Qt Developer Conference 2011

QtQuick 培训课程

第二部分

Qt Developer Conference 2011

第二部分

目标

1 定位元素

绝对定位

相对定位

Anchors

2 实现动画

如何创建States

设定渐变和动画

各种类型的变换曲线和动画

第二部分

目标

3 QtQuick 和 Javascript 的无缝结合

通告方法和命令方法的结合（Declarative and imperative together）

在一个QtQuick 文件中创建javascript 函数

导入一个javascript 文件

QtQuick中的面向组件编程

第二部分

主题

1 定位元素

2 实现动画

3 QtQuick 和 Javascript 的无缝结合

4 问题解答

5 练习

Qt Developer Conference 2011

绝对定位

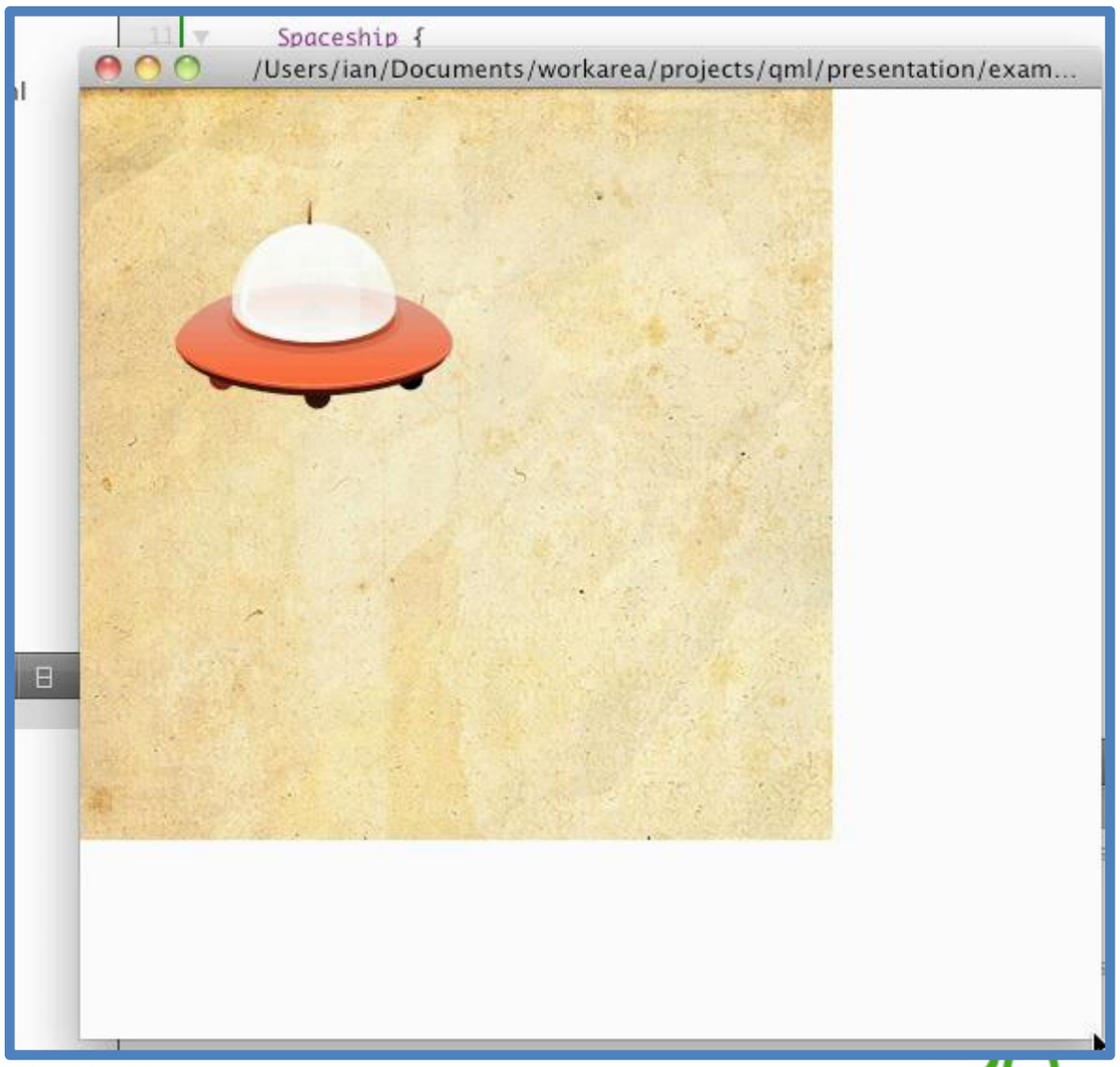
相对于父item，确定当前item的位置和大小

x, y, width and height

```
Item {
    width: 400
    height: 400

    Image {
        source: "images/background.png"
    }

    Spaceship {
        id: spaceship
        x: 50
        y: 60
    }
}
```



