Subject: Re: styling of widgets (animation / look and feel)
Posted by dodobar on Thu, 20 Apr 2023 11:06:16 GMT
View Forum Message <> Reply to Message

Some Generic Curve processing :roll:

```
BaseAnimation& BaseAnimation::EasingCurve(EasingCurve curve) {
    easing_curve_ = curve;
    return *this;
}

//allow for a user style curve
BaseAnimation& BaseAnimation::CustomCurve(const Vector<Pointf>& curve_points) {
    easing_curve_ = CUSTOM;
    custom_curve_points_ = curve_points;
    return *this;
}

double BaseAnimation::ApplyEasingCurve(double progress) {
    switch (easing_curve_) {
        case EASE_IN_QUAD:
            return progress * progress;
        case EASE_OUT_QUAD:
            return progress * (2 - progress);
        case EASE_IN_OUT_QUAD:
            return progress < 0.5 ? 2 * progress * progress : -1 + (4 - 2 * progress) * progress;
        case EASE_IN_CUBIC:
            return progress * progress * progress;
        case EASE_OUT_CUBIC:
            return (--progress) * progress * progress + 1;
        case EASE_IN_OUT_CUBIC:
            return progress < 0.5 ? 4 * progress * progress * progress : (progress - 1) * (2 * progress -
2) * (2 * progress - 2) + 1;
        case CUSTOM:
            if (custom_curve_points_.IsEmpty()) {
                return progress;
            }

            int index = 0;
            while (index + 1 < custom_curve_points_.GetCount() && progress >
custom_curve_points_[index + 1].x) {
                ++index;
            }

            if (index + 1 < custom_curve_points_.GetCount()) {
                Pointf p1 = custom_curve_points_[index];
                Pointf p2 = custom_curve_points_[index + 1];
```

```
            double ratio = (progress - p1.x) / (p2.x - p1.x);
            return p1.y + ratio * (p2.y - p1.y);
        } else {
            return custom_curve_points_.Top().y;
        }
    case LINEAR:
    default:
        return progress;
  }
}
```