Subject: Zooming layouts and different behaviour windows/linux
Posted by mdelfede on Wed, 10 Aug 2011 17:18:21 GMT
View Forum Message <> Reply to Message

Hi,

As some of my customers complained about small fonts in windows version of my app, I started do dig a bit inside.
The worst stuff was a RichTextCtrl with a drawing inside, which got completely different scale.

Resuming, with NO changes in code, here the windows (from a full-screen running virtualizer, resolution 1440x900 as native one) :

You can also notice the picture upper right (a Picture control),
this one is made by aid of Painter + Drawing; internal sizes calculations are based on fixed font size, so the windows version get shrinked (font size 15, windows one 11 IIRC).

Adding this couple of lines makes it better :

```
#ifdef PLATFORM_WIN32
 SetStdFont(StdFont(15));
 Ctrl::SetZoomSize(Size(147, 21), Size(99, 13));
 InitRichTextZoom();
#endif
```

But IMHO it's really an ugly hack... but I had to do it because I was in hurry. It's taken from Linux values and code.
Btw, you can notice that tabs are NOT zoomed vertically, so the 2 editfields at top overlap with tab contents when zoomed, so I
believe it's a bug.... or not ?

Finally, the nice-looking linux version here; that one has fonts quite bigger than windows version, and I don't know why :

So, some questions :

1) Why are font sizes SO different from windows to linux ?
2) It's correct that tabs are not zoomed vertically ?
3) Applying this :

```
 Ctrl::SetZoomSize(GetStdFont().GetSize(), Size(99, 13));
```

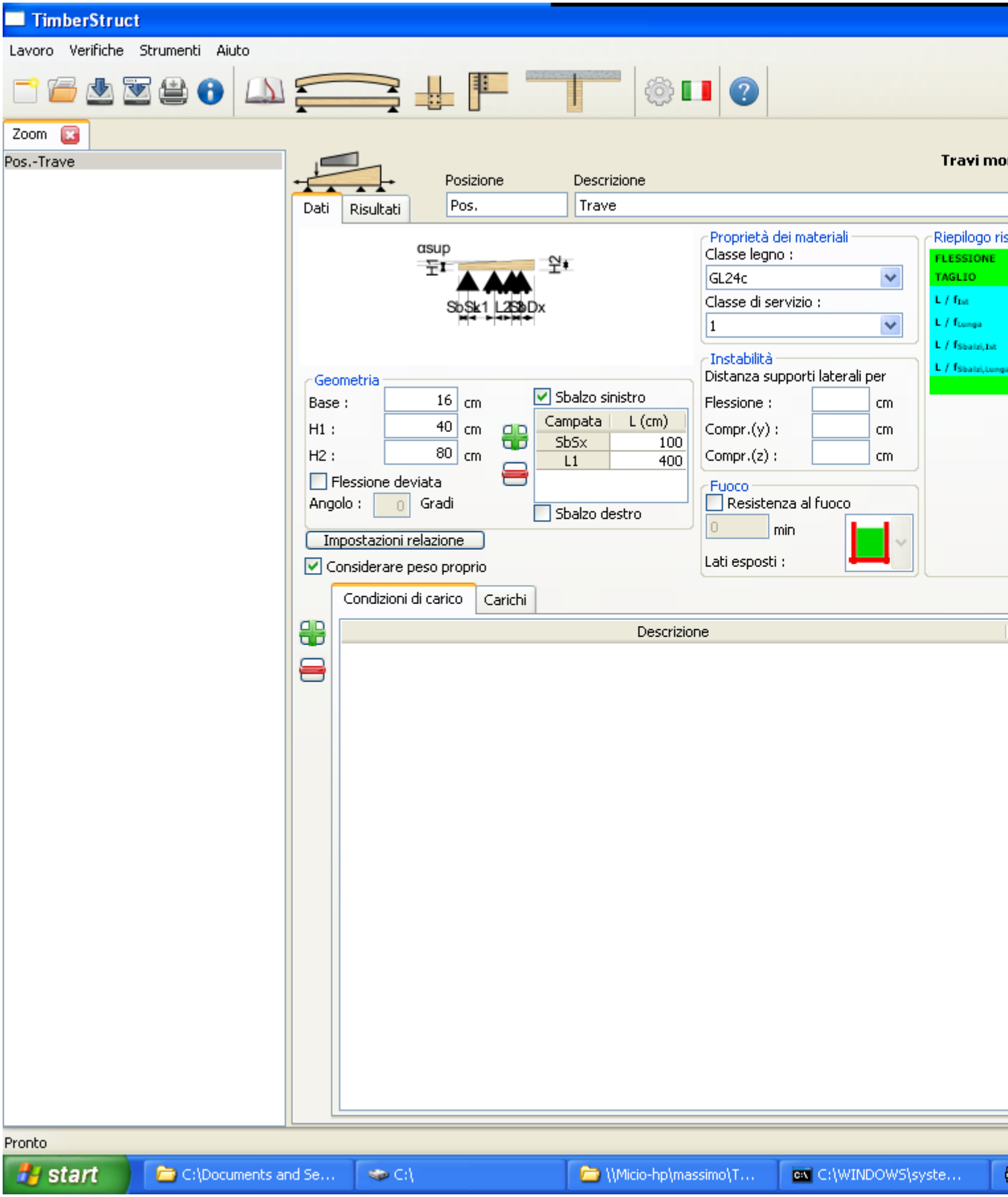
Sometimes the "growth" of elements don't match text one, so, for example, for some text sizes the buttons become too small to contain its label. Shouldn't they grow exactly (almost) as the text ?