查看视频: addon/module-002/videos/basic-positioners.mov

查看示例: addon/module-002/examples/basic-positioners.qml

定位元素

相对定位

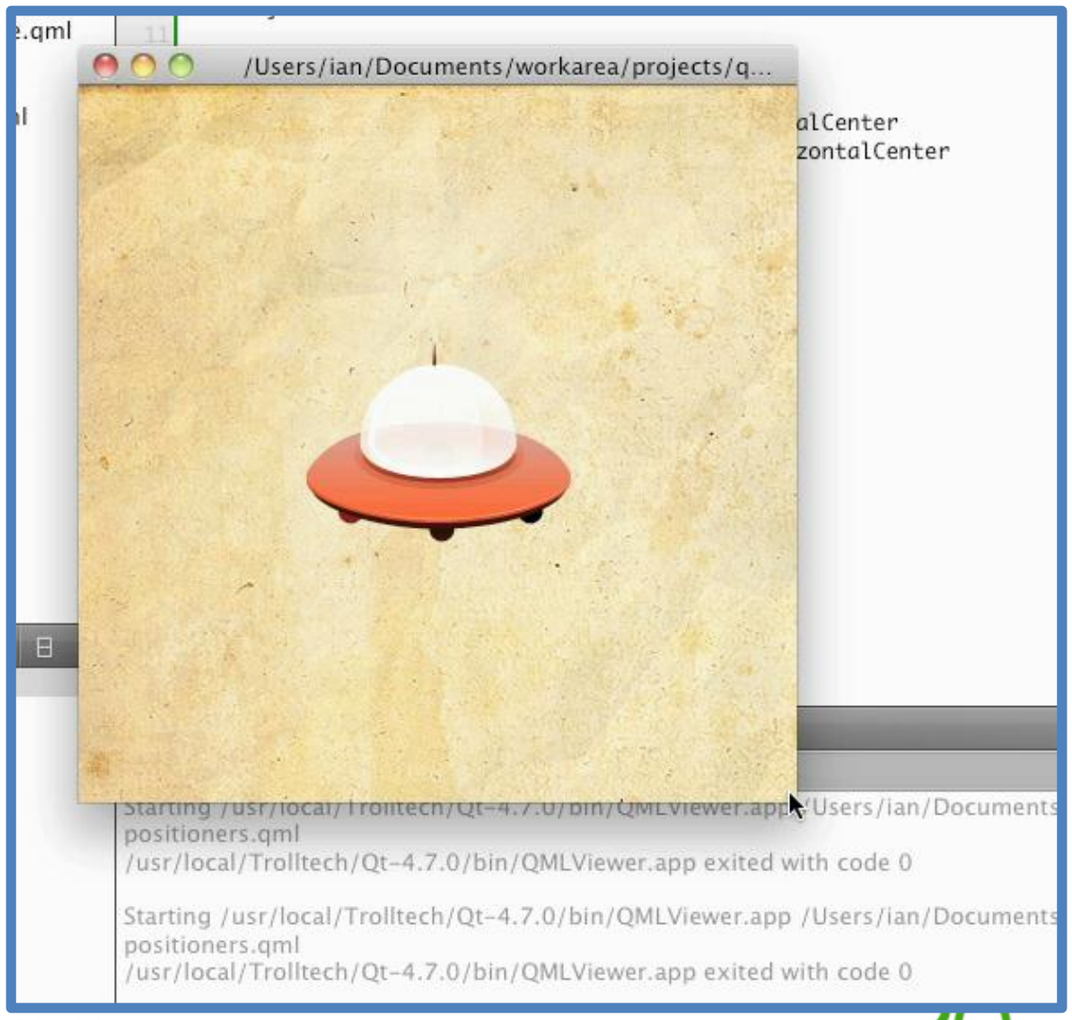
Anchors 提供一种方法:可以确定一个Item与其他item的相对位置

anchors

```
Item {
    width: 400; height: 400

    Image {
        anchors.fill: parent
        source: "images/background.png“
