Carousel Animation with the Qt graphics view framework

This article shows how to create a carousel animation using the Qt Graphics View framework. The carousel shows just how easy it is to create powerful animations even on devices that lack 3D acceleration hardware.

02 Jan 2011

Introduction

Carousel animation is a popular animation which shows items (for example: shapes, text, pixmaps) rotating around a center. This article illustrates how to create the carousel effect using the 2D Qt graphics view framework, a framework that allows you to enhance the appearance of mobile applications even in those phones where the 3D acceleration is not available.

Since the goal of this example is to show how to implement just the animation, the example code does not respond to user interaction. The media player is loading...

Implementation

The idea behind this animation is quite simple: Since it involves a group of items, the carousel class has been designed as subclass of `QGraphicsItemGroup`. Items can be added and removed using the `addToGroup` and `removeFromGroup` methods. In this example simple rectangle items have been used, but actually you can use every `QGraphicsItem` derived item.

To animate items, a `variantAnimator` class has been used to increase the rotation angle. That angle is used to calculate the position (x, y) of the items according to the ellipse parametric equation. The second variable of the ellipse parametric equation is the angle between Y and major ellipse axis, which is set to 0 in this example to keep the ellipse horizontally; this angle can be used to achieve different animations.

The `doAnimation` slot is the key part of the carousel class and it moves the items along the ellipse. Moreover, it scales and fades them according to the items' Y values. To show the item correctly the Z position of the items has been changed according to the Y one. Changing the Z position of the item could be a problem for the "*foreach(QGraphicsItem* i, items)*" loop if "items = childItems()". Indeed, the *foreach* loop has to keep the items in the same order all the time in order to avoid mixing the items in each frame. To solve this problem a new graphics item QList has been added.

Code

This is the carousel class:

```
#ifndef CAROUSEL_H
#define CAROUSEL_H
#include <QGraphicsItemGroup>
#include <QObject>

class variantAnimator;

class carousel: public QObject, public QGraphicsItemGroup
{
    Q_OBJECT

 public:
    carousel(QGraphicsItem * parent = 0);

    virtual void addToGroup(QGraphicsItem *item);
    virtual void removeFromGroup (QGraphicsItem *item);

 public slots:
    void doAnimation(const QVariant& angle);

 private:
    variantAnimator *mAngleAnimator;
```


(C) Copyright Nokia 2014. All rights reserved.

```
    double phi; // ellipse angle
    double a; // ellipse semi-major axis
    double b; // ellipse semi-minor axis
    uint duration; // animation duration
    QList <QGraphicsItem*> items; // childItems() cannot be used because of dynamic
graphicsItem Z value
};

#endif // CAROUSEL_H
```

```
#include "carousel.h"
#include "variantanimator.h"
#include <math.h>

#include <QDebug>

double pi = 3.14159265;

carousel::carousel(QGraphicsItem *parent) : QObject(0),
                                            QGraphicsItemGroup(parent),
                                            a(150),
                                            b(50),
                                            phi(0),
                                            duration(10000)
{
    // Set up animator
    mAngleAnimator = new variantAnimator;
    mAngleAnimator->setStartValue(0);
    mAngleAnimator->setEasingCurve(QEasingCurve::Linear);
    connect(mAngleAnimator, SIGNAL(valueChanged(const QVariant&)),
SLOT(doAnimation(const QVariant&)));

    mAngleAnimator->setEndValue(360);
    mAngleAnimator->setDuration(duration);
    mAngleAnimator->start();
}

void carousel::addToGroup(QGraphicsItem *item){
    items << item;
    QGraphicsItemGroup::addToGroup(item);
}

void carousel::removeFromGroup(QGraphicsItem *item){
    items.removeAll(item);
    QGraphicsItemGroup::removeFromGroup(item);
}

void carousel::doAnimation(const QVariant& angle){
    if (!scene())
        return;

    const int steps = items.count();
    int step = angle.toInt();

    foreach(QGraphicsItem* i, items){
```


(C) Copyright Nokia 2014. All rights reserved.

```
        step += 360/steps;

        const double alpha = step * (pi/180.0); // Converts degree into radians
        const double beta = -phi * (pi/180.0);

        double x = (a * cos(alpha) * cos(beta)) - (b * sin(alpha) * sin(beta));
        double y = (a * cos(alpha) * sin(beta)) + (b * sin(alpha) * cos(beta));

        /* Scaling and Fading */
        {
            // Setting parameters
            const qreal range = 2 * b; // y range [-b,b]
            const qreal max = 1.5;     // bigger value at b
            const qreal min = 0.4;     // lower value at -b

            // linear equation: y = mx +c
            qreal adjustedY = y + b;
            qreal scaleFactor = ((max - min)/range)* adjustedY + min;

            i->setScale(scaleFactor);
            i->setOpacity(qBound(0.0, scaleFactor, 1.0));
        }

        /* Stack position */
        {
            // Setting parameters
            const qreal range = 2 * b; // y range [-b,b]
            const qreal max = 1.0;     // bigger value at b
            const qreal min = 0.0;     // lower value at -b

            // linear equation: y = mx +c
            qreal adjustedY = y + b;
            qreal z = ((max - min)/range)* adjustedY + min;

            i->setZValue(z);
        }

        /* Moving items using the item's center as origin */
        {

            // bounding rectangle is in item coordinates, so it is not affected by
transformations such as scale
            x -= i->sceneBoundingRect().width()/2;
            y -= i->sceneBoundingRect().height()/2;

            i->setPos(x, y);
        }
    }
}
```

Example

The full code of this example is available here: Media:QtCarouselAnimation.zip

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.