Subject: Re: styling of widgets ( animation / look and feel)
Posted by dodobar on Fri, 21 Apr 2023 09:40:54 GMT
View Forum Message <> Reply to Message

Thanks,
Yes I did not like the widgets having to pass back information to the style but I suppose 2 brains are better than one :roll:
baring any dependency issues the solution is nice and simple approach, I assume this is what you mean.

```
/**
@brief Structure representing a style in the GUI toolkit.
This structure encapsulates the visual and animation properties of a style
for use in the GUI toolkit. It includes values for look, colors, font, images,
and animation properties such as duration and easing function. By using this
structure, the look and feel of various widgets in the toolkit can be easily
customized and animated.
*/
template <class T>
struct ChStyle {
    // ...
    std::vector<T> animationStates;
    double animationDuration;
    EasingFunction easingFunction;
    Ctrl* widget; // Pointer to the widget this style is applied to
    // ...
};

struct Style : ChStyle<Style> {
    Value look[4];
    Color monocolor[4], textcolor[4];
    Point pressoffset;
    int   focusmargin;
    int   overpaint;
    Font  font;
    Image ok, cancel, exit;
    bool  transparent;
    bool  focus_use_ok;

    // Animation properties
    enum AnimationFinishBehavior {
        GOTO_FIRST,      // Go to the first frame when the animation is interrupted
        GOTO_LAST,       // Go to the last frame when the animation is interrupted
        COMPLETE,        // Complete the animation normally when interrupted
        COMPLETE_FAST,   // Complete the animation quickly when interrupted
        HOLD,            // Hold the current frame when the animation is interrupted
        REVERSE         // Reverse the animation when interrupted
    };

    AnimationFinishBehavior animationFinishBehavior;
    double animation_duration; // Duration of the animation in seconds
    double (*animation_easing_function)(double); // Easing function for the animation
};
```

One potential issue with adding the widget pointer to the Style struct is that it may not provide enough control or understanding for some widgets. For example, if a widget wants to animate a specific area of its display that is not accounted for in the Style struct, it may not be able to do so easily.
at some level though the designer will need to set up the Style how he wants the animation/design.

looking at the base ChStyle, I'm not sure why you are using three specific images (naming) in a style "Image ok, cancel, exit;"
I can imagine you could have a MainImage being the master image/Icon used and an additional SubImage potentially covering the case of an icon with another icon representing a menu drop-down or something and if you really feel like it a userImage .
Some additional notes:
might be good to have an additional colour to represent border and text background.
also Gradents (how is this handled)

```
/**
@brief Structure representing a style in the GUI toolkit.
This structure encapsulates the visual and animation properties of a style
for use in the GUI toolkit. It includes values for look, colors, font, images,
and animation properties such as duration and easing function. By using this
structure, the look and feel of various widgets in the toolkit can be easily
customized and animated.
*/
struct Style : ChStyle<Style> {
    Value look[4];
    //Color monocolor[4], textcolor[4];
    Point pressoffset;
    int   focusmargin;
    int   overpaint;
    Font  font;
    //Image ok, cancel, exit;
    bool  transparent;
    bool  focus_use_ok;

    // Adjusted properties for images and colors (not sure if you need the array of 4 this might be for
states)
    Image MainImage, SubImage, UserImage;
    Color textcolor[4], textBackground[4];
    Color bordercolour[4], foregroundcolor[4], altforegroundcolor[4]
backgroundcolor[4],altbackgroundcolor[4];
    //EDIT: monocolor should just be desaturated of the main and not needed
    // Animation properties
    enum AnimationFinishBehavior {
        GOTO_FIRST,      // Go to the first frame when the animation is interrupted
        GOTO_LAST,       // Go to the last frame when the animation is interrupted
        COMPLETE,        // Complete the animation normally when interrupted
        COMPLETE_FAST,   // Complete the animation quickly when interrupted
        HOLD,            // Hold the current frame when the animation is interrupted
        REVERSE          // Reverse the animation when interrupted
    };

    AnimationFinishBehavior animationFinishBehavior;
    double animation_duration; // Duration of the animation in seconds
    double (*animation_easing_function)(double); // Easing function for the animation
};
```