Ciao

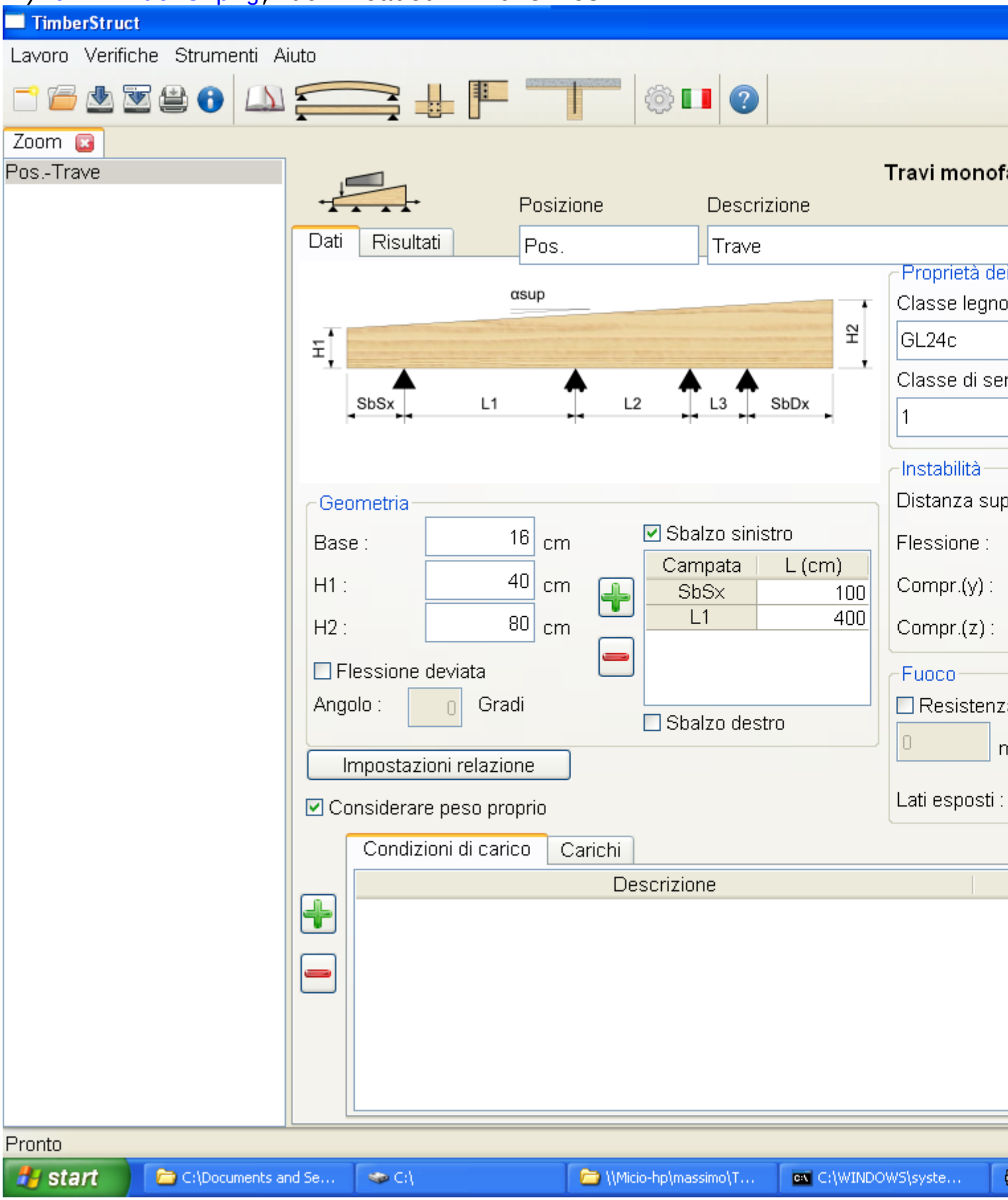
Max

File Attachments

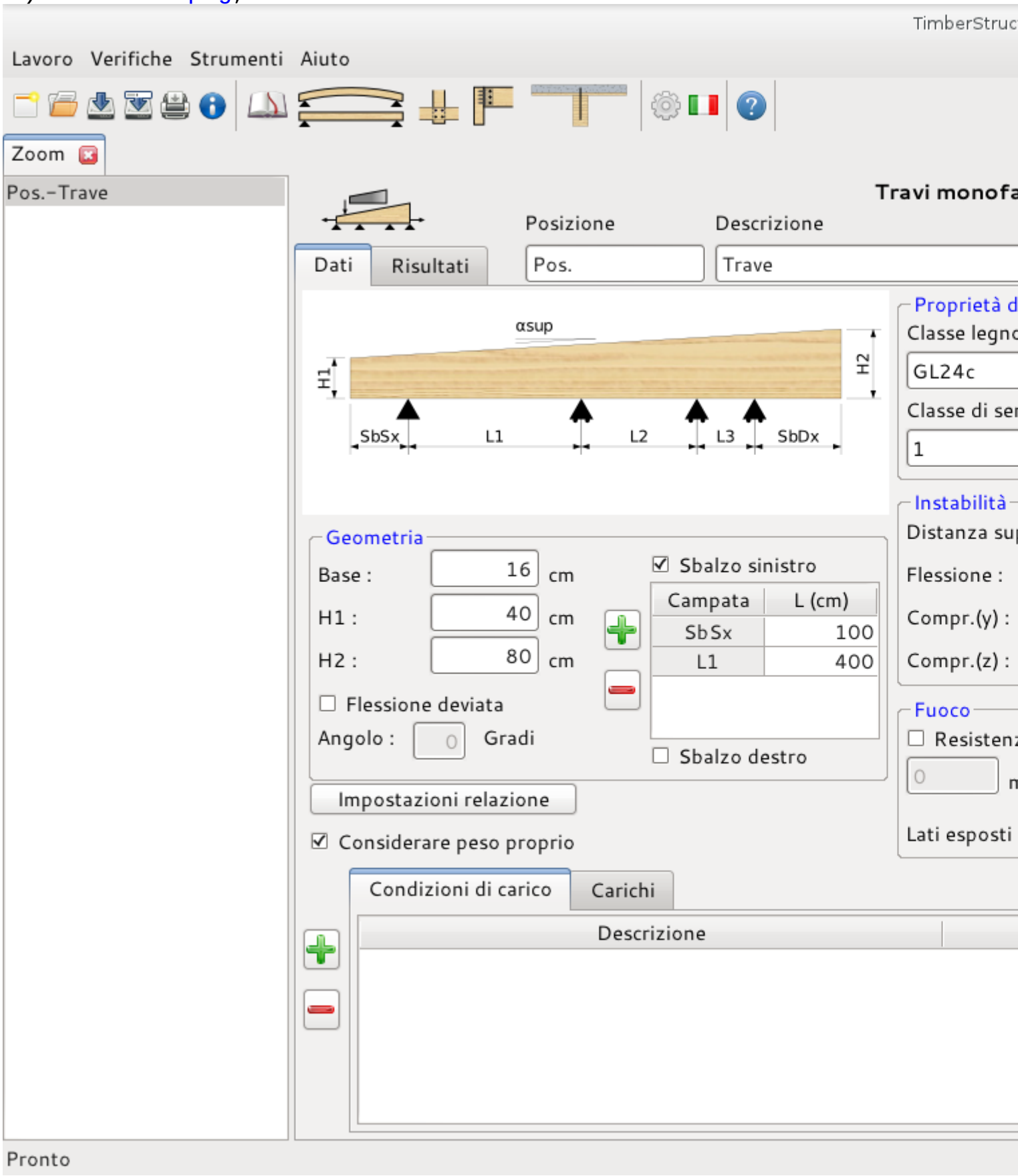
1) 01-windows.png, downloaded 1319 times


