Bouncing ball animations using Qt Quick

This article demonstrates how to use different types of QML Animations and Transitions using the example of an animated bouncing ball.

Introduction

This article shows how to perform basic animations of QML; once you are familiar with the basic animation you will find easy to deal with the advanced animations.

PropertyAnimation and Animation elements are the base of QML animation. All other animation elements are derived from these. The names of each of the elements tell us what is they are used for. PropertyAnimation can be used to animate various property types.

Basic Idea

Before we go ahead and start on different types of animation lets create a simple bouncing ball animation and see how easy to understand.

Bouncing Ball Animation

main.qml

```
import QtQuick 1.0
Rectangle {
    width: 360
    height: 360
    id:rect;
    Image{
    id:img
    source:"BouncingBallQtQuick.png"
    anchors.horizontalCenter: parent.horizontalCenter
    SequentialAnimation on y{
    id:anim
    running:true;loops:Animation.Infinite
    NumberAnimation{
    to:300-img.height;duration:2000; easing.type:Easing.OutBounce
    }
    PauseAnimation{duration:1000}
    NumberAnimation{
    to:0;duration:1000; easing.type:Easing.OutQuad
        }
       }
      }
     }
```

The SequentialAnimation element allows animations to be run sequentially. The NumberAnimation element animates changes in *qreal-type* values . The PauseAnimation element provides a pause for an animation.

Animation Types

Standalone animations

- It can be started and stopped from a script using the `running` property or `start()` , `stop()` methods

```
    Item{
    width:200;height:200
    PropertyAnimation{
        id:animation
        target:rect;
        properties:"x,y";
        duration:1000
    }
    Rectangle{
        id:rect;width:100;height:100;color:"red"
        MouseArea{
            anchors.fill:parent
            onClicked:{
                animation.to=50;
                animation.running=true;//Or:animation.start();
            }
        }
    }
}
```


(C) Copyright Nokia 2013. All rights reserved.

Behavioral animations

- It triggered when a property's value changes
- Here we don't need to specify the `from` and `to` value
- It uses the `enabled` property to enable and disable the animation

```
      Item{
  width:200;height:200
  Rectangle{
      id:rect; width:100;height:100; color:"red"
      Behavior on x{PropertyAnimation{duration:500}}
      Behavior on y{PropertyAnimation{duration:500}} }
  MouseArea{ anchors.fill:parent
      onClicked:{rect.x=mouse.x;rect.y=mouse.y}
            }
    }
```

Animations as property value sources

- The animation is triggered as soon as the target element is loaded.
- In this case animation runs from property's current value to specified `to` value.
- We can also specify a `from` value if needed.
- Can run for once or can also run in an infinite loop.

```
 Item{
    width:200;height:200
    Rectangle{
        width:100;height:100; color:"red"
    PropertyAnimation on x{//x = 0 by default
    to:50;duration:1000;loops:Animation.Infinite
    }
    PropertyAnimation on y{//y = 0 by default
    to:50;duration:1000;loops:Animation.Infinite
        }
      }
    }
```

Animations within signal handlers

- It gets triggered with a signal is received
- It is not bound to any specific element.
- It must specify `target`, property, and `to` value

```
Item{
        width:200;height:200
        Rectangle{
            id:rect; width:100;height:100; color:"red"
            MouseArea{
                anchors.fill:parent
                onClicked:PropertyAnimation{
                    target:rect
                    properties:"x,y"
                    to:50
                    duration:1000
                }
            }
        }
    }
```

Easing Curves

Animation runs with a linear fashion with a constant speed from start to finish. Easing curves can be used to change this default behavior with the help of `easing.type` or any `Animatoin` element.

```
Item{
    width:200;height:200
    Rectangle{
    width:50;height:50; color:"red"
    PropertyAnimation on x{//x = 0 by default
    to:150;duration:1000; easing.type: Easing.OutBounce;loops:Animation.Infinite
            }
        }
    }
```

For more about Easing Curves please refer to the Qt Demo Tool under Animation category.

Source Code

The full source code of bouncing ball example is available here: File:BouncingBallQtQuick.zip

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.