Subject: Font bug in X11
Posted by mrjt on Mon, 11 Jun 2007 13:03:34 GMT
View Forum Message <> Reply to Message

This is the simplest test-case I've been able to create:

```
#include <CtrlLib/CtrlLib.h>
using namespace Upp;

struct Test {
 Font font;
 Test() {font.FaceName("sans serif");}
};
Test t;

GUI_APP_MAIN
{
 printf("Hello World!");
}
```

This causes the app to stop at the following point in an X11 library:
Quote:XAddExtension () from /usr/X11R6/lib/libX11.so.6

This only happens when you set the font in this particular way (perhaps because GUI_APP_MAIN hasn't been run when the font face is set?).

I can understand if this problem is not fixable, but could there please be some sort of #error so it's easier to detect. It has taken me a lot of time to find out what caused this. Test on Slackware (KDE) with dev-3.

James

Subject: Re: Font bug in X11
Posted by mirek on Sat, 21 Jul 2007 13:30:04 GMT
View Forum Message <> Reply to Message

Sorry.... yes, anything related to GUI should not be placed in global variable.

Do you have any suggestion how to check this case?

Mirek

Subject: Re: Font bug in X11
Posted by mr_ped on Sat, 21 Jul 2007 22:52:17 GMT
View Forum Message <> Reply to Message

Straight solution is to have "is_inited" global variable for whole GUI thing and check it in every

function call (and set it up only after GUI_APP_MAIN is executed), but this will cause lot of overhead...

Maybe in debug mode only?

Another way is to move all GUI functions into "gui" object instance (UPPgui.font.FaceName("sans serif");), which can either be correctly initialized upon instantiation or the first the first instance may be stub with some ERROR() and then the object may be replaced after init with some real GUI object.
(again some overhead).

At compile time without overhead:
external tool after linking binary file to check ctor section to see what is called ahead of APP_MAIN, than scanning all sources for those ctor functions, emulating (interpretating them) and catching all GUI calls... .. oh well, this one would not really work.

Quote:
"It has taken me a lot of time to find out what caused this."

I'm sort of curious, why did it take so much time? Doesn't it crash in some nice way so you can see the stack and realize it was in FaceName, and ahead of APP_MAIN?