TimberStruct
Lavoro Verifiche Strumenti Aiuto
Zoom
Pos.-Trave
Travi mo
Posizione
Pos.
Descrizione
Trave
Dati
Risultati
asup
SbSk1L2SbDx
Proprietà dei materiali
Classe legno :
GL24c
Classe di servizio :
1
Riepilogo ris
FLESSIONE
TAGLIO
Geometria
Base : 16 cm
H1 : 40 cm
H2 : 80 cm
Flessione deviata
Angolo : 0 Gradi
Sbalzo sinistro
Campata L (cm)
SbSx 100
L1 400
Sbalzo destro
Instabilità
Distanza supporti laterali per
Flessione : cm
Compr.(y) : cm
Compr.(z) : cm
Fuoco
Resistenza al fuoco
0 min
Lati esposti :
Impostazioni relazione
Considerare peso proprio
Condizioni di carico
Carichi
Descrizione
Pronto
start
C:\Documents and Se...
C:\
\\Micio-hp\massimo\T...
C:\WINDOWS\syste...

2) 02-windows.png, downloaded 1416 times



3) 03-linux.png, downloaded 1414 times



Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mdelfede on Thu, 11 Aug 2011 08:30:04 GMT
View Forum Message <> Reply to Message

This couple of lines (another hacky stuff....) fix the tabs height problem. Height calculation is from trial-and-error, as I've not found any "intelligent" way to calculate it exactly :

```
TabCtrl::Style& tabCtrl_style = TabCtrl::StyleDefault().Write();
tabCtrl_style.tabheight = Ctrl::VertLayoutZoom(Font::GetStdFontSize().cy);
```

Thinking a bit more about my size problems, I got following conclusions :

1) Changing default font height *don't* rescale automatically the widgets.... why ? Of course, you can do it with another line of code, but imho it should to automatically.

2) Tabs needs some fixing to behave properly with zooming.
By now their size is based on font size, I guess default one; they don't take in account any scaling.

I think the best solution should be to fix point 1 (make zoom automatical with default font changes) and change automatically Tabs height depending on it.

Ciao

Max

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mirek on Fri, 12 Aug 2011 15:09:40 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Thu, 11 August 2011 04:30This couple of lines (another hacky stuff....) fix the tabs height problem. Height calculation is from trial-and-error, as I've not found any "intelligent" way to calculate it exactly :

```
TabCtrl::Style& tabCtrl_style = TabCtrl::StyleDefault().Write();
tabCtrl_style.tabheight = Ctrl::VertLayoutZoom(Font::GetStdFontSize().cy);
```

Thinking a bit more about my size problems, I got following conclusions :

1) Changing default font height *don't* rescale automatically the widgets.... why ? Of course, you can do it with another line of code, but imho it should to automatically.

2) Tabs needs some fixing to behave properly with zooming.

By now their size is based on font size, I guess default one; they don't take in account any scaling.

I think the best solution should be to fix point 1 (make zoom automatical with default font changes) and change automatically Tabs height depending on it.

Ciao

Max

Somewhat weird, as IMO it already depends on font height:

```
CH_STYLE(TabCtrl, Style, StyleDefault)
{
 font = StdFont();
 tabheight = font.Info().GetHeight() + 8;
 margin = 2;
```

One possible explanation is that layout is initialized before you change the font. Put sime DDUMPs/DLOGs to find out...

Mirek

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mdelfede on Fri, 12 Aug 2011 15:23:28 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 12 August 2011 17:09

Somewhat weird, as IMO it already depends on font height:

```
CH_STYLE(TabCtrl, Style, StyleDefault)
{
 font = StdFont();
 tabheight = font.Info().GetHeight() + 8;
 margin = 2;
```

One possible explanation is that layout is initialized before you change the font. Put sime DDUMPs/DLOGs to find out...

Mirek

Yep, it's initialized *before* for sure.

But, here we've got 2 different problems :

1) Calculated font and widgets sizes are quite different from Linux and windows. On windows we get small chars. DPI is set on 96 on my machine, so it's the normal one
I think it would be nicer to have similar font sizes on both, otherwise it's quite hard to have an uniform layout on both environment.

2) There should be a consistent way to resize fonts AND widgets on the fly, or at least at application startup... just after GUI_APP_MAIN.
Now, if I change font size there, layout don't change. If I zoom layout manually, layout changes, but tabs vertical sizes don't.

