Subject: Re: How does CtrlCore Image::Data::PaintImp work?
Posted by mirek on Tue, 25 Jan 2011 11:54:13 GMT
View Forum Message <> Reply to Message

fudadmin wrote on Mon, 24 January 2011 10:32sorry for my loose thoughts and sorry if some documentation does exist. If that's the case, please refer me to it. But I couldn't find anything useful.

I am trying to write ImageMac.cpp and I need some help.
As the topic name says I can't grasp and I need a general idea of "How does CtrlCore Image::Data::PaintImp work?"

1. Firstly, what are ResData?
my guesses (involves some optimisation - reusing?) :
 mouse pointer = cursor cheat, parts of the image being painted, or other resources like e.g brushes, masks, all of them combined (but how, in which cases?) ?

Draw is completely platform independent.

In some versions of Windows the number of image handles is limited to some thousands _per system_. It is thus advisable not to have all Images "realized" in GDI as bitmap handles.

So we keep track of number of handles in ResCount and if there is too much of them, we release some using Data...

Quote:
2. Why then ResCount>512 for X11?

Well, while there is no similar limitation for X11, as we already had mechanism available, it seemed to be a good idea to use it in X11 too, just to soften the load on system.

It is worth mentioning that U++ Image is somewhat unique and departing from "classic approach" where you usually have one class for 'client side' Image and another for 'server side'. Therefore we need to keep number of 'server side' images low, as we put them there automagically.

Quote:
And why for win32 - int max = IsWinNT() ? 250 : 100; ?
(see my comments below in the code.)

Because WinNT has unlimited number of handles, unlike Win98, but still, see the comment in 2.

Quote:
3. Of course, one of the main questions what would be the ResCount limit for Mac?

Hard to say, I would use 512 just like for X11. It is important to keep ResCount in sync though.

Having said all that, maybe all this approach is not the best one. Perhaps we should remove all "host platform hooks" from Image and simply provide a sort of cache only in SystemDraw, based on Image serial id.

I believe this is doable even now for MacOSX - all you need is to simply not use any of ResData, ResCount, SysRelease etc..

Quote:
for X11:
```
int  Image::Data::GetResCountImp() const
{
 return !!Sys().picture + !!Sys().picture8;
}
```

4. image converted to => picture + mask?

for win32:
```
int  Image::Data::GetResCountImp() const
{
 SystemData& sd = Sys();
 return !!sd.hbmp + !!sd.hmask + !!sd.himg;
}
```

5. image converted to => bitmap + mask + ??? ? (is this related somehow to DIB used somewhere? that's why the difference with X11?)

Well, these really are just counting how many handles are there for given Image.

Quote:
6. Also, what does
```
 Unlink();
 LinkAfter(ResData);
```
do?

Puts it to the front of the queue, so that least recently used Images are stripped of handles first (from the tail of the queue).

Quote:
7. How does ImageDraw::Section relate to Image::Data::PaintImp? Or what is ImageDraw::Section?

In no way, it is implementation detail for Win32.

Each platform should have its ImageDraw now. I guess that will have to change a bit with rainbow...