QML Animations

Training Course

Visit us at `http://qt.digia.com`

Produced by Digia Plc.

Material based on Qt 5.0, created on September 27, 2012

Digia Plc.

Module: Animations

- Animations
- Easing Curves
- Animation Groups

Objectives

Can apply animations to user interfaces:

- Understanding of basic concepts
  - number and property animations
  - easing curves
- Ability to queue and group animations
  - sequential and parallel animations
  - pausing animations
- Knowledge of specialized animations
  - color and rotation animations

- Animations
- Easing Curves
- Animation Groups

Why use animations, states and transitions?

- Handle form factor changes
- Outline application state changes
- Orchestrate high level logic
- Natural transitions
- Our brain expects movement
- Helps the user find its way around the GUI
- Don't abuse them!

Demo qml-animations/ex-thumbnailexplorer/thumbnailexplorer.qml

Animations can be applied to any element

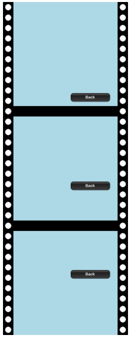
- Animations update properties to cause a visual change
- All animations are property animations
- Specialized animation types:
  - `NumberAnimation` for changes to numeric properties
  - `ColorAnimation` for changes to color properties
  - `RotationAnimation` for changes to orientation of items
  - `Vector3dAnimation` for motion in 3D space
- Easing curves are used to create variable speed animations
- Animations are used to create visual effects

See QML Animation Documentation

Number Animations

```
import QtQuick 2.0

Rectangle {
    width: 400; height: 400
    color: "lightblue"

    Image {
        x: 220
        source: "../images/backbutton.png"
        NumberAnimation on y {
            from: 350; to: 150
            duration: 1000
        }
    }
}
```

Demo qml-animations/ex-animations/number-animation.qml



Number Animations

Number animations change the values of numeric properties

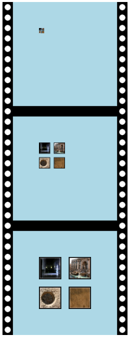
```
NumberAnimation on y {
    from: 350; to: 150
    duration: 1000

}
```

- Applied directly to properties with the `on` keyword
- The `y` property is changed by the `NumberAnimation`
  - starts at `350`
  - ends at `150`
  - takes `1000` milliseconds
- Can also be defined separately

Property Animations

```
import QtQuick 2.0

Rectangle {
    width: 400; height: 400; color: "lightblue"

    Image {
        id: image
        x: 100; y: 100
        source: "../images/thumbnails.png"
    }

    PropertyAnimation {
        target: image
        properties: "width,height"
        from: 0; to: 200; duration: 1000
        running: true
    }
}
```

Demo qml-animations/ex-animations/property-animation.qml



Property Animations

Property animations change named properties of a target

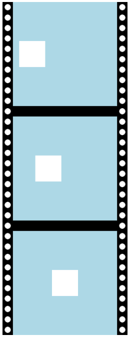
```
PropertyAnimation {
    target: image
    properties: "width,height"
    from: 0; to: 200; duration: 1000
    running: true

}
```

- Defined separately to the target element
- Applied to properties of the `target`
  - `properties` is a comma-separated string list of names
- Often used as part of a `Transition`
- Not run by default
  - set the `running` property to `true`

Number Animations Revisited

```
import QtQuick 2.0

Rectangle {
    width: 400; height: 400; color: "lightblue"

    Rectangle {
        id: rect
        x: 0; y: 150; width: 100; height: 100
    }

    NumberAnimation {
        target: rect
        properties: "x"
        from: 0; to: 150; duration: 1000
        running: true
    }
}
```

Demo qml-animations/ex-animations/number-animation2.qml



Number Animations Revisited

Number animations are just specialized property animations

```
NumberAnimation {
    target: rect
    properties: "x"
    from: 0; to: 150; duration: 1000
    running: true
}
```

