Subject: Re: howto best Ctrl Refresh handling w/ MT & very frequent refreshes
Posted by dolik.rce on Fri, 11 Jun 2010 14:06:41 GMT
View Forum Message <> Reply to Message

kohait00 wrote on Fri, 11 June 2010 14:11the sample from dolik is interesting, it intercepts the data change stuff, nice idea. tnhanks.

the only drawback is, that there are several controls where you dont only have GetData/SetData which triggers the refresh, like Label or Static or sth.. SetText(), SetLabel()...

it'd be great to have kind of a more general way of disabling refresh in entire gui for some time..

Hi Kohait,

I admit I didn't think about that. One usually expects Static widgets to be, uhm, kind of static

Anyway, if you are changing those as well, you can use template specialization. E.g. for label:#include <CtrlLib/CtrlLib.h>
using namespace Upp;

```
template <class T>
class Cached : public T{
 Value val;
 bool refreshflag;
public:
 void operator<<=(Callback action) {T::operator<<=(action);}
 void operator<<=(Value data)      {refreshflag=true; val=data;}
 void SetData(Value data)          {refreshflag=true; val=data;}
 Value GetData()                   {return val;}
 Value operator~()                 {return val;}
 void Apply()                      {if(refreshflag) T::SetData(val);}
 bool IsChanged()                  {return refreshflag;}
};
template <>
class Cached<Label> : public Label{
 Value val;
 bool refreshflag;
public:
 void operator<<=(Value data)      {refreshflag=true; val=data;}
 void SetData(Value data)          {refreshflag=true; val=data;}
 Value GetData()                   {return val;}
 Value operator~()                 {return val;}
 void Apply()                      {if(refreshflag) Label::SetLabel(AsString(val));}
 bool IsChanged()                  {return refreshflag;}
};

class guitest : public TopWindow {
public:
```

```
 typedef guitest CLASSNAME;
 Cached<EditIntSpin> s;  Cached<Label> l;
 guitest(){
  s.SetRect(10,10,50,24); l.SetRect(10,40,100,54);
  Add(s);   Add(l);
  s<<=0;   l<<="Nothing yet";
  s.Apply();  l.Apply();
 }
 void LeftDown(Point p,dword keyflags){
  s<<=int(~s)+1;  l<<=String("Last value: ")+IntStr(s.EditIntSpin::GetData());
 }
 void RightDown(Point p,dword keyflags){
  s.Apply();  l.Apply();
 }
};

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

This will unify the interface between usual widgets and Label. Of course you could also make a different template that would keep the SetLabel interface and just added the caching capabilities. That is up to your choice

BTW: I guess you don't need it, but I quite like the possibility to access the actual value displayed by the control, regardless on what is in cache. So I put it into the example code as a little bonus

Best regards,
Honza
