QtQuick Training Course

Module Four

Module Four

Objectives

1 Simple Views and Positioners

Column

Row

Repeater

Grid

Flow

2 Tips

Understanding binding

It’s a State problem

Making a QtQuick app faster

Module Four

Topics

1 Simple Views and Positioners

2 Tips

3 Questions

4 Lab

Simple views and positioners

More About Positioning Elements

Columns and Rows

Repeater

Grid and Flow

Columns

The column item arranges all its children vertically

```
...
Column {
        spacing: 20
        Image {
            source: "pic_1.jpg"
            Text {
                text: "1"
            }
        }
        Image {
            source: "pic_2.jpg"
            Text {
                text: "2"
            }
        }
        Image {
            source: "pic_3.jpg"
            Text {
                text: "3"
            }
        }
    }
...
```

spacing is the amount in pixels left empty between items, and defaults to 0.

Columns

This is the result using thumbnails from Flickr

See example: addon/module-004/examples/columnsExample.qml

Columns

The column also accepts items that have different sizes

See example: addon/module-004/examples/columnsItems.qml

Rows

The row item arranges all its children horizontally

See example: addon/module-004/examples/rowsExample.qml

Simple views and positioners

More About Positioning Elements

Columns and Rows

Repeater

Grid and Flow

Repeater

It allows you to repeat an Item-based component using a model

```
...
Row {
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.verticalCenter: parent.verticalCenter
        spacing: 20

        Repeater {
            id: repeater
            model: itemModel
            Image {
                source: model.url
                Text {
                    text: model.number
                }
            }
        }
    }
...
```

You can use the Repeater in any simple view. The result will look like the same one from the previous example, but your code will be much more organized

See example: addon/module-004/examples/repeatExample.qml

Repeater

The repeater model can be also an array or a number

Array

```
Repeater {
    id: repeater
    model: [“item1”, “item2”, “item3”]
    Text {
        text: modelData
    }
}
```

Number

```
Repeater {
    id: repeater
    model: 3
    Text {
        text: “item”+index
    }
}
```

See example: addon/module-004/examples/repeatExample.qml

Simple views and positioners

More About Positioning Elements

Columns and Rows

Repeater

Grid and Flow

Simple views and positioners

Grid

It positions its children in a grid

```
...
Grid {
        rows: 3
        columns: 2
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.verticalCenter: parent.verticalCenter
        spacing: 20
        Repeater {
            id: repeater
            model: itemModel
            Image {
                source: model.url
                Text {
                    text: model.number
                }
            }
        }
    }
...
```

See example: addon/module-004/examples/gridExample.qml

Grid

Grid inherits Item. It is simpler than GridView that inherits Flickable

See example: addon/module-004/examples/gridExample.qml

Simple views and positioners

Flow

It will organize its children according to their sizes in a specific area

```
...
Flow {
        width: 400
        anchors.horizontalCenter: parent.horizontalCenter
        anchors.verticalCenter: parent.verticalCenter
        spacing: 20
        Repeater {
            id: repeater
            model: itemModel
            Image {
                source: model.url
                Text {
                    text: model.number
                }
            }
        }
    }
...
```

See example: addon/module-004/examples/flowExample.qml

Simple views and positioners

Flow

This view is useful when you need to align items with different sizes respecting the spaces between them

See example: addon/module-004/examples/flowExample.qml

Animating Items

It’s as easy as animate items inside a ListView or GridView

See video: addon/module-004/videos/animating-items.mov

```
...
Grid {
        ...
        move: Transition {
            NumberAnimation {
                properties: "x,y"
                easing.type: Easing.OutExpo
            }
        }
        Repeater {
            id: repeater
            model: itemModel
            Image {
              ...
               Behavior on opacity { NumberAnimation {}
}
            }
...
```

