搜索

创建Qt快速应用程序

本教程说明了Qt Quick的基本概念。有关您拥有的 UI 选项的详细信息，请参阅用户界面。

本教程描述了当使用Qt 6作为最低Qt版本和CMake作为构建系统时，如何使用Qt Creator来实现状态和转换。

我们使用编辑模式创建一个应用程序，当您单击三个矩形时，该应用程序会在三个矩形之间移动Qt徽标。

有关更多示例，请参阅Qt 快速示例和教程。

您也可以在Qt Design Studio中开发Qt Quick应用程序。有关更多信息，请参阅Qt Design Studio 手册。

创建项目

1. 选择**文件**>**新建项目**>**应用程序 （Qt）**>**Qt 快速应用程序**。
2. 选择"选择"以打开"**项目位置**"对话框。
3. 在"**名称**"字段中，输入应用程序的名称。在命名自己的项目时，请记住，以后无法轻松重命名它们。
4. 在"**创建位置**"字段中，输入项目文件的路径。您可以稍后毫无问题地移动项目文件夹。
5. 选择**下一步**（或在 macOS 上**继续**）以打开定义**构建系统**对话框。
6. 在"生成系统"字段中，选择"CMake"作为用于生成和运行项目的**生成系统**。

> **注意：** 如果选择qmake，则配置项目的说明将不适用。

8. 在"**所需的最低 Qt 版本**"字段中，选择 Qt 6.2。
9. 选择"**下一步**"以打开"**翻译文件**"对话框。
10. 选择"**下一步**"以使用默认设置并打开"**套件选择**"对话框。
11. 为要为其构建应用程序的平台选择 Qt 6.2 或更高版本的工具包。要为移动设备构建应用程序，请同时选择适用于 Android 和 iOS 的工具包。

    > **注意：**如果在**Edit**>Preferences>Kits（在 Windows 和 Linux 上）或**Qt Creator**>**Preferences**>**Kits**（在 macOS 上）中指定了工具包，则会列出它们。有关详细信息，请参阅添加工具包。

12. 选择"**下一步**"以打开"**项目管理**"对话框。
13. 查看项目设置，然后选择"**完成**"（或在 macOS 上选择"**完成**"）以创建项目。

有关跳过的设置和其他可用的向导模板的更多信息，请参阅创建Qt快速应用程序。

Qt Creator生成一个组件文件*main.qml*，并在**编辑**模式下打开它。

部署应用程序

应用程序的主视图在视图左上角的矩形和两个空矩形内显示Qt徽标。

我们在本教程中使用*qt-logo.png*图像，但您也可以使用任何其他图像或组件。

若要在运行应用程序时显示映像，必须在向导为您创建的*CMakeLists.txt*文件的部分中将其指定为资源：
RESOURCES

```
qt_add_qml_module(apptransitions
    URI transitions
    VERSION 1.0
    QML_FILES main.qml Page.qml
    RESOURCES qt-logo.png
)
```

创建自定义 QML 类型

由于WindowQML 类型要求将状态添加到子组件中，因此我们使用向导创建一个名为*Page*的自定义 QML 类型，我们将从*main.qml*引用该类型。

要创建自定义 QML 类型，请执行以下操作：

1. 选择**文件**>**新建文件**>Qt>**QML 文件（Qt Quick 2）**。
2. 选择"**选择**"以打开"**位置**"对话框。
3. 在**文件名**字段中，输入自定义 QML 类型的名称。在此示例中，我们调用类型*页面*。
4. 选择"**下一步**"以打开"**项目管理**"对话框。
5. 选择"**完成**"以创建*Page.qml*。

