Reusable QML Components

Avnee Nathani
@AvneeNathani
About me…

- Avnee Nathani
- Post-graduate student of Computer applications
- Open-source enthusiast
- Nokia Developer Champion (2012, 2013)
- Qt Ambassador (2011)

What do I work on?

- Qt Projects (github.com/AvneeNathani)
- KDE Games (KDiamond)
- Mobile applications (Windows Phone, Android)
- Research on Handwriting recognition (http://avnee.wordpress.com)
Agenda

- QML application development in brief
- QML basic components
- Need for custom QML components
- Custom components in KDiamond game
- Demo – Creating custom UI components
- Writing QML components
- ‘Reusability’ is the key when designing
- Importing Reusable components
- Conclusion
What does Qt Quick offer?

- **Qt Quick = QML UI Creation Kit**
- New way to create Qt User Interfaces
- QML works on UI, Qt helps with backend code implementation
- Declarative language: “QML” - CSS & JavaScript like syntax
- Supports states and transitions
QML Basic Components

- Rectangle (basic component, holds other components)
- Image
- BorderImage
- Text
- ListView

These are some of the primary components. Complex components are built by grouping primary components e.g. List with attributes such as icon, name and description.
QML Basic Components - Rectangle

Rectangle {
    width: 100
    height: 100
    color: "red"
    border.color: "black"
    border.width: 5
    radius: 10
}

Shows the effects of some of the common properties on a Rectangle item, which in this case is used to create a square
An unscaled image is displayed using an Image.

A BorderImage is used to display the image, and it is given a size that is larger than the original image.
QML Basic Components - Text

Text items can display both plain and rich text.

Red text with a specific font and size

```
Text {
    text: "Hello World!"
    font.family: "Helvetica"
    font.pointSize: 24
    color: "red"
}
```

Rich text is defined using HTML-style markup

```
Text {
    text: "<b>Hello</b> <i>World!</i>"
}
```
A ListView has a **model**, which defines the data to be displayed, and a **delegate**, which defines how the data should be displayed.

Displays data from models created from built-in QML types like ListModel and XmlListModel, or custom model classes defined in C++

**ListView with Model and Delegate**

```qml
ListView {
    width: 180; height: 200
    model: contactModel
    delegate: Text {
        text: name + ": " + number
    }
}
```

**ListModel for the list view - ListModel**

```qml
ListModel {
    id: contactModel
    ListElement {
        name: "Avnee"
        number: "876 5432"
    }
    ListElement {
        name: "Neha"
        number: "345 8426"
    }
    ...
}
```
Need for custom QML components

What is a custom QML component?

A custom component is essentially a QML document that defines a single QML component that you can use in an application.

Even though Qt Quick 2.0 comes with lots of useful controls right out of the box, you will likely come to a point where you must create your own custom components.

For example, in your QML applications, you may need to use custom Dialog boxes, prompts, toasts, notifications, buttons, labels, etc.

**Custom** – in terms of look and feel

**Custom** – in terms of functionality
Custom components in KDiamond

KDiamond is a single player puzzle game. The object of the game is to build lines of three similar diamonds.

Some of the custom QML components used,

- Buttons
- List Dialog
- Popup Dialog
- Quit Dialog
Custom components in KDiamond

**List Dialog:** A list dialog provides a dialog box with a list of items to choose from. Basically, a dialog that prompts for one element out of a list of elements.

![List Dialog in KDiamond](image.png)

**Fig: ListDialog for selecting Difficulty Level**

![Screenshot of ListDialog in KDiamond](image.png)

**Fig: Screenshot of ListDialog in KDiamond**
Custom components in KDiamond

**Popup Dialog:** A popup is used to show small info or notifications to the user.

![Screenshot of pause game popup in KDiamond](image-url)
Custom components in KDiamond

**Quit Dialog:** Quit Dialog is a component which is used to ask for confirmation before exiting an application or a game.

![Fig: Using quit dialog in KDiamond](image)
Demo – Creating custom UI components

Lets create some simple custom UI components!

- Custom Line Edit
- Custom Button
- ListDialog from KDiamond
Custom Button

Basic functionality,

- Triggers an event on clicking
- Allows a text caption

Image{
  id: buttonbase
  width: caption.width + 10
  height: caption.height + 20
  signal clicked
  property bool pressed: mousearea.pressed
  property alias buttonText: caption.text
  property string buttonPressed: buttonbase.source
  property string buttonReleased : buttonbase.source
  source: pressed ? buttonPressed : buttonReleased
}

Text{
  id: caption
  color: "white"
  text: buttonText
  font.pointSize: 12
  font.bold: true
  anchors.centerIn: parent
}

MouseArea{
  id: mousearea
  anchors.fill: parent
  onClicked: parent.clicked()
}
Writing QML components

Summarizing basic steps to create custom QML components

• Create a .qml for the component. File name needs to start with a capital letter. For e.g. ‘Button.qml’
• Define the component UI
• Define the interactions

Import and use this component in your applications.

Make them reusable…
Reusability is the key

If everyone keeps writing the same components, there would be too many versions of such components.

• Expose the component properties using property alias. This allows customization of components. For example, allow to change the color, background image, text caption of a Button component by property alias.

• Make your components public
Share the components with the community, put them on github if possible. Others can fork them and make them more versatile.

Remember, reusability is the key when writing custom components.
Questions?