Qt Quick Composing UIs

Qt Essentials - Training Course

Produced by Nokia, Qt Development Frameworks

Material based on Qt 4.7, created on January 18, 2011

http://qt.nokia.com

Module: Composing User Interfaces

- Nested Elements
- Graphical Elements
- Text Elements
- Anchor Layout

Objectives

- Elements are often nested
  - one element contains others
  - manage collections of elements
- Colors, gradients and images
  - create appealing UIs
- Text
  - displaying text
  - handling text input
- Anchors and alignment
  - allow elements to be placed in an intuitive way
  - maintain spatial relationships between elements

Module: Composing User Interfaces

- Nested Elements
- Graphical Elements
- Text Elements
- Anchor Layout

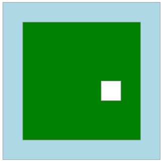
Nested Elements

```
import QtQuick 1.0

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

    Rectangle {
        x: 50; y: 50; width: 300; height: 300
        color: "green"

        Rectangle {
            x: 200; y: 150; width: 50; height: 50
            color: "white"
        }
    }
}
```



- Nested `Rectangle` elements
- Each element positioned relative to its parents

Demo qml-composing-uis/ex-elements/nested2.qml

Module: Composing User Interfaces

- Nested Elements
- Graphical Elements
- Text Elements
- Anchor Layout

Colors

The colors of elements can be specified in many ways:

- As a named color in a string (using SVG names):
  - `"red"`, `"green"`, `"blue"`, ...
- With color components in a string:
  - red, green and blue: `#<rr><gg><bb>`
  - `"#ff0000"`, `"#008000"`, `"#0000ff"`, ...
- Using a built-in function (red, green, blue, alpha):
  - `Qt.rgba(0,0.5,0,1)`
- With an opacity:
  - using the `opacity` property
  - values from `0.0` (transparent) to `1.0` (opaque)

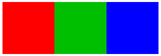
See QML Basic Type: color Documentation

Colors

```
import QtQuick 1.0

Item {
  width: 300; height: 100

  Rectangle {
    x: 0; y: 0; width: 100; height: 100; color: "#ff0000"
  }
  Rectangle {
    x: 100; y: 0; width: 100; height: 100
    color: Qt.rgba(0,0.75,0,1)
  }
  Rectangle {
    x: 200; y: 0; width: 100; height: 100; color: "blue"
  }
}
```



- Three different ways to specify colors

Demo qml-composing-uis/ex-elements/colors.qml

Images

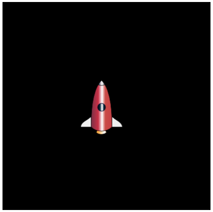
- Represented by the `Image` element
- Refer to image files with the `source` property
  - using absolute URLs
  - or relative to the QML file
- Can be transformed
  - scaled, rotated
  - about an axis or central point

Images

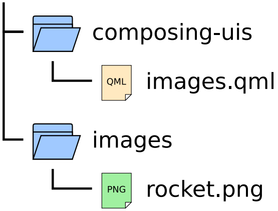
```
import QtQuick 1.0

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

    Image {
        x: 150; y: 150
        source: "../images/rocket.png"
    }
}
```



- source contains a relative path
  - "../" refers to the parent directory
- width and height are obtained from the image file



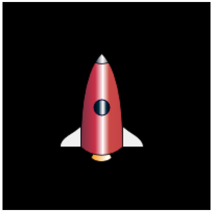
Demo qml-composing-uis/ex-elements/images.qml

Image Scaling

```
import QtQuick 1.0

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

    Image {
        x: 150; y: 150
        source: "../images/rocket.png"
        scale: 2.0
    }
}
```



- Set the `scale` property
- By default, the center of the item remains in the same place

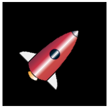
Demo qml-composing-uis/ex-elements/image-scaling.qml

Image Rotation

```
import QtQuick 1.0

Rectangle {
    width: 200; height: 200
    color: "black"

    Image {
        x: 50; y: 35
        source: "../images/rocket.svg"
        rotation: 45.0
    }
}
```



- Set the `rotate` property
- By default, the center of the item remains in the same place

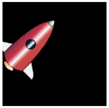
Demo qml-composing-uis/ex-elements/image-rotation.qml

Image Rotation

```
import QtQuick 1.0

Rectangle {
    width: 200; height: 200
    color: "black"

    Image {
        x: 50; y: 35
        source: "../images/rocket.svg"
        rotation: 45.0
        transformOrigin: Item.Top
    }
}
```



- Set the `transformOrigin` property
- Now the image rotates about the top of the item