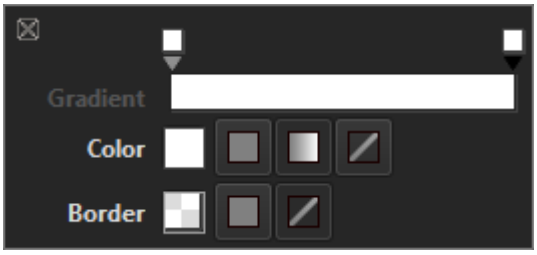
Qt Creator在**编辑**模式下打开*Page.qml*。它包含我们用矩形类型替换的Item类型的根项。我们为类型提供 ID*页面*，将其锚定到所有侧面的父项，并将其颜色设置为白色：

```
Rectangle {
    id: page
    anchors.fill: parent
    color: "#ffffff"
```

因为我们使用 Qt 6 进行开发，其中版本号不与模块一起使用，所以我们从 import 语句中删除版本号。

当您开始键入QML类型名称时，Qt Creator会建议可用的类型和属性来完成代码。

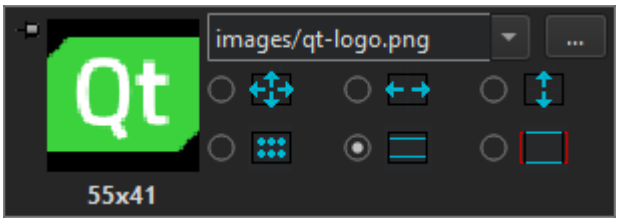
选择类型名称旁边的灯泡图标以打开矩形的 Qt 快速工具栏。可以使用它来指定矩形属性，如颜色、透明度和渐变。



接下来，我们添加一个以*qt-logo.png*作为源的图像类型。我们将图像放置在矩形的左上角：

```
    Image {
        id: icon
        x: 20
        y: 20
        source: "qt-logo.png"
    }
```

您可以使用图像的 Qt 快速工具栏来指定图像属性，例如源文件和填充模式。



现在，我们创建图像将在其间移动的矩形。它们的大小应与图像大小匹配，并且应该是透明的，以便图像可见。我们将边框颜色设置为浅灰色以使矩形可见：

```
    Rectangle {
        id: topLeftRect
        width: 55
        height: 41
        color: "#00ffffff"
        border.color: "#808080"
```

```
        anchors.left: parent.left
        anchors.top: parent.top
        anchors.leftMargin: 20
        anchors.topMargin: 20
```

我们添加了一个鼠标区域类型，使矩形可被用户单击：

```
        MouseArea {
            id: mouseArea
            anchors.fill: parent
```

若要检查代码，可以将其与*Page.qml*示例文件进行比较。

接下来，我们将通过添加状态并将鼠标单击连接到状态更改，使图像在用户单击矩形时在矩形之间移动。

将鼠标单击连接到状态更改

为了使图像在用户单击矩形时在矩形之间移动，我们向 Page 组件添加状态，在其中我们更改*图标*的 andproperties 的值以匹配中间右边和左上角矩形的值。为了确保在不同大小的屏幕上缩放视图时图像显示在矩形内，我们将*图标*的属性值*绑定*到矩形的值：xyxy

```
    ...
    states: [
        State {
            name: "State1"
        },
        State {
            name: "State2"

            PropertyChanges {
                target: icon
                x: middleRightRect.x
                y: middleRightRect.y
            }
        },
        State {
            name: "State3"

            PropertyChanges {
                target: icon
                x: bottomLeftRect.x
                y: bottomLeftRect.y
            }
        }
    ]
```

然后，我们将鼠标区域的信号连接到状态更改 [illegible]

```
            Connections {
                target: mouseArea
                function onClicked()
                {
                    page.state = "State1"
                }
```

因为我们使用 Qt 6 进行开发，所以我们必须将连接指定为函数。

将页面添加到主视图

现在，我们打开*main.qml*进行编辑，并向其中添加页面自定义组件的实例：

```
import QtQuick

Window {
    width: 640
    height: 480
    visible: true
    title: qsTr("Transitions")

    Page {
        id: page
        anchors.fill: parent
    }
}
```