Max

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mirek on Fri, 12 Aug 2011 15:28:51 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Fri, 12 August 2011 11:23mirek wrote on Fri, 12 August 2011 17:09

Somewhat weird, as IMO it already depends on font height:

```
CH_STYLE(TabCtrl, Style, StyleDefault)
{
 font = StdFont();
 tabheight = font.Info().GetHeight() + 8;
 margin = 2;
```

One possible explanation is that layout is initialized before you change the font. Put sime DDUMPs/DLOGs to find out...

Mirek

Yep, it's initialized *before* for sure.
But, here we've got 2 different problems :

1) Calculated font and widgets sizes are quite different from Linux and windows. On windows we get small chars. DPI is set on 96 on my machine, so it's the normal one
I think it would be nicer to have similar font sizes on both, otherwise it's quite hard to have an uniform layout on both environment.

2) There should be a consistent way to resize fonts AND widgets on the fly, or at least at application startup... just after GUI_APP_MAIN.
Now, if I change font size there, layout don't change. If I zoom layout manually, layout changes,

but tabs vertical sizes don't.

Max

IMO, font size should basically follow the font size of host machine - that is what is chameleon for...

Now, it is unfortunately true that chameleon picks values only at beginning. However, if you would want to change it during program run, you would have to have some way how to set new values everywhere value is present. E.g. we have something like AttrText with SetFont method that is supposed to fill ArrayCtrl - you would have to refill ArrayCtrl if settings change. That IMO would require a lot of work for client code....

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mdelfede on Fri, 12 Aug 2011 15:34:00 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 12 August 2011 17:28

IMO, font size should basically follow the font size of host machine - that is what is chameleon for...

Now, it is unfortunately true that chameleon picks values only at beginning. However, if you would want to change it during program run, you would have to have some way how to set new values everywhere value is present. E.g. we have something like AttrText with SetFont method that is supposed to fill ArrayCtrl - you would have to refill ArrayCtrl if settings change. That IMO would require a lot of work for client code....

It's true, but it doesn't seem to me that fonts are *so* different.... windows are smaller, of course, but the difference spotted by Upp is huge.

Anyways, it should be a way to adjust fonts/layouts sizes *before* chamaleon grab them... otherwise, resizing fonts is useless.
What about some way to call an user function that do the job before Upp does its settings ? maybe with a #define which, when set, calls an user function, or something like this....

Max

Subject: Re: Zooming layouts and different behaviour windows/linux

Posted by mirek on Sat, 13 Aug 2011 01:12:31 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Fri, 12 August 2011 11:34mirek wrote on Fri, 12 August 2011 17:28

IMO, font size should basically follow the font size of host machine - that is what is chameleon for...

Now, it is unfortunately true that chameleon picks values only at beginning. However, if you would want to change it during program run, you would have to have some way how to set new values everywhere value is present. E.g. we have something like AttrText with SetFont method that is supposed to fill ArrayCtrl - you would have to refill ArrayCtrl if settings change. That IMO would require a lot of work for client code....

It's true, but it doesn't seem to me that fonts are *so* different.... windows are smaller, of course, but the difference spotted by Upp is huge.

If the font size differs from one that is present on host machine, well, that is another issue, one worth fixing. I have already pointed out where the font size is fetched from gtk...

Quote:
Anyways, it should be a way to adjust fonts/layouts sizes *before* chamaleon grab them... otherwise, resizing fonts is useless.
What about some way to call an user function that do the job before Upp does its settings ? maybe with a #define which, when set, calls an user function, or something like this....

Putting it to the beginning of GUI_APP_MAIN should definitely work. Please put breakpoint to e.g. TabCtrl chameleon init to find out where it gets called from... (and then tell me).

Mirek

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mdelfede on Sun, 14 Aug 2011 00:13:22 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 13 August 2011 03:12
If the font size differs from one that is present on host machine, well, that is another issue, one worth fixing. I have already pointed out where the font size is fetched from gtk...

ops, I've missed that post.

Quote:
Putting it to the beginning of GUI_APP_MAIN should definitely work. Please put breakpoint to e.g. TabCtrl chameleon init to find out where it gets called from... (and then tell me).

It doesn't, at least, on my app if I change font size just after GUI_APP_MAIN i does resize fonts, but not layout.
As you can see in my second posted picture, I've got to resize "manually" the layout with the ugly SetZoomSize(), but tabs don't get resized because Chamaleon calculates its size before.
I had to change manually "tabheigh" to make it better.

I'll do some check on next days and I'll report you.

Ciao

Max

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mdelfede on Sun, 14 Aug 2011 15:50:49 GMT
View Forum Message <> Reply to Message

mhhh... I tried in Linux, putting a breakpoint here :

```
CH_STYLE(TabCtrl, Style, StyleDefault)
{
 font = StdFont();
 tabheight = font.Info().GetHeight() + 8;
 margin = 2;
 sel = Rect(2, 2, 2, 2);
```

and one just at the beginning of GUI_APP_MAIN.
The CH_STYLE is hit BEFORE, so tabs get wrong height.
Here the backtrace after debug hit :

```
Upp::TabCtrl__Style__StyleDefault::Init (this=0x222b2c0) at
/home/massimo/sources/upp-svn/uppsrc/CtrlLib/TabCtrl.cpp:7
Upp::TabCtrl::StyleDefault () at /home/massimo/sources/upp-svn/uppsrc/CtrlLib/TabCtrl.cpp:5
Upp::ChHostSkin () at /home/massimo/sources/upp-svn/uppsrc/CtrlLib/ChGtk.cpp:483
Upp::Ctrl::ChSync () at /home/massimo/sources/upp-svn/uppsrc/CtrlCore/Ctrl.cpp:908
Upp::Ctrl::InitX11 (display=0x0) at
/home/massimo/sources/upp-svn/uppsrc/CtrlCore/X11App.cpp:412
main (argc=1, argv=0x7fffffffe698, envptr=0x7fffffffe6a8) at
```

/home/massimo/sources/upp-svn/TimberStruct/TimberStruct/TimberStruct.cpp:727

ciao

Max

p.s.: my ide hangs on windows when I run under debugger with breakpoint set. Is my problem or a but ? Using MSC9 and related sdk.