- Animation can be defined separately
- Applied to properties of the `target`
  - `properties` contains a comma-separated list of property names
- Not run by default
  - set the `running` property to `true`

The Behavior Element

- Behavior allows you to set up an animation whenever a property changes.

```
Behavior on x {
    SpringAnimation {
        spring: 1
        damping: 0.2
    }
}
```

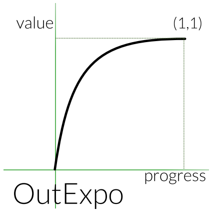
Demo qml-animations/ex-animations/spring-animation.qml

- Animations
- Easing Curves
- Animation Groups

```
import QtQuick 2.0

Rectangle {
    width: 400; height: 400
    color: "lightblue"

    Image {
        x: 220
        source: "../images/backbutton.png"
        NumberAnimation on y {
            from: 0; to: 350
            duration: 1000
            easing.type: "OutExpo"
        }
    }
}
```

Demo qml-animations/ex-animations/easing-curve.qml

Demo $QTDIR/examples/animation/easing



Apply an easing curve to an animation:

```
NumberAnimation on y {
    from: 0; to: 350; duration: 1000
    easing.type: "OutExpo"

}
```

- Sets the `easing.type` property
- Relates the elapsed time
  - to a value interpolated between the `from` and `to` values
  - using a function for the easing curve
  - in this case, the `"OutExpo"` curve

- Animations
- Easing Curves
- Animation Groups

Sequential and Parallel Animations

Animations can be performed sequentially and in parallel

- `SequentialAnimation` defines a sequence
  - with each child animation run in sequence
- For example:
  - a rescaling animation, followed by
  - an opacity changing animation

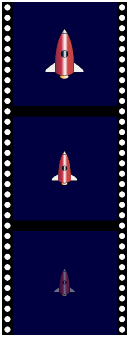
- `ParallelAnimation` defines a parallel group
  - with all child animations run at the same time
- For example:
  - simultaneous rescaling and opacity changing animations

Sequential and parallel animations can be nested

Sequential Animations

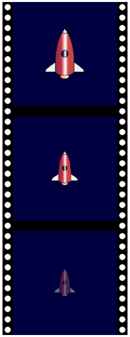
```
Image {
    id: rocket
    anchors.centerIn: parent
    source: "../images/rocket.png"
}

SequentialAnimation {
    NumberAnimation {
        target: rocket; properties: "scale"
        from: 1.0; to: 0.5; duration: 1000
    }
    NumberAnimation {
        target: rocket; properties: "opacity"
        from: 1.0; to: 0.0; duration: 1000
    }
    running: true

}
```

Demo qml-animations/ex-animations/sequential-animation.qml



Sequential Animations

```
SequentialAnimation {
    NumberAnimation {
        target: rocket; properties: "scale"
        from: 1.0; to: 0.5; duration: 1000
    }
    NumberAnimation {
        target: rocket; properties: "opacity"
        from: 1.0; to: 0.0; duration: 1000
    }
    running: true
}
```



- Child elements define a two-stage animation:
  - first, the rocket is scaled down
  - then it fades out
- `SequentialAnimation` does not itself have a `target`
  - it only groups other animations

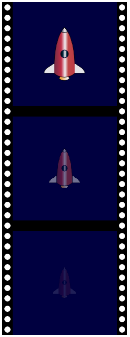
Pausing between Animations

```
SequentialAnimation {
    NumberAnimation {
        target: rocket; properties: "scale"
        from: 0.0; to: 1.0; duration: 1000
    }
    PauseAnimation {
        duration: 1000
    }
    NumberAnimation {
        target: rocket; properties: "scale"
        from: 1.0; to: 0.0; duration: 1000
    }
    running: true
}
```

Parallel Animations

```
Image {
    id: rocket
    anchors.centerIn: parent
    source: "../images/rocket.png"
}
ParallelAnimation {
    NumberAnimation {
        target: rocket; properties: "scale"
        from: 1.0; to: 0.5; duration: 1000
    }
    NumberAnimation {
        target: rocket; properties: "opacity"
        from: 1.0; to: 0.0; duration: 1000
    }
    running: true
}
```