按**Ctrl+R**运行应用程序，然后单击矩形将 Qt 徽标从一个矩形移动到另一个矩形。

动画过渡

现在，我们将创建过渡以将动画应用于图像。例如，图像在移动到*middleRightRect*并缓和进入*bottomLeftRect*时会反弹。

我们指定从每种状态切换到其他两种状态的转换：

```
    transitions: [
        Transition {
            id: toState1
            ParallelAnimation {
```

Qt创建者手册8.0.2
Topics

```
                }

                PropertyAnimation {
                    target: icon
                    property: "y"
                    duration: 200
```

```
            SequentialAnimation {
                PauseAnimation {
                    duration: 0
                }

                PropertyAnimation {
                    target: icon
                    property: "x"
                    duration: 200
                }
            }
        }
        to: "State1"
        from: "State2,State3"
    },
```

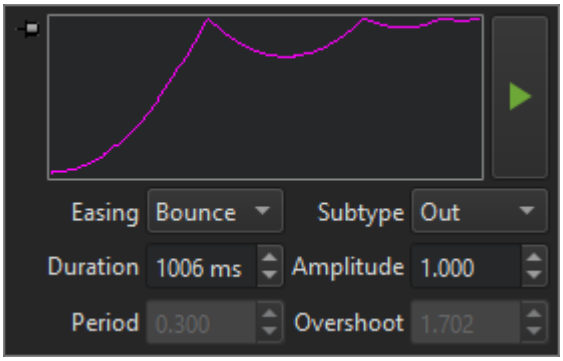
我们将过渡的缓动曲线类型从线性更改为*State2*，以创建反弹效果：`Easing.OutBounce`

```
    Transition {
        id: toState2
        ParallelAnimation {
            SequentialAnimation {
                PauseAnimation {
                    duration: 0
                }

                PropertyAnimation {
                    target: icon
                    property: "y"
                    easing.type: Easing.OutBounce
                    duration: 1006
                }
            }

            SequentialAnimation {
                PauseAnimation {
                    duration: 0
                }

                PropertyAnimation {
                    target: icon
                    property: "x"
                    easing.type: Easing.OutBounce
                    duration: 1006
                }
            }
        }
        to: "State2"
        from: "State1,State3"
    },
```



然后，我们将过渡到 State2 的缓动曲线类型从线性更改为*State2*，以创建缓动效果：`Easing.InOutQuad`

```
        Transition {
            id: toState3
            ParallelAnimation {
                SequentialAnimation {
                    PauseAnimation {
                        duration: 0
                    }

                    PropertyAnimation {
                        target: icon
                        property: "y"
                        easing.type: Easing.InOutQuad
                        duration: 2000
                    }
                }

                SequentialAnimation {
                    PauseAnimation {
                        duration: 0
                    }

                    PropertyAnimation {
                        target: icon
                        property: "x"
                        easing.type: Easing.InOutQuad
                        duration: 2000
                    }
                }
            }
            to: "State3"
            from: "State1,State2"
        }
    ]
}
```

按Ctrl+R运行应用程序，然后单击矩形以查看动画过渡。

文件：

- transitions/main.qml

图像:

- transitions/qt-logo.png

‹ 教程 创建基于 Qt Widget 的应用程序 ›

©2022 Qt有限公司 此处包含的文档贡献的版权归 他们各自的所有者。此处提供的文档根据自由软件基金会发布的GNU 自由文档许可证版本 1.3的条款进行许可。Qt和相应的徽标是Qt有限公司在芬兰和/或全球其他国家的商标。所有其他商标均为财产 其各自所有者。

The Qt Company

联系我们

公司

关于我们
投资者
编辑部
职业
办公地点

发 牌

条款和条件
开源
常见问题

支持

支持服务
专业服务
合作 伙伴
训练

对于客户

支持中心
下载
Qt登录
联系我们
客户成功案例

社区

为Qt做贡献
论坛
维基
下载
市场

Qt DOCUMENTATION

© 2022 Qt公司