See example: addon/module-004/examples/gridExampleMoving.qml

Module Four

Topics

1 Simple Views and Positioners

2 Tips

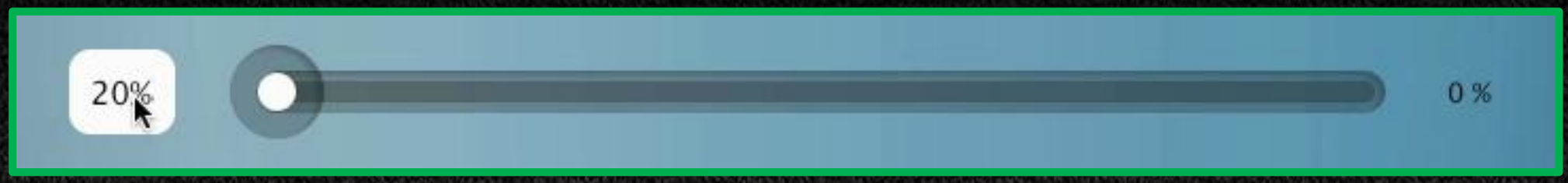
3 Questions

4 Lab

Tips

Understanding Binding

This example demonstrates a slider that has two types of interaction: dragging the knob or clicking on the button on the left



You need to move the knob x axis according to the mouse interaction and collect its movement percentage, so the knob.x property has a binding. The example above doesn’t work properly because of a binding misuse on the button click.

Slider.qml

```
Image {
        id: knob
        ...
        property real internalValue: Math.floor(100 * knob.x /
area.drag.maximumX)
        x: (internalValue / 100) * area.drag.maximumX
        MouseArea {
           ...
 }
```

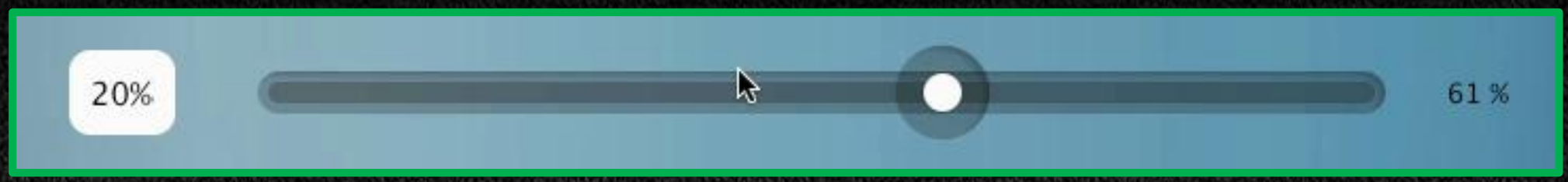
See video: addon/module-004/videos/binding-wrong.mov

See example: addon/module-004/examples/slider-with-property-overriden-and-exposed/main.qml

Tips

Understanding Binding

What happens is that you need to bind another variable, not doing this, the knob.x property would be overridden.



Slider.qml

```
...
Image {
        id: trackInside
        source: "../images/track_inside.png"
        anchors.verticalCenter: parent.verticalCenter
        anchors.horizontalCenter: parent.horizontalCenter
        MouseArea {
            id: grooveArea
            height: trackInside.height
            width: trackInside.width - marginSpace
            // This would override the binding that exists in "knob.x"
            //onClicked: knob.x = mouse.x
            onClicked: knob.xToBeSetFromMouseArea = mouse.x
        }
    }
...
```

See video: addon/module-004/videos/binding-right.mov

Tips

It’s a State problem

This example demonstrates a common problem. Sometimes you need to have two or more States to be set on the same component at the same time. See the example below

The pressed state is inside Button.qml but on root scope. Is this correct?

