helloQt is the Qt development
branch of Bergx2 GmbH

Fürstenstr. 15
80333 München
Germany

info@helloqt.de
www.helloqt.de

Implementing a timer-based action in QML with Qt

11 February 2025, by Lucas Moreira de Oliveira

The following use case is based on Qt application which manages digital signage displays. The QML Timer component controls the cursor activation based on user activity. This component allows developers to schedule actions after time delay or at regular intervals.

In this blog post, we will show how to use the Timer component to automatically hide the cursor in a digital signage application we developed at Begx2 GmbH. The goal is to hide the cursor after 15 seconds of inactivity, keeping the screen clean and free from distractions.

Understanding the QML Timer

The Timer element in QML provides a straightforward way to handle timed events. It can be configured to trigger once after a delay or repeatedly at specified intervals. Here is a quick look at its key properties:

- Interval: Specifies the duration in milliseconds before the timer triggers.
- Running: A boolean that determines whether the timer is active.
- Repeat: If true, the timer runs continuously at the specified interval.
- OnTriggered: The event handler that executes when the timer fires.

Implementing cursor deactivation on inactivity

For our digital signage use case, we need a timer that resets every time the user interacts with the system. If no activity is detected for 15 seconds, the cursor disappears. Below is an implementation using QML and Qt's MouseArea component:

```
import QtQuick
import QtQuick.Layouts
import ui as UI

Window {
    id: _window
    property real globalScale: Math.min(_window.width / 1920, _window.height /
1080)
    visible: true
    visibility: UI.AppController.isAppAlwaysFullscreen ? Window.FullScreen :
Window.Windowed
    title: qsTr("ScreenWay Media Player")
    width: Screen.width
    height: Screen.height
    color: "black"

    // 1) Properties for hiding/showing cursor
    property bool mouseHidden: true   // We want the cursor hidden by default
    property real lastMouseX: 0
    property real lastMouseY: 0

    // 2) Timer for inactivity (15 seconds)
    Timer {
        id: inactivityTimer
        interval: 15000  // 15 seconds
        repeat: false
        running: false
        onTriggered: {
            // Hide the cursor after 15s
            mouseHidden = true
        }
    }

    // 3) Full-screen MouseArea that tracks movement but does NOT accept 
clicks
    //    so that the existing MouseArea for settings popup still works.
    MouseArea {
        id: fullScreenTracker
        anchors.fill: parent
        hoverEnabled: true
        acceptedButtons: Qt.NoButton
        z: 0

        // Switch between blank and arrow cursor
        cursorShape: mouseHidden ? Qt.BlankCursor : Qt.ArrowCursor

        onPositionChanged: function(event) {
            // If user moves mouse ≥ 10 pixels in any direction, show the 
cursor
            var dx = event.x - lastMouseX
            var dy = event.y - lastMouseY
            if (Math.sqrt(dx * dx + dy * dy) >= 10) {
                if (mouseHidden) {
                    mouseHidden = false
                }
                // Restart the 15-second timer every time we detect 
significant movement
                inactivityTimer.restart()
            }
            lastMouseX = event.x
            lastMouseY = event.y
        }
    }
    MouseArea {
        x: parent.width - width
        z: 1
        width: UI.Constants.settingsOverlayOpenAreaSize
        height: UI.Constants.settingsOverlayOpenAreaSize

        scale: _window.globalScale

        hoverEnabled: true
        cursorShape: containsMouse ? Qt.PointingHandCursor : Qt.ArrowCursor

        onClicked: {
            _settings.open()
        }
    }

    Item {
        id: _rootItem
        rotation: UI.AppController.orientation
        anchors.fill: parent

        StackLayout {
            id: _screensLayout
            anchors.fill: parent
            currentIndex: UI.AppController.currentScreen

            UI.InitialScreen { }
            UI.PairingScreen { }
            UI.LoadScreen { }
            UI.PlayerScreen { }
            UI.WaitingScreen { }
        }

        UI.SettingsPopup {
            id: _settings
            x: (_window.width - width) * 0.5
            y: (_window.height - height) * 0.5
            z: 2
            scale: _window.globalScale
            onClosed: {
                _window.mouseHidden = false
            }
        }
    }

    Component.onCompleted: {
        UI.AppController.init();
        mouseHidden = true
        inactivityTimer.start()
    }
}
```

How it works

1. A MouseArea component detects user interactions such as clicks and presses.
2. Each interaction resets the timer using the resetTimer() function.
3. If no interaction occurs for 15 seconds, the onTriggered handler calls hideCursor(), making the cursor invisible.
4. Any subsequent interaction restarts the timer and reactivates the cursor.
5. Minor movements (due to environmental factors) are ignored using the checkMovement() function, which ensures only significant movements reset the timer.

Enhancements and considerations

- Keyboard activity: To further refine this feature, consider adding key event listeners to reset the timer on keyboard input.
- Visual indicator: If needed, a subtle fade-out effect can be added before hiding the cursor.
- User configuration: Allow users to configure the timeout duration in the application settings.

Conclusion

Using the Timer component in QML, we have successfully implemented an automatic cursor deactivation mechanism that improves the usability of a digital signage application. This approach can be extended to various other use cases requiring automated inactivity detection.

By leveraging Qt robust QML framework, you can efficiently handle timed events and enhance user interactions within your applications. The Qt team at helloQt will be happy to help you with the implementation of your Qt project.