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mirek on Mon, 15 Aug 2011 13:28:38 GMT
View Forum Message <> Reply to Message

OK, I have tried to make chameleon detect changes made to the font later, so perhaps it will now work...

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mdelfede on Wed, 17 Aug 2011 11:43:58 GMT
View Forum Message <> Reply to Message

Yep it works now... it's enough to change font height on GUI_APP_MAIN and all layouts gets resized.

Thank you Mirek !

Max

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by Sender Ghost on Wed, 17 Aug 2011 17:12:02 GMT
View Forum Message <> Reply to Message

Hello, Mirek and Massimo.

mirek wrote on Mon, 15 August 2011 15:28
OK, I have tried to make chameleon detect changes made to the font later, so perhaps it will now work...

It is not necessary, in my opinion, at least for this case. What it really adds is some overhead to check font of each Ctrl in constructor, which noticable for applications with many Ctrls, even for TheIDE.

mdelfede wrote on Wed, 17 August 2011 13:43
Yep it works now... it's enough to change font height on GUI_APP_MAIN and all layouts gets resized.

With changes, which Mirek added, it is possible to change default font size for Ctrls inside GUI_APP_MAIN, as you said. You could achieve the same with INITBLOCK, without such changes before:

```
#include <CtrlLib/CtrlLib.h>

using namespace Upp;

INITBLOCK {
 SetStdFont(StdFont(20));
}

class App : public TopWindow {
public:
 typedef App CLASSNAME;
 App();

 TabCtrl tabs;
 EditString text;
};

App::App()
{
 Title("TabCtrl size test");
 CenterScreen().Sizeable().MinimizeBox().MaximizeBox();
 SetRect(Size(640, 480));

 Add(tabs.VSizePosZ(4, 4).HSizePosZ(4, 4));
 Add(text.RightPosZ(4, 100).TopPosZ(4, 19));

 tabs.Add("First"); tabs.Add("Second"); tabs.Add("Third");
 text.SetData("Some text");
}

GUI_APP_MAIN
{
 App app;
 app.Run();
}
```

I suggest you, Massimo, to provide full testcase with your problem, not just screenshots.
I see from screenshots, that different fonts and font sizes used for Windows and Linux environments, which configurable.

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mdelfede on Wed, 17 Aug 2011 23:34:02 GMT
View Forum Message <> Reply to Message

Well, I've not seen how much overhead is added, but I guess that a check on a boolean before control creation (which I guess it's all needed to do the job...) is no noticeable overhead.

Your solution, with INITBLOCK, could work too.... but how do you know that INITBLOCK comes *before* chamaleon init ? And, worse,
how do you know your INITBLOCK comes *after* gui init, X11 in particular, without which we could have a crash, I guess ?

I already had bad experiences with INITBLOCKS expecting something already initialized which it wasn't.....

Ciao

Max

edit : about the testcase, IMHO is not needed.
I know we've got different font sizes in windows and linux, and every gui app can show it. What I see is that it seems to me that fonts in Linux are quite big, it seems to me even bigger than default gui fonts, but it's maybe just an illusion....

What I've seen is that my app got too different layout sizes between linux and windows, and I needed a more uniform aspect;
that's more important than keeping default gui font size in this case, so I needed a way to change it at startup.

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by Sender Ghost on Thu, 18 Aug 2011 06:24:27 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Thu, 18 August 2011 01:34
Well, I've not seen how much overhead is added, but I guess that a check on a boolean before control creation (which I guess it's all needed to do the job...) is no noticeable overhead.

Well, may be on Linux it is not very noticable (Ctrl constructor -> SyncCh -> GetStdFont -> InitStdFont).

At first it just felt differently, after starting TheIDE, compiled with new changes. Then I created some (unrealistic) testing application:

```
#include <CtrlLib/CtrlLib.h>

using namespace Upp;
```

```
GUI_APP_MAIN
{
 Array<EditString> list;
 TimeStop e;

 const int n = 1000000;
 for (int i=0; i < n; ++i)
  list.Add();

 PromptOK(AsString(e.Elapsed()));
}
```

The results for changed source code:
4406
For not changed:
4266

Agree, not much, just 3% difference.

mdelfede wrote on Thu, 18 August 2011 01:34
Your solution, with INITBLOCK, could work too.... but how do you know that INITBLOCK comes *before* chamaleon init ? And, worse,
how do you know your INITBLOCK comes *after* gui init, X11 in particular, without which we could have a crash, I guess ?

I already had bad experiences with INITBLOCKS expecting something already initialized which it wasn't.....

As about INITBLOCK:
"Upp provides a nice solution to do init / deinit work of your package's static or global stuff (i.e. if it's not Plain Ol' Data and needs some function calls).
If your INITBLOCK / EXITBLOCK resides in a .cpp file and the file contains code that is actually referenced (used) somewhere else, everything works as expected, no precautions need to be taken. If not, the linker will drop the file, your init code won't be invoked.
This is because the INITBLOCK / EXITBLOCK actually registers itself in an init facility from upper Upp code. So no code ref downwards is added."

Briefly, it depends from building order, where other INITBLOCKs (residing inside *.icpp files) linked before main code.

mdelfede wrote on Thu, 18 August 2011 01:34
What I've seen is that my app got too different layout sizes between linux and windows, and I needed a more uniform aspect;
that's more important than keeping default gui font size in this case, so I needed a way to change it at startup.

I think, this is because different DPI used for Linux and Windows environments by default. For

example, I already increased DPI value on Windows for TFT monitor, but it feels much bigger for CRT monitor. Also the font sizes by default: 8 for Windows, 10 for Linux environment, such as GNOME.

