Qt 美化之基础动画

Xin.Zhang

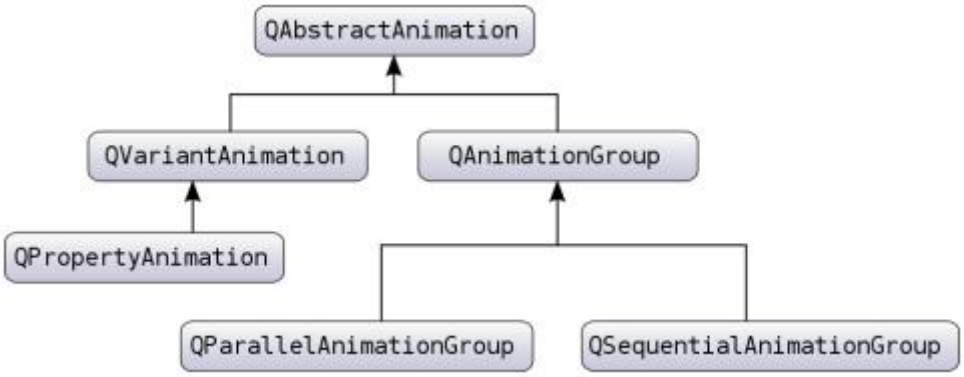
2017-02-22

前沿

界面动画能增加美感，在 QML 中使用动画很方便，但是在 Native 中使用稍微有些难度，这里将 Qt 内置的动画功能统一梳理一遍。

1. Qt 动画系统

Qt 的动画框架已经很简单了，可以用来对 QWidget 以及其他 QObject 进行属性动画。它也可以与 Graphics View 搭配，这套系统同样可以在 QML 中使用，并且更方便。



如上图 QAbstractAnimation 是所有动画的基类，它提供了一般动画所需要的所有元素，包含开始、停止、暂停等。它也接受时间变化的通知。

QPropertyAnimation（属性动画）是一个经常使用的动画，它一类于 Qt 的 Meta-object 系统，它使用一系列曲线的插值进行动画。

复杂的动画可以通过建立一个 QAbstractAnimation 树来完成，这些动画可以被放在 QAnimationGroup 中进行分组，并且在分组的动画中还可以包含其他组。

动画系统虽然一般使用插值系统，但也可以使用状态机。通过 QState 与 QPropertyAnimation 的绑定，我们可以控制状态的进入和离开。

注意：动画系统使用了一个全局的 Timer，它通过 updates 函数更新所有动画的播放情况。

QAbstractAnimation	所有动画的基类
QAnimationGroup	动画分组容器
QParallelAnimationGroup	并行动画组
QPauseAnimation	串行动画的暂停
QPropertyAnimation	属性动画
QSequentialAnimationGroup	串行动画组
QVariantAnimation	属性动画的基类，最基础的变量动画
QEasingCurve	控制动画的运动曲线
QTimeLine	控制动画的时间线

2. 动画实例

2.1 简单属性动画

之所以叫做属性动画就是因为，这里的动画都是改变 Widget 的某一个属性。但是通过串行/并行动画，可以完成复杂的效果。

```
QPushButton button("Animated Button");
button.show();
QPropertyAnimation animation(&button, "geometry");
animation.setDuration(10000);
animation.setStartValue(QRect(0, 0, 100, 30));
animation.setEndValue(QRect(250, 250, 100, 30));
animation.start();
```

```
QPushButton button("Animated Button");
button.show();
QPropertyAnimation animation(&button, "geometry");
animation.setDuration(10000);
animation.setKeyValueAt(0, QRect(0, 0, 100, 30)); // 相当于自定义的运动曲线
animation.setKeyValueAt(0.8, QRect(250, 250, 100, 30));
animation.setKeyValueAt(1, QRect(0, 0, 100, 30));
animation.start();
```

```
QPushButton button("Animated Button");
button.show();
QPropertyAnimation animation(&button, "geometry");
animation.setDuration(3000);
animation.setStartValue(QRect(0, 0, 100, 30));
animation.setEndValue(QRect(250, 250, 100, 30));

animation.setEasingCurve(QEasingCurve::OutBounce); // 这里设置速度曲线

animation.start();
```

