Subject: Re: styling of widgets (animation / look and feel)
Posted by dodobar on Thu, 20 Apr 2023 17:17:57 GMT
View Forum Message <> Reply to Message

Thanks, however the base classes I provided could still be relevant with a few modifications
To be more in line with what you're suggesting , possibly functions passed to it as the easing

this also would allow for custom using functions I've included a couple of generics.
(but you get the idea)

```
typedef double (*EasingFunction)(double);

struct ChDuration : ChStyle<ChDuration> { int value; };
#define CH_DURATION(name, init) CH_VAR(ChDuration, int, name, init)

struct ChEasing : ChStyle<ChEasing> { EasingFunction value; };

#define CH_EASING(name, init) CH_VAR(ChEasing, EasingFunction, name, init)
double LinearEasing(double t) {
    return t;
}

double EaseInQuad(double t) {
    return t * t;
}

double EaseOutQuad(double t) {
    return t * (2 - t);
}

double EaseInOutQuad(double t) {
    return t < 0.5 ? 2 * t * t : -1 + (4 - 2 * t) * t;
}

double EaseInCubic(double t) {
    return t * t * t;
}

double EaseOutCubic(double t) {
    double f = t - 1;
    return f * f * f + 1;
}

double EaseInOutCubic(double t) {
    return t < 0.5 ? 4 * t * t * t : (t - 1) * (2 * t - 2) * (2 * t - 2) + 1;
}
```

```
double EaseInExpo(double t) {
    return (t == 0) ? 0 : pow(2, 10 * (t - 1));
}

double EaseOutExpo(double t) {
    return (t == 1) ? 1 : 1 - pow(2, -10 * t);
}

double EaseInOutExpo(double t) {
    if (t == 0) return 0;
    if (t == 1) return 1;
    if ((t *= 2) < 1) return 0.5 * pow(2, 10 * (t - 1));
    return 0.5 * (-pow(2, -10 * --t) + 2);
}
```

example usage?

```
CH_DURATION(ButtonAnimationDuration, 300) // Define animation duration for button, e.g., 300ms
CH_EASING(ButtonEasingFunction, EaseInOutQuad) // Define easing function for button animation
```

Im guessing at this point but I assume will need new function definitions to include the new animation parameters:
perhaps a blank function so existing code is not effected (EasingFunction NoEasing)

```
void ChLookFn(Value (*fn)(Draw& w, const Rect& r, const Value& look, int lookop, Color ink, int duration, EasingFunction easing=NoEasing ?? ));
```

Perhaps even defining custom functions?

```
Value MyCustomLookFunction(Draw& w, const Rect& r, const Value& look, int lookop, Color ink, int duration, EasingFunction easing);
```

then example usage ? possibly? (Again these are my estimations is to the real code.)

```
ChLookFn(MyCustomLookFunction, ButtonAnimationDuration(), ButtonEasingFunction());
```

The Base class would look a little different taking the easing functions and providing a simpler set of functions to process animatable style components ?

```
class BaseAnimation {
public:
    BaseAnimation(int duration, EasingFunction easing)
        : duration_(duration), easing_(easing) {}

    double GetProgress(double t) const {
        return easing_(t / duration_);
    }

    Color LerpColor(const Color& start, const Color& end, double progress) const {
        int red = start.GetR() + progress * (end.GetR() - start.GetR());
        int green = start.GetG() + progress * (end.GetG() - start.GetG());
        int blue = start.GetB() + progress * (end.GetB() - start.GetB());
        int alpha = start.GetA() + progress * (end.GetA() - start.GetA());
        return Color(red, green, blue, alpha);
    }

    Point LerpPosition(const Point& start, const Point& end, double progress) const {
        int x = start.x + progress * (end.x - start.x);
        int y = start.y + progress * (end.y - start.y);
        return Point(x, y);
    }

    Size LerpSize(const Size& start, const Size& end, double progress) const {
        int cx = start.cx + progress * (end.cx - start.cx);
        int cy = start.cy + progress * (end.cy - start.cy);
        return Size(cx, cy);
    }

    float LerpFloat(float start, float end, double progress) const {
        return start + progress * (end - start);
    }

    double LerpScale(double start, double end, double progress) const {
        return start + progress * (end - start);
    }

private:
    int duration_;
    EasingFunction easing_;
};
```

example implementation using the CHpaint

```
Color start_color = SColorFace();
Color end_color = SColorHighlight();
Color current_color = animation.LerpColor(start_color, end_color);

...

ChPaint(w, r, look, current_color);

...

ChLookFn(MyCustomLookFunction, ButtonAnimationDuration(), ButtonEasingFunction());
```