
text characters into the framebuffer):

CGContextSetTextMatrix
CGContextSetTextPosition
CGContextSetShouldAntialias
CTLineDraw

The CT... functions are part of CoreText.

The CG... functions are part of Quartz 2D API.

According to Apple, one is better off drawing text
not using Quartz, but instead the Core Text
functions:
"Quartz 2D provides a limited, low-level interface for
drawing text encoded in the MacRoman text encoding and for
drawing glyphs"
The code technique above is used in FLTK
(see below) and mixes Quartz (CG)/CT.

In a C or C++ file one can access CG and CT by including
the ApplicationServices header and library (and these
functions are directly callable from C/C++).

Regarding the Core Graphics calls, Apple says:
"You can obtain a graphics context by using Quartz graphics
context creation functions or by using higher-level functions
provided in the Carbon, Cocoa, or Printing frameworks. Quartz
provides creation functions for various flavors of Quartz
graphics contexts including bitmap images and PDF. The Cocoa
framework provides functions for obtaining window graphics
contexts. "

Modern x86_64 apps apparently need to use Cocoa
only for non-drawing functions, and to create the
graphics context and associate it with a window
(and this must be done in Objective-C/C++).
The .mm file and -framework Cocoa
needs to be passed into gcc, maybe as a post build
step in UPP. I will try this out.

n.b. Quartz 2D does not seem to have the MacOS functions for
kybd, mouse, etc. (I will check; maybe that was in Carbon).

So for UPP on MacOS: text/font drawing into a
non-X11 (native) window could be
implemented now with the existing C/C++ tools
(after the window is created).

--

This info I learned from examining the Fast Light
Toolkit (FLTK), which I've worked with and
enjoyed using. They've ported it to MacOSX.

I think FLTK gets the Mac OS port done in a simpler and smaller amount
of code than Firefox, wx, or Tk/tcl:

fl_cocoa.mm window, event, kybd, mouse, dnd code
Objective-C++ (3500 lines)
mac.H header mapping FLTK to mac functions
fl_font_mac.cxx font/text info & drawing code in C++

FLTK does not use any high-level widgets on any platform;
it works a lot more like UPP, and draws itself whatever
is needed from basic 2D elements: polylines/polygons,
images/pixmaps, and text/fonts. So, for these
reasons I recommend taking a look at it.

Dave