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mdelfede on Thu, 18 Aug 2011 12:59:40 GMT
View Forum Message <> Reply to Message

Sender Ghost wrote on Thu, 18 August 2011 08:24

Agree, not much, just 3% difference.

Inded.... and the advantage is much bigger....

Quote:

"Upp provides a nice solution to do init / deinit work of your package's static or global stuff (i.e. if it's not Plain Ol' Data and needs some function calls).....

.....Briefly, it depends from building order, where other INITBLOCKs (residing inside *.icpp files) linked before main code.

The problem here is *not* if INITBLOCK gets linked; if put in main code, it'll be linked for sure.
The problem is INITBLOCK random order; you can't assume *any* stuff being initialized before it.
So, you can't assume that X11 initialization has been done

Quote:
I think, this is because different DPI used for Linux and Windows
.....

Yep, it's for sure... But I can't ask my customers to change Dpi

Ciao

Max

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by Sender Ghost on Thu, 18 Aug 2011 14:56:44 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Thu, 18 August 2011 14:59

Yep, it's for sure... But I can't ask my customers to change Dpi

There is no problem with DPI, if you like to see small font sizes by default on TFT monitor with different pixel size, than CRT, on Windows. Moreover, this is not related for this case, but it changes layout sizes for U++ application.

The question is: "Do you need this overhead (bigger or smaller), when you didn't change fonts?".

What I really sure, that I will do differently to get normal font sizes in application. It can be problem with RichText, but there is no testcase to fix this for this particular application.

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by Sender Ghost on Fri, 19 Aug 2011 08:00:53 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 15 August 2011 15:28OK, I have tried to make chameleon detect changes made to the font later, so perhaps it will now work...
Related to latest changes, I think, there is more flexible way. Even there is example on how to do it, e.g. from Ctrl::InitX11 function. What we need is just public Ctrl::ChSync function, instead of private. Then it will be possible to apply font changes not only for concrete Ctrl, but for all of them:

```
GUI_APP_MAIN
{
 SetStdFont(StdFont(20));
 Ctrl::ChSync();
}
```

and this without unneeded overhead in Ctrl constructor, when you didn't change fonts.

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mirek on Fri, 26 Aug 2011 12:02:56 GMT
View Forum Message <> Reply to Message

Sender Ghost wrote on Thu, 18 August 2011 10:56mdelfede wrote on Thu, 18 August 2011 14:59
Yep, it's for sure... But I can't ask my customers to change Dpi

There is no problem with DPI, if you like to see small font sizes by default on TFT monitor with different pixel size, than CRT, on Windows. Moreover, this is not related for this case, but it changes layout sizes for U++ application.

The question is: "Do you need this overhead (bigger or smaller), when you didn't change fonts?".

I agree, I will optimize.

Mirek

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mirek on Sat, 27 Aug 2011 00:30:03 GMT
View Forum Message <> Reply to Message

I have tried

```
#include <CtrlLib/CtrlLib.h>

using namespace Upp;

GUI_APP_MAIN
{
 Array<EditString> list;
 RTIMING("chsync");
 const int n = 1000000;
 for (int i=0; i < n; ++i)
  list.Add();
}
```

and in release optimal mode, I cannot detect any difference at all...

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by Sender Ghost on Sat, 27 Aug 2011 08:53:14 GMT
View Forum Message <> Reply to Message

I have tried your code with RTIMING on Windows XP with different compilers (TDM GCC v4.5.2 and C/C++ compiler from Microsoft Visual Studio 2008 (MSC9)) for Optimal build mode, with following results (for 5 invocations, after and before changes):

TDM GCC v4.5.2:

after changes:
```
TIMING chsync       :  4.54 s  -  4.54 s  ( 4.54 s  / 1 ), min:  4.54 s , max:  4.54 s , nesting: 1 - 1
TIMING chsync       :  4.53 s  -  4.53 s  ( 4.53 s  / 1 ), min:  4.53 s , max:  4.53 s , nesting: 1 - 1
TIMING chsync       :  4.52 s  -  4.52 s  ( 4.53 s  / 1 ), min:  4.53 s , max:  4.53 s , nesting: 1 - 1
TIMING chsync       :  4.55 s  -  4.55 s  ( 4.55 s  / 1 ), min:  4.55 s , max:  4.55 s , nesting: 1 - 1
TIMING chsync       :  4.55 s  -  4.55 s  ( 4.55 s  / 1 ), min:  4.55 s , max:  4.55 s , nesting: 1 - 1
```

before changes:
```
TIMING chsync       :  4.45 s  -  4.45 s  ( 4.45 s  / 1 ), min:  4.45 s , max:  4.45 s , nesting: 1 - 1
```

```
TIMING chsync      :  4.45 s  -  4.45 s  ( 4.45 s  / 1 ), min:  4.45 s , max:  4.45 s , nesting: 1 - 1
TIMING chsync      :  4.46 s  -  4.46 s  ( 4.46 s  / 1 ), min:  4.46 s , max:  4.46 s , nesting: 1 - 1
TIMING chsync      :  4.45 s  -  4.45 s  ( 4.45 s  / 1 ), min:  4.45 s , max:  4.45 s , nesting: 1 - 1
TIMING chsync      :  4.44 s  -  4.44 s  ( 4.45 s  / 1 ), min:  4.45 s , max:  4.45 s , nesting: 1 - 1
```