    }

    Spaceship {
        id: spaceship
        anchors.verticalCenter: parent.verticalCenter
        anchors.horizontalCenter: parent.horizontalCenter
    }
}
```



查看视频: addon/module-002/videos/anchors-positioners.mov

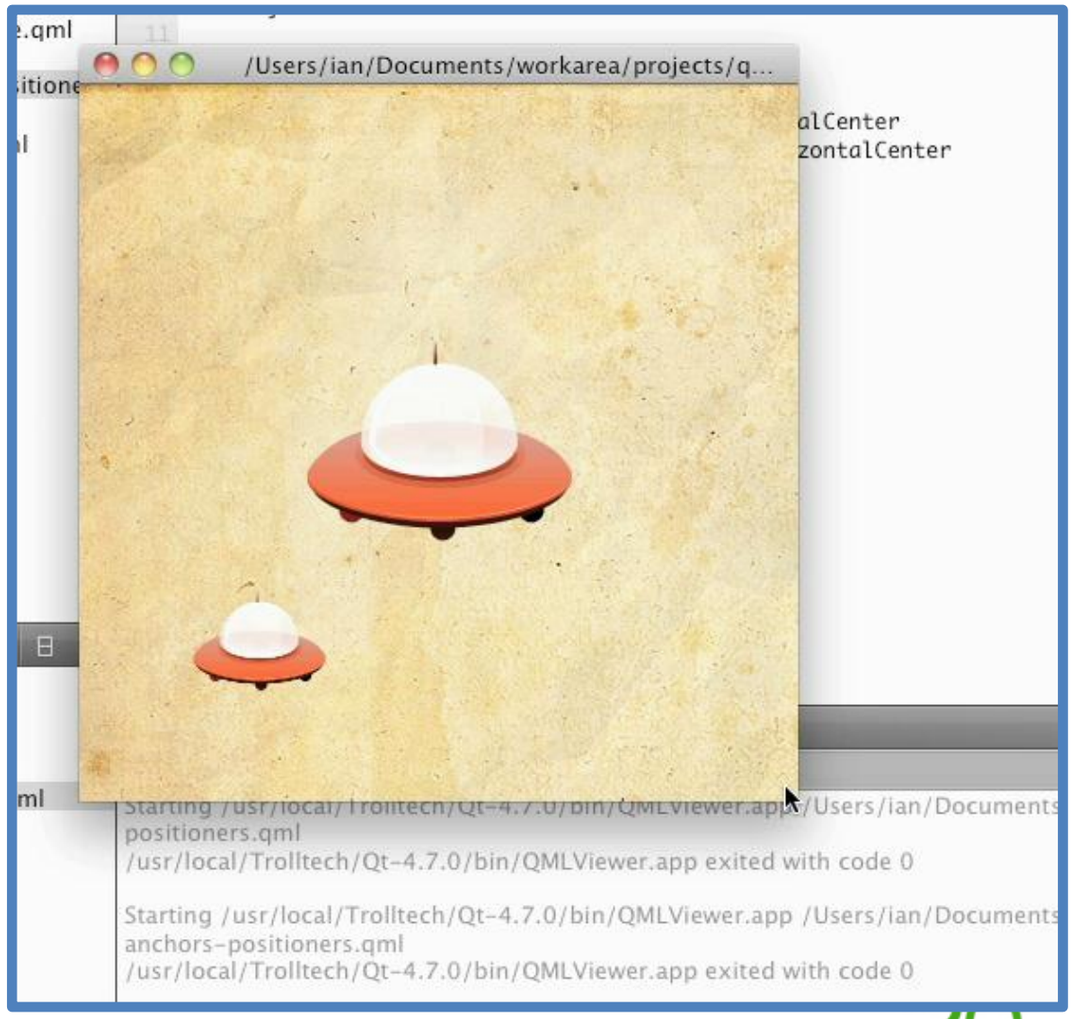
查看示例: addon/module-002/examples/anchors-positioners.qml

