Subject: Re: styling of widgets (animation / look and feel)
Posted by dodobar on Thu, 20 Apr 2023 11:00:33 GMT
View Forum Message <> Reply to Message

my understanding of the fundamental framework is Limited but the general implementation overview could be:

*Base animation class for common functionality.
*Specific animation subclasses (e.g., position, size, color).
*Modify U++ widgets to support animated properties.
*Implement an animation timing system (e.g., timer, loop, generic ease in\out curves).
*Integrate animations with the event system.
*Provide helper functions for common animations and transitions.?

Rough base class examples:

```
#include <Core/Core.h>
#include <CtrlCore/CtrlCore.h>

class BaseAnimation {
public:

    enum EasingCurveType {
        LINEAR,
        EASE_IN_QUAD,
        EASE_OUT_QUAD,
        EASE_IN_OUT_QUAD,
        EASE_IN_CUBIC,
        EASE_OUT_CUBIC,
        EASE_IN_OUT_CUBIC,
        CUSTOM
    };

    BaseAnimation(Ctrl& target, int duration)
        : target_(target), duration_(duration), easing_curve_(LINEAR) {}

    Callback WhenAnimationCompleted;

    BaseAnimation& Loop(bool loop);
    bool IsLooping() const;

    BaseAnimation& Reverse(bool reverse);
    bool IsReversed() const;

    BaseAnimation& Duration(int duration);
    int GetDuration() const;
```

```
    void Start();
    void Pause();
    void Resume();
    void Stop();

    double GetProgress() const;

    BaseAnimation& EasingCurve(EasingCurveType curve);
    BaseAnimation& CustomCurve(const Vector<Pointf>& curve_points);

protected:
    virtual void UpdateProgress(double progress) = 0;
    double ApplyEasingCurve(double progress);

private:
    Ctrl& target_;
    int duration_;
    EasingCurveType easing_curve_;
    Vector<Pointf> custom_curve_points_;
    int timer_;

    void OnTimer();
};
```

Example Position (sorry if the coding style is not 100% U++)

```
#include "BaseAnimation.h"
#include <CtrlCore/ctrl.h> // For Color and Point
#include <CtrlLib/CtrlLib.h> // For Widget class

class PositionAnimation : public BaseAnimation {
public:
    PositionAnimation(Widget& target, const Point& end_position, int duration);

protected:
    virtual void UpdateProgress(double progress) override;

private:
    Widget& target_;
    Point start_position_;
    Point end_position_;
};

PositionAnimation::PositionAnimation(Widget& target, const Point& end_position, int duration)
    : BaseAnimation(duration), target_(target), end_position_(end_position) {
    start_position_ = target.GetScreenView().TopLeft();
```

```
}

void PositionAnimation::UpdateProgress(double progress) {
    Point current_position;
    current_position.x = start_position_.x + (end_position_.x - start_position_.x) * progress;
    current_position.y = start_position_.y + (end_position_.y - start_position_.y) * progress;
    target_.SetRect(target_.GetScreenView().SetTopLeft(current_position));
}
```

Example Colour ?

```
class ColorAnimation : public BaseAnimation {
public:
    ColorAnimation(Widget& target, const Color& end_color, int duration);

protected:
    virtual void UpdateProgress(double progress) override;

private:
    Widget& target_;
    Color start_color_;
    Color end_color_;
};

ColorAnimation::ColorAnimation(Widget& target, const Color& end_color, int duration)
    : BaseAnimation(duration), target_(target), end_color_(end_color) {
    start_color_ = target.GetBackground();
}

void ColorAnimation::UpdateProgress(double progress) {
    Color current_color;
    current_color.r = start_color_.r + (end_color_.r - start_color_.r) * progress;
    current_color.g = start_color_.g + (end_color_.g - start_color_.g) * progress;
    current_color.b = start_color_.b + (end_color_.b - start_color_.b) * progress;
    target_.SetBackground(current_color);
}
```

Not So sure about the integration to the widgets but a approach ?

```
#include "PositionAnimation.h"
#include "ColorAnimation.h"

namespace Upp {
```

```
class StaticText : public Ctrl {
public:
    // ... existing class content ...

    StaticText& AnimatePosition(const Point& end_position, int duration);
    StaticText& AnimateColor(const Color& end_color, int duration);
    // ... other animations ?
private:
    void OnAnimationCompleted();
};

} // namespace Upp
```

Example Usage

```
#include <CtrlLib/CtrlLib.h>

using namespace Upp;

GUI_APP_MAIN {
    TopWindow win;
    win.SetRect(0, 0, 800, 600);

    StaticText static_text;
    static_text.SetLabel("Hello, Animated World!");
    static_text.SetRect(10, 10, 200, 50);
    win.Add(static_text);

    win.Run();

    // Example usage of the animation methods
    static_text.AnimatePosition(Point(100, 100), 500); // Move StaticText to (100, 100) in 500 ms
    static_text.AnimateColor(Blue(), 1000); // Change background color to blue in 1000 ms
}
```

Custom example

```
Vector<Pointf> custom_curve_points = {
    {0.0, 0.0},
    {0.4, 0.8},
    {0.6, 0.2},
    {1.0, 1.0}
};
```

```
PositionAnimation animation(static_text, Point(100, 100), 500);
animation.CustomCurve(custom_curve_points);
animation.Start();
```