
Subject: Re: Switch skin

Posted by [galious](#) on Fri, 18 Apr 2014 19:42:43 GMT

[View Forum Message](#) <> [Reply to Message](#)

Sorry for being late with the patch. Work, personal activities and some deep thoughts about backwards compatibility are to be blamed...

Nevertheless I've created the following class OptionCh(ameleon) which should be a 100% drop-in replacement of the Option class.

OptionCh.h

```
#ifndef _CtrlLibExt_OptionCh_h_
#define _CtrlLibExt_OptionCh_h_

class OptionCh : public Option {
public:
    virtual void Paint(Draw& draw);

public:
    struct Style : ChStyle<Style> {
        Value look[3][4];
        Size size;
        Color textcolor[3][4];
        Value facecolor;
        Font font;
    };

public:
    enum { UNCHECKED = 0, CHECKED = 1, UNDETERMINED = 2 };

protected:
    virtual void RefreshPush();
    const Style *St() const;

private:
    const Style *style;
    const Style *styleSwitch;

public:
    static const Style& StyleDefault();
    static const Style& StyleSwitch();
    OptionCh& SetStyle(const Style& s);
    OptionCh& SetSwitchStyle(const Style& s);
};

#endif
```

OptionCh.cpp

```
#include "CtrlLibExt.h"

namespace CtrlLibExt
{

CH_STYLE(OptionCh, Style, StyleDefault)
{
    CtrlImageLook(look[OptionCh::UNCHECKED], CtrlImg::I_O0);
    CtrlImageLook(look[OptionCh::CHECKED], CtrlImg::I_O1);
    CtrlImageLook(look[OptionCh::UNDETERMINED], CtrlImg::I_O2);
    size = Null;

    textcolor[OptionCh::UNCHECKED][CTRL_NORMAL] =
    textcolor[OptionCh::CHECKED][CTRL_NORMAL] =
    textcolor[OptionCh::UNDETERMINED][CTRL_NORMAL] =
    textcolor[OptionCh::UNCHECKED][CTRL_HOT] =
    textcolor[OptionCh::CHECKED][CTRL_HOT] =
    textcolor[OptionCh::UNDETERMINED][CTRL_HOT] =
    textcolor[OptionCh::UNCHECKED][CTRL_PRESSED] =
    textcolor[OptionCh::CHECKED][CTRL_PRESSED] =
    textcolor[OptionCh::UNDETERMINED][CTRL_PRESSED] = SColorLabel();

    textcolor[OptionCh::UNCHECKED][CTRL_DISABLED] =
    textcolor[OptionCh::CHECKED][CTRL_DISABLED] =
    textcolor[OptionCh::UNDETERMINED][CTRL_DISABLED] = SColorDisabled();

    facecolor = SColorFace();
    font = StdFont();
}

CH_STYLE(OptionCh, Style, StyleSwitch)
{
    CtrlImageLook(look[OptionCh::UNCHECKED], CtrlImg::I_S0);
    CtrlImageLook(look[OptionCh::CHECKED], CtrlImg::I_S1);
    size = Null;

    textcolor[OptionCh::UNCHECKED][CTRL_NORMAL] =
    textcolor[OptionCh::CHECKED][CTRL_NORMAL] =
    textcolor[OptionCh::UNCHECKED][CTRL_HOT] =
    textcolor[OptionCh::CHECKED][CTRL_HOT] =
    textcolor[OptionCh::UNCHECKED][CTRL_PRESSED] =
    textcolor[OptionCh::CHECKED][CTRL_PRESSED] = SColorLabel();

    textcolor[OptionCh::UNCHECKED][CTRL_DISABLED] =
    textcolor[OptionCh::CHECKED][CTRL_DISABLED] = SColorDisabled();
}
```

```

facecolor = SColorFace();
font = StdFont();
}

```

```

OptionCh& OptionCh::SetStyle(const OptionCh::Style& s)
{
    if(style != &s) {
        style = &s;
        RefreshLayout();
        Refresh();
    }
    return *this;
}

```

```

OptionCh& OptionCh::SetSwitchStyle(const OptionCh::Style& s)
{
    if(styleSwitch != &s) {
        styleSwitch = &s;
        RefreshLayout();
        Refresh();
    }
    return *this;
}

```

```

void OptionCh::RefreshPush() {
    Pusher::RefreshPush();
}

```

```

const OptionCh::Style *OptionCh::St() const
{
    const Style *st;
    if(switchimage) {
        st = styleSwitch ? styleSwitch : &StyleSwitch();
    } else {
        st = style ? style : &StyleDefault();
    }

    return st;
}

```

```

void OptionCh::Paint(Draw& w) {
    const OptionCh::Style *st = St();
    Size sz = GetSize();

    if(!IsTransparent())
        ChPaint(w, 0, 0, sz.cx, sz.cy, st->facecolor);
}

```

```

Size isz = st->size;
if(!IsNull(isz) && st->look[0][0].Is<Image>()) {
    isz = st->look[0][0].To<Image>().GetSize();
}
if(!IsNull(isz)) {
    int h = GetSmartTextHeight("M", INT_MAX);
    isz = Size(h, h);
}

Size tsz(0, 0);
int ix = 0, iy = 0, ty = 0;

if(showlabel) {
    tsz = GetSmartTextSize(label, st->font);
    ty = (sz.cy - tsz.cy) / 2;
    iy = (tsz.cy - isz.cy) / 2 + ty;
} else {
    ix = (sz.cx - isz.cx) / 2;
    iy = (sz.cy - isz.cy) / 2;
}

int q = GetVisualState();
int r = !switchimage && (!notnull || threestate) && IsNull(option)
    ? OptionCh::UNDETERMINED
    : option == 1 ? OptionCh::CHECKED
    : OptionCh::UNCHECKED;

ChPaint(w, ix, iy, isz.cx, isz.cy, st->look[r][q]);

if(showlabel) {
    bool ds = !IsShowEnabled();
    DrawSmartText(w, isz.cx + 4, ty, tsz.cx, label, st->font,
        st->textcolor[r][q],
        VisibleAccessKeys() ? accesskey : 0);
    if(HasFocus())
        DrawFocus(w, RectC(isz.cx + 2, ty - 1, tsz.cx + 3, tsz.cy + 2) & sz);
}
}
}

```

CtrlLibExt.h

```

#ifndef _CtrlLibExt_CtrlLibExt_h
#define _CtrlLibExt_CtrlLibExt_h

#include <CtrlLib/CtrlLib.h>

```

```
using namespace Upp;

namespace CtrlLibExt
{

#include "OptionCh.h"
// After cleaning these up these will be published as well
#include "RichTextScroll.h"
#include "Splash.h"

} // namespace CtrlLibExt

#endif
```

Hopefully this can serve as a solid base to include skinning to the default Option control.

Best regards,
--Martin