Button.qml

```
Item {
    id: root
    property string playerIcon: "../images/ico_pause.png"
...
    Image {
        id: icon
        ...
        source: playerIcon
    }
...
    states: State {
            name: "pressed"
            when: area.pressed && area.containsMouse
            PropertyChanges { target: buttonImage; source:
"../images/button_pressed.png" }
        }
...
```

See video: addon/module-004/videos/state-wrong.mov

See example: addon/module-004/examples/startstop-tip/main.qml

Tips

It’s a State problem

If you create a component that inherits Button and change a Button state from outside, there will be some problems. The states are on the same scope

The pressed state inside Button.qml must be in another scope

StartStopButton.qml

```
Button {
    id: root

    property bool running: false

    playerIcon: "../images/ico_pause.png"

    states: State {
        name: "running"
        when: root.running
        PropertyChanges { target: root; playerIcon: "../images/ico_play.png" }
    }

    onClicked: root.running = !root.running
}
```

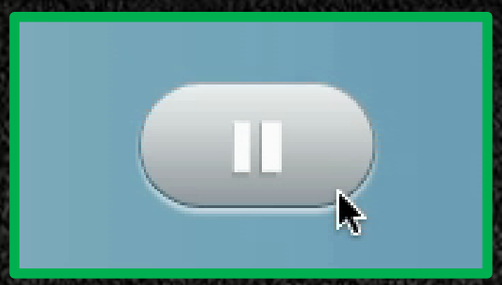
See video: addon/module-004/videos/state-wrong.mov

See example: addon/module-004/examples/startstop-tip/main.qml

Tips

It’s a State problem

It is possible to have two states at the same time if you change the state’s action scope. It is now protected inside MouseArea in this example below



Button.qml

```
Item {
    id: root
    property string playerIcon: "../images/ico_pause.png"
...
    MouseArea {
        id: area
        anchors.fill: parent
        onClicked: root.clicked()

        // The state is protected in here
        states: State {
            name: "pressed"
            when: area.pressed && area.containsMouse
            PropertyChanges {
                target: buttonImage; source: "../images/button_pressed.png" }
        }
...
```

See video: addon/module-004/videos/state-right.mov

See example: addon/module-004/examples/startstop-tip/main.qml

Tips

Making a QtQuick app faster

Don’t use Repeater if you have a huge amount of Items

Repeater can be inefficient if there are a large number of delegate items and not all of the items are required to be visible at the same time. If this is the case, consider using other view elements like ListView (which only creates delegate items when they are scrolled into view)

ListView cacheBuffer

This is interesting, because it is a matter of choice. If you set a huge number for the cacheBuffer, it will take more time to load but it will run smoother. You will need to find a proper number for your needs, specially if you want to run your app in a cellphone

Tips

Making a QtQuick app faster

Image smooth property

Set this property if you want the image to be smoothly filtered when scaled or transformed. Smooth filtering gives better visual quality, but is slower. If the image is displayed at its natural size, this property has no visual or performance effect.

Image width and height

If you set the smooth property to true and hardcode the Image width and height, even if you are changing just one pixel, it is probable that the whole application will run extremely slow. Avoid hardcoding the width and height, just let QtQuick define it for you.

Tips

Making a QtQuick app faster

If it is possible, use Grid instead of GridView

This also works for Columns or Rows instead of using ListView. Grid is a dummy view, all its properties are simplified in comparison to the GridView. It’s a view for you to use when you just need the Items in a certain position and don’t want them to flick. You must keep in your mind that Grid isn’t scalable in terms of amount of Items and GridView is.

Module Four

Topics

1 Simple Views and Positioners

2 Tips

3 Questions

4 Lab

Questions

What is the difference between GridView and Grid?

When a property is considered bound?

What is a Repeater?

Where can you use a Repeater?

How do you set two different states in a component?

What do a cacheBuffer do in a ListView?

Module Four

Topics

1 Simple Views and Positioners

2 Tips

3 Questions

4 Lab

Lab

Add a play/pause button to a ListView

See video: addon/module-004/videos/lab-slider.mov

Optional: Create a slider to control the animation velocity

See lab: labs/lab-four/lab-twitter-playpause/labFour.qmlproject

(c) 2011 Nokia Corporation and its Subsidiary(-ies).