2.2 并行动画

使用 QAnimationGroup (QSequentialAnimationGroup 或 QParallelAnimationGroup) 将多个动画一起执行：

```
QPushButton button("Animated Button");
button.show();
QPropertyAnimation anim1(&button, "geometry");
anim1.setDuration(3000);
anim1.setStartValue(QRect(0, 0, 100, 30));
```

```
anim1.setEndValue(QRect(500, 500, 100, 30));

QPropertyAnimation anim2(&button, "geometry");
anim2.setDuration(3000);
anim2.setStartValue(QRect(500, 500, 100, 30));
anim2.setEndValue(QRect(1000, 500, 100, 30));

QSequentialAnimationGroup group;
group.addAnimation(&anim1);
group.addAnimation(&anim2);
group.start();
```

2.3 状态机

状态机，每次切换状态都会触发一个属性动画：

```
QPushButton *button = new QPushButton("Animated Button");
button->show();
QStateMachine *machine = new QStateMachine;

QState *state1 = new QState(machine);
state1->assignProperty(button, "geometry", QRect(0, 0, 100, 30));
machine->setInitialState(state1);

QState *state2 = new QState(machine);
state2->assignProperty(button, "geometry", QRect(250, 250, 100, 30));

QSignalTransition *transition1 = state1->addTransition(button,
SIGNAL(clicked()), state2);
transition1->addAnimation(new QPropertyAnimation(button, "geometry"));

QSignalTransition *transition2 = state2->addTransition(button,
SIGNAL(clicked()), state1);
transition2->addAnimation(new QPropertyAnimation(button, "geometry"));

machine->start();
```

2.4 粒子特效

只有 OpenGL 有例子特效，可以使用 QML 完成。

3. 不正经的动画

3.1 GIF 动画

在实际项目中，我们经常遇到要播放 GIF 动画的需求，这种“动画”，不是严格的动画，它需要使用 QMovie 这个类来进行：

```
QMovie *Movie = new QMovie(":/movie/1");
Movie->setSpeed(1000);
Movie->setBackgroundColor(QColor(10, 10, 10));
QLabel    *Label = new QLabel();
Label->setMovie(Movie);
Label->show();
Movie->start();
```

内部使用的是状态机完成的，当缓存所有帧之后，可以使用 jump 函数手动跳转帧图片。

3.2 Flash 动画

1. 在 Qt 中直接播放 Flash

还算简单，使用 QAxWidget 控件调用 COM 可以实现，还可以通过 dynamicCall 函数调用 COM 接口。

但是，他有一个缺陷：不支持透明 Flash。

Qt 与 Flash 互相调用：http://blog.csdn.net/ydzheu/article/details/53925189

2. 使用 Web 框架

直接通过 qwebview 显示 flash，需要下载 webkit 的 flash 插件 NPSWF32.dll。通过调用本地文件（如：file:///d:/myswf.swf ）就可以直接使用。

```
#include <QApplication>
#include <QWebView>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);

    QWebSettings *settings = QWebSettings::globalSettings();
    settings->setAttribute(QWebSettings::PluginsEnabled, true);
    settings->setAttribute(QWebSettings::JavascriptEnabled, true);

    QWebView *webView = new QWebView;
```

```
    webView->load(QUrl("file:///d:/myswf.swf"));
    webView->show();

    return app.exec();
}
```

3. 编写 WebKit 插件

这个办法可以用来解析任何文件，但是比较复杂。

4. 内嵌 OpenGL 动画

将 OpenGL 动画内嵌到窗口中，需要借助 QOpenGLWidget。OpenGL 变成比较复杂。默认情况下，OpenGL 期望它们按逆时针顺序。

5. 内嵌 QML 动画

这也是一种可行的方案，需要借助 QQuickWidget。