1 Behavior

定义一个属性变化的时候的动画效果，一个属性不可用超过一个动画的behavior提供多个动画效果应使用parallelAnimation 以及SequentialAnimation

Defines a default animation for a property change

A Behavior defines the default animation to be applied whenever a particular property value changes.

Note that a property cannot have more than one assigned Behavior. To provide multiple animations within a Behavior, use ParallelAnimation or SequentialAnimation.

import QtQuick 2.0

Rectangle {

id: rect

width: 100; height: 100

color: "red"

Behavior on width {//在width属性变化的时候使用此动画

NumberAnimation { duration: 1000 }

}

MouseArea {

anchors.fill: parent

onClicked: rect.width = 50

}

}

2 Transition 在状态变换时加入动画效果

A Transition defines the animations to be applied when a State change occurs.

if multiple Transitions are specified, only a single (best-matching) Transition will be applied for any particular state change. In the example above, when changing to state1, the first transition will be used, rather than the more generic second transition.

If a state change has a Transition that matches the same property as a Behavior, the Transition animation overrides the Behavior for that state change.

当多个transition被指定，仅仅最匹配的那个会被应用到指定的状态变化时，如果一个属性同时被用在tansition中和behavior中，那么tansition覆盖behavior

import QtQuick 2.0

Rectangle {

id: rect

width: 100; height: 100

color: "red"

MouseArea { id: mouseAreaanchors.fill: parent }

states: State {

name: "moved"; when: mouseArea.pressed

PropertyChanges { target: rect; x: 50; y: 50 }

}

transitions: Transition {

NumberAnimation { properties: "x,y"; easing.type: Easing.InOutQuad }

}

}

transitions: [

Transition {

from: "stop"; to: "go"

PropertyAnimation { target: stopLight

properties: "color"; duration: 1000 }

},

Transition {

from: "go"; to: "stop"

PropertyAnimation { target: goLight

properties: "color"; duration: 1000 }

} ]多个过渡，只有匹配的那个起作用

3 SequentialAnimation 对定义的动画串行播放

动画一个接一个的播放，可以用在behavior以及tansitions中

import QtQuick 2.0

Rectangle {

id: rect

width: 100; height: 100

color: "red"

SequentialAnimation {

running: true

NumberAnimation { target: rect; property: "x"; to: 50; duration: 1000 }

NumberAnimation { target: rect; property: "y"; to: 50; duration: 1000 }

}

}

4 ParallelAnimation 对定义的动画并行播放

5 PropertyAnimation 属性变换动画

一个属性值变化的时候提供动画效果。可以用在多种方式下，比如Behavior tansitions 单独作为属性，曹函数中，单独的块。祥见助手

**In a Transition**

Rectangle {

id: rect width: 100; height: 100color: "red"

states: State {

name: "moved"

PropertyChanges { target: rect; x: 50 }

}

transitions: Transition {

PropertyAnimation { properties: "x,y"; easing.type: Easing.InOutQuad }

}

}

**In a Behavior**

For example, to animate all changes to a rectangle's x property:

Rectangle {

width: 100; height: 100

color: "red"

Behavior on x { PropertyAnimation {} }

MouseArea { anchors.fill: parent; onClicked: parent.x = 50 }

}

**As a property value source**

For example, to repeatedly animate the rectangle's x property:

Rectangle {

width: 100; height: 100

color: "red"

SequentialAnimation on x {

loops: Animation.Infinite

PropertyAnimation { to: 50 }

PropertyAnimation { to: 0 }

}

}

**In a signal handler**

For example, to fade out theObject when clicked:

MouseArea {

anchors.fill: theObject

onClicked: PropertyAnimation { target: theObject; property: "opacity"; to: 0 }

}

**Standalone**

For example, to animate rect's width property over 500ms, from its current width to 30:

Rectangle {

id: theRect;width: 100; height: 100; color: "red"