更多关于anchors

有多种方法可以确定当前Item相对于其他Item的位置

anchors.right, anchors.rightMargin ...

```
Item {
    width: 400
    height: 400
...
    Spaceship {
        id: spaceship
        anchors.verticalCenter: parent.verticalCenter
        anchors.horizontalCenter: parent.horizontalCenter
    }
    Spaceship {
        anchors.top: spaceship.bottom
        anchors.right: spaceship.right
        anchors.rightMargin: 100
    }
}
```



查看视频: addon/module-002/videos/more-anchors.mov

查看示例: addon/module-002/examples/more-anchors-positioners.qml

第二部分

主题

1 定位元素

2 实现动画

3 QtQuick和Javascript的无缝结合

4 问题解答

5 练习

如何创建States

State 元素定义对象和其属性的配置信息.

```
Item {
    id: myItem
    width: 400
    height: 400

    Image {
        id: spaceship
        source: "images/spaceship.png"
        x: 10
        y: 50
    }

    states: [
        State {
            name: "leftXMove"
            PropertyChanges {
                target: spaceship
                x: 200
            }
        }
    ]
}
```

如何创建States

可以为对象创建所需要的任意多个State

```
...
states: [
    State {
        name: "leftXMove"
        PropertyChanges {
            target: "spaceship"
            x: 200
        }
    },
    State {
        name: "downYMove"
        PropertyChanges {
            target: "spaceship"
            y: 90
        }
    }
 ]
}
...
```

各个State之间的属性不会相互传递

创建States的作用

容易从一种state转换如另一种state

```
Image {
    id: button; source: "images/button.png"
    y: 50; x: 10
    MouseArea {
        anchors.fill: parent; onClicked: myItem.state = 'leftXMove'
    }
}
```

通过执行对应函数设置Item的不同状态

or

```
Image {
    id: button; source: "images/button.png"
    y: 50; x: 10
    MouseArea {
        id:mouseArea; anchors.fill: parent
    }
}
states: State {
    name: "leftXMove"; when: mouseArea.clicked
    PropertyChanges { target: myItem; x: 200 }
 }
```

使用 “when”方法. 它可以在一个state内改变Item的属性

创建States的作用

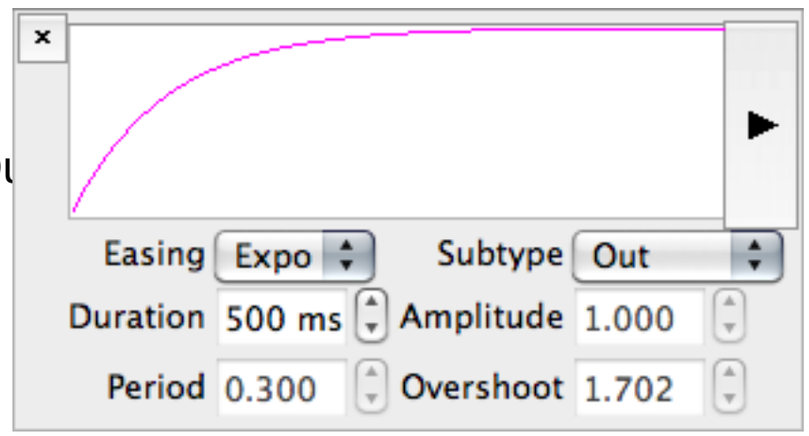
效果如下...

