Transparent QML Apps

This code example shows how to create translucent Qt Quick apps for both mobile and desktop operating systems.

11 Mar 2012

Overview

QML elements are usually rendered by Qt by QDeclarativeView. You can ensure that an app's background is painted with non-opaque colors using the inherited QWidget method QWidget::setAttribute(Qt::WA_TranslucentBackground).

There is some platform-specific implementation:

- X11: This feature relies on the use of an X server that supports ARGB visuals and a compositing window manager.
- Windows: The view needs to have the `Qt::FramelessWindowHint` window flag set for the translucency to work. Ie you need to call `setWindowFlags(Qt::FramelessWindowHint)`.

Implementation

Source (.cpp)

The following code shows the minimum implementation for a translucent QDeclarativeView that works on Linux/X11 systems, Windows and Symbian.

```
int main(int argc, char *argv[]){
QApplication a(argc, argv);

QDeclarativeView view;
view.setSource(QUrl::fromLocalFile("myqmlfile.qml"));
view.setAttribute(Qt::WA_TranslucentBackground);
view.viewport()->setAutoFillBackground(false);
#ifdef Q_WS_WIN
 view.setWindowFlags(Qt::FramelessWindowHint);
#endif

view.show();
return a.exec();
}
```

When working on a "standard" Qt Quick Application created using the project wizard in the Qt SDK v1.2 the `QDeclarativeView` is wrapped in the convenience class `QmlApplicationViewer`. The code can be copy-pasted into the template with only trivial object name changes, as shown below:

```
Q_DECL_EXPORT int main(int argc, char *argv[])
{
    QScopedPointer<QApplication> app(createApplication(argc, argv));

    QmlApplicationViewer viewer;
    viewer.setOrientation(QmlApplicationViewer::ScreenOrientationAuto);
    viewer.setMainQmlFile(QLatin1String("qml/TransparentQtQuickApps/main.qml"));

    //Copy code below to make app translucent
#ifdef Q_WS_WIN
    viewer.setWindowFlags(Qt::FramelessWindowHint);
#endif
    viewer.setAttribute(Qt::WA_TranslucentBackground);
    viewer.viewport()->setAutoFillBackground(false);
    // Copy code above
```

```
    viewer.showExpanded();
    return app->exec();
}
```

QML

Tip: Remember that QML visual elements are drawn opaque by default; there is no point making your app view transparent if your whole QML app is opaque!

The code for the QML must have at least some areas with an opacity less than 1. In the example below you can see that the rectangle onto which the text is drawn has an opacity of 0.5.

```
Rectangle {
    width: 360
    height: 360
    opacity:0.5 //note opacity<1
    color: "lightgrey"
    Text {
        text: qsTr("Hello\nWorld")
        font.pointSize : 30

        anchors.centerIn: parent
    }
    MouseArea {
        anchors.fill: parent
        onClicked: {
            Qt.quit();
        }
    }
}
```

Screenshot

```
2  import QtQuick 1.1
3
4  Rectangle {
5      width: 360
6      height: 360
7      opacity:0.5
8      color: "lightgrey"
9      Text {
10         text: qsTr("Hello\nWorld")
11         font.pointSize : 30
12
13         anchors.centerIn: parent
14     }
15     MouseArea {
16         anchors.fill: parent
17         onClicked: {
18             Qt.quit();
19         }
20     }
21 }
22
```

Hello
World

Translucent QML app on Windows

Conclusion

QML is quite powerful and it's a great thing it support translucent windows even if it's not supported at same way by all Qt

supported platforms.