Gradients

Define a gradient using the `gradient` property:

- With a `Gradient` element as the value
- Containing two or more `GradientStop` elements, each with
  - a position: a number between 0 (start point) and 1 (end point)
  - a color
- The start and end points
  - are on the top and bottom edges of the item
  - cannot be repositioned
- Issues with gradients:
  - rendering is CPU intensive
  - gradients may not be animated as you expect
  - use images of gradients instead
- Gradients override `color` definitions

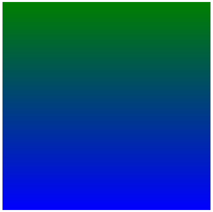
See QML Gradient Element Reference Documentation

Gradients

```
import QtQuick 1.0

Rectangle {
    width: 400; height: 400

    gradient: Gradient {
        GradientStop {
            position: 0.0; color: "green"
        }
        GradientStop {
            position: 1.0; color: "blue"
        }
    }
}
```



- A gradient with two gradient stops
- Note the definition of an element as a property value
- It is often faster to use images instead

Demo qml-composing-uis/ex-elements/gradients.qml

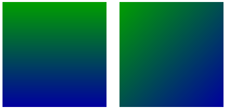
Gradient Images

```
import QtQuick 1.0

Rectangle {
    width: 425; height: 200

    Image {
        x: 0; y: 0
        source: "../images/vertical-gradient.png"
    }

    Image {
        x: 225; y: 0
        source: "../images/diagonal-gradient.png"
    }
}
```



- Use two-predefined images of gradients
- Artists can create the desired gradients

Demo qml-composing-uis/ex-elements/image-gradients.qml

Module: Composing User Interfaces

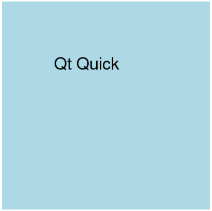
- Nested Elements
- Graphical Elements
- Text Elements
- Anchor Layout

Text Elements

```
import QtQuick 1.0

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

    Text {
        x: 100; y: 100
        text: "Qt Quick"
        font.family: "Helvetica"
        font.pixelSize: 32
    }
}
```



- Simple text display
- Width and height determined by the font metrics and text
- Can also use rich text
  - use HTML tags in the text: `"<html><b>Qt Quick</b></html>"`

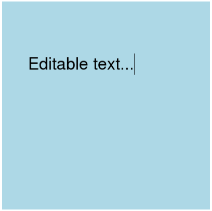
Demo qml-composing-uis/ex-elements/text.qml

Text Input

```
import QtQuick 1.0

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

    TextInput {
        x: 50; y: 100; width: 300
        text: "Editable text"
        font.family: "Helvetica"; font.pixelSize: 32
    }
}
```



- Simple editable text item
  - no decoration (not a `QLineEdit` widget)
- Gets the focus when clicked
  - need something to click on
- `text` property changes as the user enters text

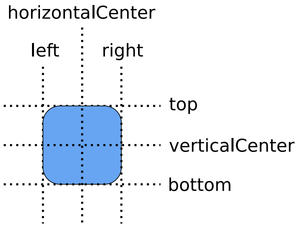
Demo qml-composing-uis/ex-elements/textinput.qml

Module: Composing User Interfaces

- Nested Elements
- Graphical Elements
- Text Elements
- Anchor Layout

Anchors

- Used to position and align items
- Line up the edges or central lines of items
- Anchors refer to
  - other items (`centerIn`, `fill`)
  - anchors of other items (`left`, `top`)



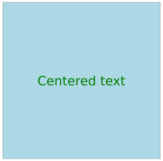
See Anchor-based Layout Documentation

Anchors

```
import QtQuick 1.0

Rectangle {
    width: 400; height: 400
    color: "lightblue"
    id: rectangle1

    Text {
        text: "Centered text"; color: "green"
        font.family: "Helvetica"; font.pixelSize: 32
        anchors.centerIn: rectangle1
    }
}
```



- `anchors.centerIn` centers the `Text` element in the `Rectangle`
  - refers to an item not an anchor

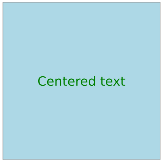
Demo qml-composing-uis/ex-anchor-layout/anchors.qml

Anchors

```
import QtQuick 1.0

Rectangle {
    // The parent element
    width: 400; height: 400
    color: "lightblue"

    Text {
        text: "Centered text"; color: "green"
        font.family: "Helvetica"; font.pixelSize: 32
        anchors.centerIn: parent
    }
}
```



