QML Map extension

This page describes an extension to Qt Mobility QML Map element. It features map scrolling (flicking), zooming, landmark move-along.

Features

Flicking

The trick here is to use a Flickable on top of the Map.

The Flickable is very large in order to avoid bouncing on the walls.

The movementEnded signal is connected to reseting the position in the Flickable to its center *without moving the map*, which is obtained by locking the access to the panMap function.

The Flickable movements are connected to the map pan function.

Zooming

Zooming is rather easy. Simply adding two buttons, connecting a MouseArea to the Map zoom property.

Landmark move-along

As of now, I could not find a trivial way to have items drawn on the Map move as the map is panned. So, here is the trick used: the model used in the repeater is reset as the map is moved around.

Also, the map is displayed with a margin. This margin is used to make sure the elements from the list are displayed when they are slightly out of the screen (an item drawn out of its parent is not drawn). An item slightly out of the screen may still need to be drawn if the icon is large.

Use

3 elements are mandatory to use this map:

- a delegate to draw the items on the map
- a model for the delegate to be repeated on
- a receiver for the signal viewPortChanged, which is called when the zoom changes and when a movement on the map is complete. This signal is useful if your model contains elements based on the viewport.

Delegate

A basic delegate may look like:

```
Item {
    id: element
    signal selected(int index)
    x: map.toScreenPosition(coord).x + map.anchors.topMargin
    y: map.toScreenPosition(coord).y + map.anchors.leftMargin
    Image {
        id: bg
        anchors.horizontalCenter: parent.left
        anchors.bottom: parent.top
        source: "images/bg.png"
        MouseArea {
            anchors.fill: parent
            onClicked: {
                console.log("Landmark clicked")
                selected(index)
            }
        }
    }
}
```

In this example, the items in the model have a coord element that contains latitude and longitude.

Use example

```
Rectangle {
    id: root
    width: 360
    height: 360
```


(C) Copyright Nokia 2013. All rights reserved.

```
    ListModel {
        id: photosModel
    }
    MyMap {
        id: photosMap
        opacity: 0
        itemMapDelegate: ThumbItem {}
        itemsModel: photosModel
        onViewportChanged: loadImages(from, to)
    }
}
```

All you need is a loadImages method that puts images in the photosModel.

Code

Here is the code for the map

```
import QtQuick 1.0
import QtMobility.location 1.1

Item {
    id: myMapRoot
    property Component itemMapDelegate
    property variant itemsModel
    signal viewportChanged(variant from, variant to)
    anchors.fill: parent
    onOpacityChanged: {
        if (opacity == 1) {
            updateViewport();
        }
    }
    function updateViewport() {
        viewportChanged(
                    map.toCoordinate(Qt.point(-map.anchors.leftMargin,-map.anchors.topMargin)),
                    map.toCoordinate(Qt.point(map.size.width + map.anchors.rightMargin,
                                              map.size.height + map.anchors.bottomMargin)))
    }

    PositionSource {
        id: me
        active: true
        updateInterval: 1000
        onPositionChanged: console.log(position.coordinate)
    }
    Map {
        id: map
        anchors.fill: parent
        anchors.margins: -80
        zoomLevel: 4

        plugin: Plugin { name : "nokia" }
        center: me.position.coordinate

        onZoomLevelChanged: {
            myMapRoot.updateViewport()
        }
        onCenterChanged: {
            var tmp = pinpointView.model
            pinpointView.model = null
            pinpointView.model = tmp
        }
        onSizeChanged: {
            var tmp = pinpointView.model
            pinpointView.model = null
            pinpointView.model = tmp
        }
    }
    Flickable {
        id: flickable
        anchors.fill: parent
        contentWidth: 8000
        contentHeight: 8000

        Component.onCompleted: setCenter()
        onMovementEnded: {
            setCenter()
            myMapRoot.updateViewport()
        }
        function setCenter() {
            lock = true
            contentX = contentWidth / 2
            contentY = contentHeight / 2
            lock = false
            prevX = contentX
            prevY = contentY
        }

        onContentXChanged: panMap()
        onContentYChanged: panMap()
        property double prevX: 0
        property double prevY: 0
        property bool lock: false
        function panMap() {
            if (lock) return
            map.pan(contentX - prevX, contentY - prevY)
            prevX = contentX
            prevY = contentY
        }
    }
    ZoomButton {
        id: plus
        value: "+"
    }
    ZoomButton {
        id: minus
        value: "-"
        anchors.right: plus.left
    }
    MouseArea {
        anchors.fill: plus
```


(C) Copyright Nokia 2013. All rights reserved.

```
        onClicked: {
            map.zoomLevel += 1
        }
    }
    MouseArea {
        anchors.fill: minus
        onClicked: {
            map.zoomLevel -= 1
        }
    }
    Item {
        id: pinpointViewContainer
        Repeater {
            id: pinpointView
            model: itemsModel
            delegate: itemMapDelegate
        }
    }
}
```

The zoom button is a simple rectangle.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.