MSC9:

after changes:

```
TIMING chsync      :  2.98 s  -  2.98 s  ( 2.98 s  / 1 ), min:  2.98 s , max:  2.98 s , nesting: 1 - 1
TIMING chsync      :  2.99 s  -  2.99 s  ( 2.99 s  / 1 ), min:  2.99 s , max:  2.99 s , nesting: 1 - 1
TIMING chsync      :  2.99 s  -  2.99 s  ( 2.99 s  / 1 ), min:  2.99 s , max:  2.99 s , nesting: 1 - 1
TIMING chsync      :  2.99 s  -  2.99 s  ( 2.99 s  / 1 ), min:  2.99 s , max:  2.99 s , nesting: 1 - 1
TIMING chsync      :  2.99 s  -  2.99 s  ( 2.99 s  / 1 ), min:  2.99 s , max:  2.99 s , nesting: 1 - 1
```

before changes:

```
TIMING chsync      :  2.92 s  -  2.92 s  ( 2.92 s  / 1 ), min:  2.92 s , max:  2.92 s , nesting: 1 - 1
TIMING chsync      :  2.94 s  -  2.94 s  ( 2.94 s  / 1 ), min:  2.94 s , max:  2.94 s , nesting: 1 - 1
TIMING chsync      :  2.93 s  -  2.93 s  ( 2.94 s  / 1 ), min:  2.94 s , max:  2.94 s , nesting: 1 - 1
TIMING chsync      :  2.95 s  -  2.95 s  ( 2.95 s  / 1 ), min:  2.95 s , max:  2.95 s , nesting: 1 - 1
TIMING chsync      :  2.93 s  -  2.93 s  ( 2.93 s  / 1 ), min:  2.93 s , max:  2.93 s , nesting: 1 - 1
```

As I already said, the differences are minor (about 2-3%) in this case. I just suggested to apply font changes "at once", when needed, instead of current approach (in constructor, for each Ctrl). This is "semi automatic" method with its (dis)advantages (compared to current "automatic" method), but implementation could be different.

In conclusion, because current approach exists and differences of changes are minor, there is no need "to fix the fix". On the other hand, it was informative.

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mirek on Sun, 28 Aug 2011 07:00:14 GMT
View Forum Message <> Reply to Message

Sender Ghost wrote on Sat, 27 August 2011 04:53
As I already said, the differences are minor (about 2-3%) in this case. I just suggested to apply font changes "at once", when needed, instead of current approach (in constructor, for each Ctrl). This is "semi automatic" method with its (dis)advantages (compared to current "automatic" method), but implementation could be different.

Well, before completely dismissing this issue, one possibility would be to place SyncCh somewhere else than Ctrl constructor.

The only problem is that I do not know where.. Promissing place might be TopWindow constructor, but that still is not really complete solution...

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mdelfede on Fri, 02 Sep 2011 13:26:02 GMT
View Forum Message <> Reply to Message

mhhhh.... what about putting it inside SetStdFont() ?

Max

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mirek on Fri, 02 Sep 2011 15:51:44 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Fri, 02 September 2011 09:26mhhhh.... what about putting it inside SetStdFont() ?

Max

SetStdFont knows nothing about CtrlCore. And, besides, you might want to alter more parameters than standard font, would not be nice if each of them issued ChSync (or maybe it would, I guess I have to check that, I mean to check how difficult ChSync is..).

Mirek

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mirek on Tue, 06 Sep 2011 12:09:37 GMT
View Forum Message <> Reply to Message

Sender Ghost wrote on Sat, 27 August 2011 04:53I have tried your code with RTIMING on Windows XP with different compilers (TDM GCC v4.5.2 and C/C++ compiler from Microsoft Visual Studio 2008 (MSC9)) for Optimal build mode, with following results (for 5 invocations, 1

I have optimized it a bit, would you find some time to measure again, it would be helpful...

Mirek

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by Sender Ghost on Tue, 06 Sep 2011 17:45:22 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 06 September 2011 14:09

I have optimized it a bit, would you find some time to measure again, it would be helpful...

Following results for 3762 and 3829 revisions:

TDM GCC v4.5.2:

after changes:
```
TIMING chsync      :  4.47 s  -  4.47 s  ( 4.47 s  / 1 ), min:  4.47 s , max:  4.47 s , nesting: 1 - 1
TIMING chsync      :  4.49 s  -  4.49 s  ( 4.49 s  / 1 ), min:  4.49 s , max:  4.49 s , nesting: 1 - 1
TIMING chsync      :  4.47 s  -  4.47 s  ( 4.47 s  / 1 ), min:  4.47 s , max:  4.47 s , nesting: 1 - 1
TIMING chsync      :  4.47 s  -  4.47 s  ( 4.47 s  / 1 ), min:  4.47 s , max:  4.47 s , nesting: 1 - 1
TIMING chsync      :  4.47 s  -  4.47 s  ( 4.47 s  / 1 ), min:  4.47 s , max:  4.47 s , nesting: 1 - 1
```

before changes:
```
TIMING chsync      :  4.43 s  -  4.43 s  ( 4.43 s  / 1 ), min:  4.43 s , max:  4.43 s , nesting: 1 - 1
TIMING chsync      :  4.45 s  -  4.45 s  ( 4.45 s  / 1 ), min:  4.45 s , max:  4.45 s , nesting: 1 - 1
TIMING chsync      :  4.44 s  -  4.44 s  ( 4.44 s  / 1 ), min:  4.44 s , max:  4.44 s , nesting: 1 - 1
TIMING chsync      :  4.43 s  -  4.43 s  ( 4.43 s  / 1 ), min:  4.43 s , max:  4.43 s , nesting: 1 - 1
TIMING chsync      :  4.43 s  -  4.43 s  ( 4.43 s  / 1 ), min:  4.43 s , max:  4.43 s , nesting: 1 - 1
```