// this is a standalone animation, it's not running by default

PropertyAnimation {

id: animation; target: theRect; property: "width"; to: 30; duration: 500 }

MouseArea { anchors.fill: parent; onClicked: animation.running = true }

}

6 NumberAnimation 对实数类型属性进行的动画

**继承于propertyAnimation用法差不多，在一个数字值属性变化的时候应用动画效果**

Vector3dAnimation 对QVector3d进行的属性

7 ColorAnimation 颜色进行的变换动画

**继承于propertyAnimation用法差不多，在一个颜色值属性变化的时候应用动画效果**

8 RotationAnimation 对旋转进行的变换动画

**顺时针或者逆时针进行旋转，direction属性指定这个方向，**RotationAnimation.Counterclockwise为逆时针，RotationAnimation.clockwise为顺时针

9 ParentAnimation 对父节点进行变换的动画，改变绑定的父节点的时候可以使用

10 AnchorAnimation 对anchor 进行改变的动画，布局改变的时候可以使用

11 PauseAnimation 多个动画的时候之间可以穿插这个延迟处理

12 SmoothedAnimation 允许属性平滑的过度

13 SpringAnimation 一种弹性的效果

PropertyAction 允许在动画过程中对属性的直接改变

ScriptAction 允许动画过程中调用脚本

14 **Scale缩放**

Rectangle {

width: 100; height: 100；color: "blue"

transform: Scale { origin.x: 25; origin.y: 25; xScale: 3}

}

15 **Rotation旋转**

Rectangle {

width: 100; height: 100

color: "blue"

transform: Rotation { origin.x: 25; origin.y: 25; angle: 45}

}

16 **Translate平移**

import QtQuick 2.0

Row {

Rectangle {

width: 100; height: 100

color: "blue"

transform: Translate { y: 20 }

}

Rectangle {

width: 100; height: 100

color: "red"

transform: Translate { y: -20 }

}

}

**A Transition defines the animations to be applied when a State change occurs.**

6. **AnimatedImage** 主要用于播放gif图片

Rectangle {

width: animation.width;

height: animation.height + 8

AnimatedImage {

id: animation;

source: "animation.gif"

}

Rectangle {

property int frames: an imation.frameCount

width: 4; height: 8

x: (animation.width - width) \* animation.currentFrame / frames

y: animation.height

color: "red"

}

}

16. **Flipable**

包含两个面，一个前面，一个后面，实现一个控件前后的翻转效果，并且在后面可以添加一些控制

Flipable {

id: flipable

width: 240

height: 240

property int angle: 0

property bool flipped: false

front: Image { source: "front.png" } // 前面

back: Image { source: "back.png" } // 后面

// 旋转动画 前后面交换

transform: Rotation {

origin.x: flipable.width/2;

origin.y: flipable.height/2

axis.x: 0; axis.y: 1; axis.z: 0 // rotate around y-axis

angle: flipable.angle

}

states: State {

name: "back"

PropertyChanges { target: flipable; angle: 180 }

when: flipable.flipped

}

transitions: Transition {

NumberAnimation { properties: "angle"; duration: 1000 }

}

MouseArea {

anchors.fill: parent

onClicked: flipable.flipped = !flipable.flipped

}

}

15. **Flickable**

显示一个200x200的框，框中显示图片上200x200的部分,并且可以拖动这个框来看图片的其他显示部分

Flickable {

width: 200; height: 200

// 设置使用图片的宽 高，而现实的是 200x200的现实框

contentWidth: image.width; contentHeight: image.height

Image { id: image; source: "../Images/need.png" }

}

23. **Behavior**

Rectangle {

id: rect

width: 100; height: 100

color: "red"

// 针对宽度的动画

Behavior on width {

NumberAnimation { duration: 1000 }

}

MouseArea {

anchors.fill: parent

onClicked: rect.width = 50

}

24. **SequentialAnimation**

串行播放多个动画

Rectangle {

id: rect1

width: 500; height: 500

Rectangle{

id: rect;

color: "red"

width: 100; height: 100

// 串行播放多个动画，先横向移动，在纵向移动

SequentialAnimation{

running: true;

NumberAnimation {target:rect; properties:"x"; to: 50; duration: 1000 }

NumberAnimation {target:rect; properties:"y"; to: 50; duration: 1000 }

}

}

}