- Each element can refer to its parent element
  - using the `parent` ID
- Can refer to ancestors and named children of ancestors

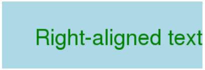
Demo qml-composing-uis/ex-anchor-layout/anchors2.qml

Anchors

```
import QtQuick 1.0

Rectangle {
    width: 300; height: 100
    color: "lightblue"

    Text {
        y: 34
        text: "Right-aligned text"; color: "green"
        font.family: "Helvetica"; font.pixelSize: 32
        anchors.right: parent.right
    }
}
```

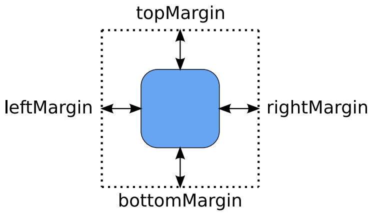


- Connecting anchors together
- Anchors of other items are referred to directly
  - use `parent.right`
  - not `parent.anchors.right`

Demo qml-composing-uis/ex-anchor-layout/anchor-to-anchor.qml

Margins

- Used with anchors to add space
- Specify distances
    - in pixels
    - between elements connected with anchors



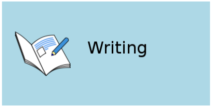
Margins

```
import QtQuick 1.0

Rectangle {
  width: 400; height: 200
  color: "lightblue"

  Image { id: book; source: "../images/book.svg"
          anchors.left: parent.left
          anchors.leftMargin: parent.width/16
          anchors.verticalCenter: parent.verticalCenter }

  Text { text: "Writing"; font.pixelSize: 32
         anchors.left: book.right
         anchors.leftMargin: 32
         anchors.baseline: book.verticalCenter }
}
```



- Use margins to add space between items

Demo qml-composing-uis/ex-anchor-layout/alignment.qml

Hints and Tips – Anchors

- Anchors can only be used with parent and sibling items
- Anchors work on constraints
  - some items need to have well-defined positions and sizes
  - items without default sizes should be anchored to fixed or well-defined items
- Anchors creates dependencies on geometries of other items
  - creates an order in which geometries are calculated
  - avoid creating circular dependencies
    - e.g., parent $\rightarrow$ child $\rightarrow$ parent
- Margins are only used if the corresponding anchors are used
  - e.g., `leftMargin` needs `left` to be defined

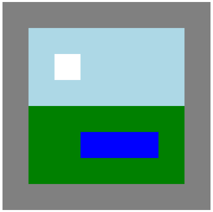
Strategies for Use – Anchors

Identify item with different roles in the user interface:

- Fixed items
  - make sure these have `id` properties defined
  - unless these items can easily be referenced as parent items
- Items that dominate the user interface
  - make sure these have `id` properties defined
- Items that react to size changes of the dominant items
  - give these anchors that refer to the dominant or fixed items

Exercise – Items

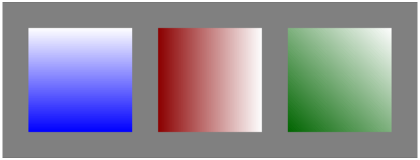
The image on the right shows two items and two child items inside a 400 × 400 rectangle.



1. Recreate the scene using `Rectangle` items.
2. Can items overlap?
   Experiment by moving the light blue or green rectangles.
3. Can child items be displayed outside their parents?
   Experiment by giving one of the child items negative coordinates.

Exercise – Colors and Gradients

1. How else can you write these colors?
   - `"blue"`
   - `"#ff0000"`
   - `Qt.rgba(0,0.5,0,1)`
2. How would you create these items using the `gradient` property?

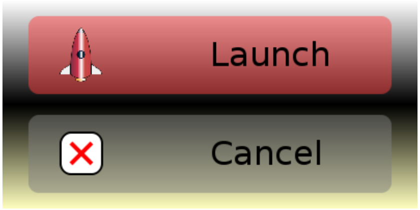


3. Describe another way to create these gradients?

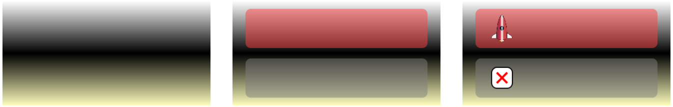
Exercise – Images and Text

1. When creating an `Image`, how do you specify the location of the image file?
2. By default, images are rotated about a point inside the image. Where is this point?
3. How do you change the text in a `Text` element?

Lab – Images, Text and Anchors



- Using the partial solutions as hints, create a user interface similar to the one shown above.
- Use the background image supplied in the common `images` directory for the background gradient.



Lab qml-composing-uis/lab-text-images-anchors