Demo qml-animations/ex-animations/parallel-animation.qml

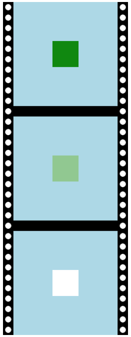


- Other animations
  - `ColorAnimation` for changes to color properties
  - `RotationAnimation` for changes to orientation of items
  - `Vector3dAnimation` for motion in 3D space
  - `AnchorAnimation` animate an anchor change
  - `ParentAnimation` animates changes in parent values.
  - `SpringAnimation` allows a property to track a value in a spring-like motion

  - `PropertyAction` the PropertyAction element allows immediate property changes during animation
  - `ScriptAction` allows scripts to be run during an animation

- ColorAnimation describes color changes to items
- Component-wise blending of RGBA values

```
ColorAnimation {
    target: rectangle1
    property: "color"
    from: Qt.rgba(0,0.5,0,1)
    to: Qt.rgba(1,1,1,1)
    duration: 1000
    running: true

}
```



Rotation Animation

- `RotationAnimation` describes rotation of items
- Easier to use than `NumberAnimation` for the same purpose
- Applied to the `rotation` property of an element
- Value of `direction` property controls rotation:
  - `RotationAnimation.Clockwise`
  - `RotationAnimation.Counterclockwise`
  - `RotationAnimation.Shortest` – the direction of least angle between `from` and `to` values

Rotation Animation

```
Image {
    id: ball
    source: "../images/ball.png"
    anchors.centerIn: parent
    smooth: true

    RotationAnimation on rotation {
        from: 45; to: 315
        direction: RotationAnimation.Shortest
        duration: 1000
    }

}
```

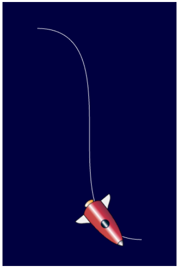
- 1 second animation
- Counter-clockwise from 45° to 315°
  - shortest angle of rotation is via 0°

Path Animation

- `PathAnimation` animates an item along a path
- Manipulates the `x`, `y` and `rotation` properties of an element
- The `target` element will be animated along the `path`
- Value of `orientation` property controls the `target` rotation:
  - `PathAnimation.Fixed`
  - `PathAnimation.RightFirst`
  - `PathAnimation.LeftFirst`
  - `PathAnimation.TopFirst`
  - `PathAnimation.BottomFirst`
- Value of `path` is specified using `Path` element and its helpers
  - PathLine, PathQuad, PathCubic, PathCurve, PathArc, PathSvg

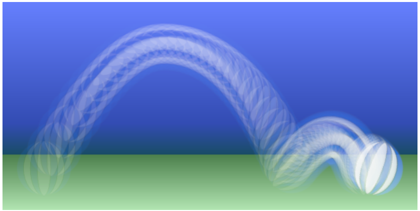
Path Animation

```
PathAnimation {
    duration: 2000
    easing.type: Easing.InOutQuad
    target: rocket
    orientation: PathAnimation.RightFirst
    anchorPoint: Qt.point(rocket.width/2,
                          rocket.height/2)

    path: Path {
        startX: rocket.width/2
        startY: rocket.height/2
        PathCubic {
            x: window.width - rocket.width/2
            y: window.height - rocket.height/2
            control1X: x; control1Y: rocket.height/2
            control2X: rocket.width/2; control2Y: y
        }
    }
}
```



Demo qml-animations/ex-animations/path-animation.qml

Lab: Bouncing Ball



Starting from the first partial solution:

- Make the ball start from the ground and return to the ground.
- Make the ball travel from left to right
- Add rotation, so the ball completes just over one rotation
- Reorganize the animations using sequential and parallel animations
- Make the animation start when the ball is clicked
- Add decoration (ground and sky)

Lab qml-animations/lab-animations