查看视频: addon/module-002/videos/spaceship-no-motion.mov

从一个State到另外State的动画效果

只需要在不同state间添加一个transition元素就能达到效果

```
...
transitions: [
    Transition {
        from: "";
        to: "leftXMove"
        NumberAnimation {
            properties: "x, y";
            duration: 500;
            easing.type: Easing.Ou
        }
    }
]
...
```



助手: Qt Quick 工具栏

第一个state是为NULL(默认状态)

查看示例: addon/module-002/examples/animation-example.qml

主要的transition 和 animation 元素

from and to

元素的初始状态和最终状态

target

动画元素的Id

Properties

在动画过程中需要变换的各种属性，这些属性可以放在数组中

easing.type

选择一种变换曲线为动画添加特效

了解更多关于animation 内容参见:
http://doc.qt.nokia.com/4.7-snapshot/qdeclarativeelements.html

Animation 类型

有多种类型可以满足应用的需求

NumberAnimation

ParallelAnimation

SequentialAnimation

PauseAnimation

RotationAnimation

了解更多关于不同类型的animation 参见:
http://doc.qt.nokia.com/4.7-snapshot/qdeclarativeelements.html

NumberAnimation

通过动态改变“数类型的属性”实现动画

```
...
NumberAnimation {
    properties: "x, y";
    duration: 500;
    easing.type: Easing.OutExpo;
}
...
```

这是一种基本的动画元素，大部分的动画都是和改变数字相关的.

并行和顺序

可以在特定的序列中动态改变特定的属性

ParallelAnimation

```
ParallelAnimation {
    NumberAnimation {
        target :  myRect
        properties: "x";
        duration: 500;
        easing.type: Easing.OutExpo;
    }

    NumberAnimation {
        target :  myRect
        properties: "y";
        duration: 500;
        easing.type: Easing.OutExpo;
    }
}
```

SequentialAnimation

```
SequentialAnimation {
    NumberAnimation {
        target :  myRect
        properties: "x";
        duration: 500;
        easing.type: Easing.OutExpo;
    }

    NumberAnimation {
        target :  myRect
        properties: "y";
        duration: 500;
        easing.type: Easing.OutExpo;
    }
}
```

并行和顺序

可以在特定的序列中动态改变特定的属性

ParallelAnimation

查看视频: addon/module-002/videos/parallel.mov

SequentialAnimation

查看视频: addon/module-002/videos/sequential.mov

其它动画元素

RotationAnimation元素可以为动画添加特定的旋转属性

```
states: {
    State {
        name: "180";
        PropertyChanges { target: myItem; rotation: 180 }
    }
    State {
        name: "-90";
        PropertyChanges { target: myItem; rotation: -90 }
    }
}
transition: Transition {
    RotationAnimation {
        direction: RotationAnimation.Shortest
    }
}
```

其它动画元素

PauseAnimation元素可以为动画添加暂停效果

```
PauseAnimation {
    target :  myRect
    duration: 100
}
```

针对behavior的动画

```
Rectangle {
    width: 20; height: 20; color: "#00ff00"
    y: 200
    Behavior on y {
        NumberAnimation {
            easing.type: Easing.OutBounce
            duration: 200
        }
    }
}
```

第二部分

主题

1 定位元素

2 实现动画

3 QtQuick和Javascript 的无缝结合

4 问题解答

5 练习

通告方法和命令方法的结合（Declarative and Imperative together）

Qt Quick 通过属性的静态数据列出各种元素，JavaScript 可以为元素定义更复杂的行为

```
Item {
    id: label1
    x: 80
    width: 100
    height: 100

    Image {
        source: {
            if(pressed) {
                return "img2.png";
            }  else {
                return "img1.png";
            }
        }
    }
}
```

QtQuick中javascript函数

可以在QtQuick文件的任何地方添加javascript函数

```
function randomState()
{
    var statesArray = ["topLeft", "topRight", "bottomLeft", "bottomRight"];
    var randomNumber = Math.floor(Math.random()*statesArray.length);
    return statesArray[randomNumber];
}
```

