Subject: Re: New age of GUI
Posted by piotr5 on Tue, 11 Jun 2013 20:34:14 GMT
View Forum Message <> Reply to Message

well, I did the mistake of speaking too abstractly. let's look at two examples:

suppose you want to create a slideshow-program. once upon a time these were programs where you press space or any key in order to move on to the next picture or animation. now with tablets however keyboard is no option. if a person has to put the screen-keyboard onto the display in order to press the key for showing next slide, s/he'll likely buy an ipad instead. similarily I have seen slideshow programs that used mouse-buttons for going forward and backward in the pictures. right mousebutton was go right and left button was for going back -- matching our western habit of reading from left to right. with tablets however users might prefer other input methods. a touchscreen doesn't have right button. some tablets have accelerometer though, so the user can knock on the device to move to the next pic. or maybe there's a remote control when the computer is attached to the tv set.

another program I have seen is a bit difficult to describe. its a sound-edit program. it is a modified version of a big sound-editing suite with many filters and such. of course a big sound-editing software is rather something for desktop. but this program is different, it's for improving sound-quality of some recorded speech. now port that to a tablet! having a menu in the usual way is not a good idea, better show some nice icons. icons however usually come with tooltips. however a touchscreen does not have the option to "hover" over it for some time, touching translates into a click on that icon. better would be if eye-movement would trigger the tooltips. or maybe use accelerometer to move around some additional cursor for that purpose. of course the user could plug in a trackball or something (although I have never seen any trackball specifically made for mobile devices). but using a mobile device means a friend gave you a recording he made, you receive it in the middle of the street and want to plug in the headphones and listen while you walk. if it's bad quality, maybe let this program run over it on autopilot. maybe some part of the recording was too silent, select it and turn up the volume of the selected section before you give the file to another person. now in that situation, the last thing you want to do is to pull out the mouse/trackball! if the program is not easy to control with the tablet without external devices, people will just try to concentrate a bit more deeply while listening, instead of using that program, maybe even turn up volume to a deafening level. maybe a user-interface alike to the one depicted in the first message of the thread would be better than reusing the gui of that desktop only program?

as for java and python, those offer the possibility to sort of compress the sources, remove comments and such, since text files take up too much space. this is what I call "compile into javacode". (a similar way of using the word "compile" are the various programming languages which get compiled into actual c-sources.) what I want is to use such things like metaprogramming with templates (boost) inside of java programs. one would assume translating from c++ to java should be quite easy, why are there no such programming-environments? of course you can write libs in c++ and compile them for the native architecture. however, if you distribute a program, and this program is using a c++ lib that wasn't installed in the os, you have a problem with maintaining a huge set of binaries, one for each platform that will appear in future. especially the testing is a nightmare. so unless the lib needs hardware access, make it so that lib is actually distributed as

java even though it's written in c++. it's related here because especially a java-port of u++ libs would be nice...
