Workaround to hide VKB in QML apps (Known Issue)

This article explains how to close the virtual keyboard in pure QML apps.

Introduction

Pure QML applications on MeeGo Harmattan have an annoying bug in which the Virtual Keyboard (VKB) is not closed when you tap outside a selected text editing element.

What happens is that when a QML element that is able to listen for key events receives the focus, a virtual keyboard is displayed that the user can use for entering text. When the user taps outside the element to indicate they have finished entering text the element doesn't actually lose the focus, so the keyboard is not closed. This goes against the Harmattan style guidelines (and common sense).

This article describes a number of workarounds to the problem.

Note: This bug has was reported as fixed in Harmattan > PR 1.1. Note however that more recently it has been reported as still valid on N9 handset, version PR1.2 (30.2012.07-1.348.2_PR_348) - compiling with Qt Creator 2.4.1, based on Qt 4.7.4, part of Nokia SDK as released in April 2012. The workaround is effective on that version.

Pure QML Solution

One workaround is to use closeSoftwareInputPanel QML method. The example below is the implementation of this fix inside a simple QML app made up of a root `Rectangle` element and a `TextEdit`. The virtual keyboard is shown when the `TextEdit` is clicked, but the virtual keyboard is not hidden when the user taps outside.

The main problem here is related to the focus. Tapping outside the `TextEdit` element is not enough to change the focus and close the VKB, so we have to add some additional code. The fix is implemented basically in two parts:

- `MouseArea` - a big and transparent to events `MousArea` overlays the entire application. This allows us to give the focus to the clicked item
- `onActiveFocusChanged` slot inside the any element able to receive text input. This slot actually closes the VKB

The rest of example is pretty simple and the complete code can be seen below:

```
import QtQuick 1.0
    Rectangle {
      id: root
      width: 800
      height: 600
      focus: true
        Rectangle {
          width: 360
          height: 360
          color: "cyan"
          TextEdit {
               id: mText
               anchors.fill: parent
                 onActiveFocusChanged: {
                   if (!mText.activeFocus) mText.closeSoftwareInputPanel();
               }
           }
      }
       MouseArea {
          anchors.fill: parent
            onPressed: {
              // Let the mousePress events go to below items
              mouse.accepted = false;
                // Clicked item below the MouseArea gets the focus
```


(C) Copyright Nokia 2014. All rights reserved.

```
                var selectedItem = root.childAt(mouse.x, mouse.y);
                if (!selectedItem) selectedItem = root;
                selectedItem.focus = true;
            }
        }
    }
```

C++ workaround [BETTER SOLUTION]

A pure C++ workaround consists of adding an event filter to the affected view instance:

```
class EventFilter : public QObject
{
protected:
    bool eventFilter(QObject *obj, QEvent *event) {
        QInputContext *ic = qApp->inputContext();
        if (ic) {
            if (ic->focusWidget() == 0 && prevFocusWidget) {
                QEvent closeSIPEvent(QEvent::CloseSoftwareInputPanel);
                ic->filterEvent(&closeSIPEvent);
            } else if (prevFocusWidget == 0 && ic->focusWidget()) {
                QEvent openSIPEvent(QEvent::RequestSoftwareInputPanel);
                ic->filterEvent(&openSIPEvent);
            }
            prevFocusWidget = ic->focusWidget();
        }
        return QObject::eventFilter(obj,event);
    }

private:
    QWidget *prevFocusWidget;
};
```

This filter can be installed on a QGraphicsView or (in QML) on a QDeclarativeView instance:

```
EventFilter ef;
view.installEventFilter(&ef);
```

MeeGo Components are not affected!

While this bug affects plain QML apps, MeeGo component based apps are not affected! So if possible you should use the Meego elements set instead of the standard QML ones. The only one drawback here is the compatibility with Symbian set. In fact, if your target is to run the app both on Symbian and MeeGo polatforms then the second solution shown above is the right way to go.

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.