MSC9:

after changes:
```
TIMING chsync      :  2.96 s  -  2.96 s  ( 2.96 s  / 1 ), min:  2.96 s , max:  2.96 s , nesting: 1 - 1
TIMING chsync      :  2.95 s  -  2.95 s  ( 2.95 s  / 1 ), min:  2.95 s , max:  2.95 s , nesting: 1 - 1
TIMING chsync      :  2.95 s  -  2.95 s  ( 2.95 s  / 1 ), min:  2.95 s , max:  2.95 s , nesting: 1 - 1
TIMING chsync      :  2.95 s  -  2.95 s  ( 2.95 s  / 1 ), min:  2.95 s , max:  2.95 s , nesting: 1 - 1
TIMING chsync      :  2.96 s  -  2.96 s  ( 2.96 s  / 1 ), min:  2.96 s , max:  2.96 s , nesting: 1 - 1
```

before changes:
```
TIMING chsync      :  2.92 s  -  2.92 s  ( 2.92 s  / 1 ), min:  2.92 s , max:  2.92 s , nesting: 1 - 1
TIMING chsync      :  2.90 s  -  2.90 s  ( 2.90 s  / 1 ), min:  2.90 s , max:  2.90 s , nesting: 1 - 1
TIMING chsync      :  2.91 s  -  2.91 s  ( 2.91 s  / 1 ), min:  2.91 s , max:  2.91 s , nesting: 1 - 1
TIMING chsync      :  2.89 s  -  2.89 s  ( 2.89 s  / 1 ), min:  2.89 s , max:  2.89 s , nesting: 1 - 1
TIMING chsync      :  2.90 s  -  2.90 s  ( 2.90 s  / 1 ), min:  2.90 s , max:  2.90 s , nesting: 1 - 1
```

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mirek on Thu, 08 Sep 2011 07:56:42 GMT
View Forum Message <> Reply to Message

Thanks. Looks better now; I guess I can live with this...

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mdelfede on Sun, 24 Jun 2012 10:46:47 GMT

View Forum Message <> Reply to Message

Well... I hate this, but I must revive this thread

I have this in my code :

```
SetStdFont(StdFont(globalSettings().GetGraphicSettings().guiCharSize));
InitRichTextZoom();
```

which reads the gui std font size from a global variable and sets it. I can't do it on very beginning of GUI_APP_MAIN, but I do it indeed BEFORE creating the application main window.
But... the code does nothing. The gui font size don't change at all.

Any hint ? I'd really need a configurable gui font size, many customers asked for it.

Ciao

Max

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mirek on Sun, 24 Jun 2012 11:04:11 GMT
View Forum Message <> Reply to Message

I have just tried this:

```
GUI_APP_MAIN
{
 SetStdFont(StdFont(30));
 PromptOK("OK");
}
```

- works as expected... To dig deeper, I would need some more detailed testcase, I am afraid...

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mdelfede on Sun, 24 Jun 2012 11:31:12 GMT
View Forum Message <> Reply to Message

That's weird... on a simple test case, it works.
I'll look at it deeper...

Max

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mdelfede on Sun, 24 Jun 2012 12:00:56 GMT
View Forum Message <> Reply to Message

Well.... I'm really out of ideas on WHY, but it started working, at least partially... Maybe it was an old config file which I deleted somewhere.

Well, the 'almost' is because I'd like to do it in real time (I mean, application running), and the layout are not resynced.... Texts change size, but objects positions and sizes don't change up to next app restart.
Is there some function to force layouting ?
I mean, globally.... something like

GetTopWindow()->ReLayout();

Max

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mdelfede on Sun, 24 Jun 2012 12:36:54 GMT
View Forum Message <> Reply to Message

Well... here a simple testcase which shows a couple of interesting things :

TestLayoutZoom.lay :

```
LAYOUT(TestLayoutZoomLayout, 200, 100)
 ITEM(EditString, test2, LeftPosZ(112, 64).TopPosZ(40, 19))
 ITEM(EditIntSpin, charSize, LeftPosZ(112, 64).TopPosZ(16, 19))
 ITEM(Label, dv___2, SetLabel(t_("Label2 :")).LeftPosZ(4, 104).TopPosZ(44, 16))
 ITEM(Label, dv___3, SetLabel(t_("Label1 :")).LeftPosZ(4, 104).TopPosZ(20, 16))
END_LAYOUT
```

main.cpp :

```
#include <CtrlLib/CtrlLib.h>

using namespace Upp;
```

```
#define LAYOUTFILE <TestLayoutZoom/TestLayoutZoom.lay>
#include <CtrlCore/lay.h>

class TestLayoutZoom : public WithTestLayoutZoomLayout<TopWindow>
{
 void sizeEditCb(void);
 public:
  typedef TestLayoutZoom CLASSNAME;

  TestLayoutZoom();
};

void TestLayoutZoom::sizeEditCb(void)
{
 SetStdFont(StdFont(~charSize));
}

TestLayoutZoom::TestLayoutZoom()
{
 CtrlLayout(*this, "Window title");
 charSize <<= 15;
 charSize <<= THISBACK(sizeEditCb);
}

GUI_APP_MAIN
{
 TestLayoutZoom().Run();
}
```

On FIRST editspin change, I see just the label sizes which change; on next editspin changes, nothing more happens.
The layout is unchanged anyways.

Expected behaviour : layout and its elements should follow edispin value.

Besides of layoyut don't being updated, it seems that calls to SetStdFont() following first one are ignored.

Max

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mirek on Sun, 24 Jun 2012 13:09:18 GMT
View Forum Message <> Reply to Message

mdelfede wrote on Sun, 24 June 2012 08:00Well.... I'm really out of ideas on WHY, but it started working, at least partially... Maybe it was an old config file which I deleted somewhere.

Well, the 'almost' is because I'd like to do it in real time (I mean, application running), and the layout are not resynced.... Texts change size, but objects positions and sizes don't change up to next app restart.
Is there some function to force layouting ?

No. Standard font is, first of all, considered to be system setting that does not change during the application run. This is similiar to skin changes of host OS.

I am afraid that the only practical way is to somehow restart the application.

Mirek

Subject: Re: Zooming layouts and different behaviour windows/linux
Posted by mdelfede on Sun, 24 Jun 2012 13:13:34 GMT
View Forum Message <> Reply to Message

Well... it's a pity, but I guess I can live with it.

I found the reason of former problem, I was calling SetStdFont on app startup once, and it was ignoring following calls.
So, I guess I've to do it just once on app startup.

Thank you for answering

Max