该函数根据定义的数组长度来产生一个随机数，然后返回数组中对应的值(状态)

查看示例: addon/module-002/examples/javascript-example.qml

QtQuick中的Javascript函数

在上一个示例中添加了该函数后的效果

查看视频: addon/module-002/videos/javascript.mov

查看源文件: addon/module-002/examples/javascript-example.qml

导入javascript 文件

可以通过导入javascript文件的方法，使代码更有条理

```
import QtQuick 1.0
import "random.js" as RandomFunction

Item {
    id: myItem
    width: 400

...

    MouseArea {
        anchors.fill: parent
        onClicked: {
            myItem.state = RandomFunction.randomState();
        }
    }
}
```

查看示例: addon/module-002/examples/importing-javascript.qml

QtQuick中的面向组件编程

为元素创建可复用的组件是循环使用代码段的好方法。例如：一个按钮!

Button.qml

```
Image {
    id: button
    source: "images/button.png"

    property string labelText

    Text {
        id: label
        text: labelText
        color: "white"
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.top: parent.top
        anchors.topMargin: 6
    }

}
```

查看示例: addon/module-002/examples/reusing-button.qml

QtQuick的组件编程

这就是一个可复用的按钮组件，不过它还需要一些改进。如：
onClicked 事件应放在 Button.qml文件中

reusing-button.qml

```
...
Button {
    id: button
    labelText: "PRESS"
    anchors.horizontalCenter: myItem.horizontalCenter
    anchors.bottom: myItem.bottom
    anchors.bottomMargin: 20

    MouseArea {
        anchors.fill: parent
        onClicked: {myItem.state = RandomFunction.randomState();}
    }
}
...
```

查看示例: addon/module-002/examples/reusing-button.qml

QtQuick中的组件编程

QtQuick中有一些默认属性可以在不同的类之间进行通信

Button.qml

```
Image {
    id: button
    source: "images/button.png"

    property string labelText
    signal buttonClicked

    Text {
        id: label
        text: labelText
        color: "white"
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.top: parent.top
        anchors.topMargin: 6
    }

}
```

这些属性都在文件的开始处进行声明

QtQuick中的组件编程

当按钮被点击，就会发出一个信号，你需要做的是根据信号做出对应的响应

Button.qml

```
Image {
    id: button
    source: "images/button.png"

    property string labelText
    signal buttonClicked
    ...
    MouseArea {
        anchors.fill: parent
        onClicked: {
            button.buttonClicked();
        }
    }
}
```

QtQuick中的组件编程

MouseArea 在Button类中定义了，当按钮点击信号发出后，函数就会被执行

reusing-button.qml

```
...
Button {
    id: button
    labelText: "PRESS"
    anchors.horizontalCenter: myItem.horizontalCenter
    anchors.bottom: myItem.bottom
    anchors.bottomMargin: 20

    onButtonClicked  {
        myItem.state = RandomFunction.randomState();
    }
}
...
```

查看示例: addon/module-002/examples/reusing-button.qml

第二部分

主题

1 定位元素

2 实现动画

3 QtQuick 和 Javascript 的无缝结合

4 问题解答

5 练习

Qt Developer Conference 2011

如何在不同state间创建动画?

顺序动画和并行动画的不同点有哪些?

如果我们没有声明from and to的渐变属性会发生什么?

在QtQuick文件中如何创建一个javascript函数?

如何在QtQuick中执行一个包含在导入js文件中的函数?

QtQuick中可以重用代码吗?

什么是信号?

第二部分

主题

1 定位元素

2 实现动画

3 QtQuick 和 Javascript 的无缝结合

4 问题解答

5 练习

太空大战! 重新实现下面的动画

查看视频: addon/module-002/videos/spaceship-attack.mov

选作部分: 把太空船用组件文件实现

查看练习: addon/module-002/labs/lab-animation/labTwo.qmlproject

Qt Developer Conference 2011