Subject: Re: Porting SystemDraw to Frambuffer
Posted by kohait00 on Wed, 16 Dec 2009 10:19:59 GMT
View Forum Message <> Reply to Message

Ok, people, this framebuffer topic is about to have a revival.
since we have the Painter stuff now and the headless drawing, and even the Docking facilities in bazaar, we almost have a "window manager".. but I need help

I have started to prepare some porting stubs for the use of framebuffer in the CtrlCore. Well, for now, ist really stubs, just moslty plain copy and paste from X11 code (which lies next), and commenting doubtable or unneeded/unneccessary parts away.

i realized that the underlying interface is quite "distributed" over a lot of sections, without explicit "tagging" what is what. it'd be esier to have the commen interface functions somehow grouped, so one could know, these functions are required. current situation is that its pretty mixed with the helper functions for the apropriate interface as well. both, in headers and code. so this is maybe to be reorganised, especially if we take in account, that this might not be the last porting we face (next important would be to port to apple graphics), and there will be even more i think. so we need to make things clearer here. (at least a short description on what so far is used by both interfaces WIN32/X11 and is required in any other port, and what are their interface specific helpers, and maybe a sentence of description to each, especially when it comes to Clipboard,DragandDrop and the like).

back to topic.

base idea:
using /dev/fb0 to display the one and only (no window manager) TopWindow instance of an application (naturally fullscreen), which is driven directly by user input extracted from /dev/input layer

class
SystemDraw : public BufferPainter

where the BufferPainter can be initialized with an ImageBuffer
which in turn should somehow represent the /dev/fb0.
-->problem: the fb0 can have really weired dimensions, bitwisths per color, strides and the like..is the ImageBuffer capable to cope with that? or do we need to have the double buffering as only solution (where the normal ImageBuffer 24bit is overlayed/calculated on tob of the weired fb0 layout)?

the control input section should be driven directly with /dev/input/, this normally does invoke a repaint in some sort

we need to provide a full implementation of own drag and drop, which can be significantly simpler i think, because we dont drag between multiple windows od Win32/X11, but simply inside the TopWindow (which is between Ctrls)

we need to provide a full replacement for Clipboard stuff, how does this work?

we need to provide a full replacement of the TimerThread (which is used heavily by Ctrls, it also needs to be reentrant (means a new occurance of timer event is handled either by interrupting a possibly currently still working timerthread procedure, or by spawning another thread to handle this, which leads either to application lock (due to repeated reentrance) or to thread explosion if handling callbacks are misdimensioned somehow.

The GUI thread (which is the main thread) would be far the simplest part i think, cause it creates, opens and runs the TopWindow and realizes all the drawing and user input.

is there any description (besides the code itself) available in terms of how Clip, Drag and Drop, the timer thread and the GUI thread are set up (basic ideas should be enough)

the GUI flag, which is used currently can be abandoned maybe, because it is evaluated only in POSIX environment, in WIN32 GUI is somewhat implicit, as soon as one uses the package CtrlCore, one uses gui stuff in Win32. in Linux, one should be able to specify GUI X11 or GUI FB somehow.. but this is small peanuts. backwards compatibility can be done imlicitly by defining a standard flag.

the stubs (NOT implementations) are half way done, as soon as it compiles and links, i will post it here, maybe mirek can take a look over it and clean out what he dooms or claims not neccessary (since i have no much idea of underlying gui facility) and maybe already integrate it in the trunk, to be able to code parallely. first thing woul be, of corse, to open the /dev/fb0 and display the main window once...

ok, till then, cheers

PS: maybe we can really use this effort to restruct code to make porting in future a lot esier.

PSS: sorry for the text chunk