StarMenu: custom QML component plugin

This article introduces `StarMenu`, a QML pop up menu.

Introduction

The `StarMenu` is a QML contextual pop up menu that can be launched by clicking or tapping on a component like with `MouseArea` element. The `StarMenu`'s menu items are defined using `StarItems`. This plugin was developed with Qt 4.7.4 and tested on linux, windows, Nokia N900, Nokia C7-00, Nokia N950 and Android Galaxy tab GT-7310 (with Necesistas).

The media player is loading...

Usage

Note: The documentation to install and use the plugin is in File:StarMenu Project.zip.

`StarMenu` is hidden by default and opens in response to a user action. The purpose of `StarMenu` is to manage all `StarItem` childs. By default they are arrange in a circle shape and, in the center of it, it is painted a star polygon shape with as many spikes as the number of StarItems. It can be visible or not by setting the boolean `starCenterVisible` property. The circle shape amplitude is given by `radiusH` and `radiusV` properties.

Clicking on the center of the current menu, will close it and goes back to parent.

It's possible to click on the center of the first menu to close all.

`visualParent` property is used to bind the close menu action to a specified item, so, instead of clicking on center of `StarMenu`, it's possible to set `visualParent` to the main window and close menu everywhere on the screen.

When using `visualParent`, it come handy the `color` property to set a transparent color to the clickable area to close the menu. Changing `StarItem` `visible` property, makes all `StarItems` position recalculated with only visible items.

```
StarMenu {
  id: starMenu
  anchors.fill: parent
  radius: 130

  StarItem {
     width: 64; height: 64
     Image {
       anchors.fill: parent
       [...]
     }
  }
   StarItem {
   [...]
     StarMenu {
       [...]
     }
   }
}
```


(C) Copyright Nokia 2013. All rights reserved.

It is possible to bind animations to `StarItems`. When a `StarMenu` is raised, the StarItems circle shape has the center located by the mouse click (or tap) position and their opacity, rotation, x, y property are set to 0. By doing this:

```
StarItem {
 [...]
 Behavior on x { NumberAnimation { duration: 1500; easing.type: Easing.OutBounce } }
 Behavior on y { NumberAnimation { duration: 1500; easing.type: Easing.OutBounce } }
 Behavior on opacity { NumberAnimation { duration: 2200; easing.type: Easing.OutBounce } }
```

StarItems are animated from center to their final position.

StarMenu has also a `delayChangePos` property. For example, when it is set to 100, the animation of the second `StarItem` is started after 100 ms, the third after 200 ms and so on. `StarItem` has also a `delayChangePos` property and if it is set, it take priority over the `StarMenu delayChangePos` property.

`StarItems` can also contain any other component instead Image component used in these examples:

```
StarItem {
   width: 320; height: 240
   scale: 0.3
   zoomHoverValue: 1
   Behavior on scale { NumberAnimation { duration: 500; easing.type: Easing.OutQuad } }
   Column {
 Button { text: "Hello Qt" }
        Switch { }
        RadioButton { }
   }
}
```

`StarItems` can be arranged in a `spiral shape`, in a `grid shape` or in a `line shape`. `StarMenu shape` property could have the following values:

- `StarMenu.ShapeCircle` (default value)
- `StarMenu.ShapeGrid`
- `StarMenu.ShapeSpiral`
- `StarMenu.ShapeLine`

Rows in grid shape are arranged centered by the mouse click and are spaced by `spacingH` and `spacingV` properties with columns count defined by `columns` property. Spiral shape takes care of `startingRadius` value to start drawing StarItems and `radiusIncrement` property to shape the spiral. Line shape has a `startingAngle`, `radiusIncrement` and `startingRadius` properties.

StarMenu has the center where user clicks. This center can be defined everywhere in scene coordinates by setting `center` property.

The media player is loading...

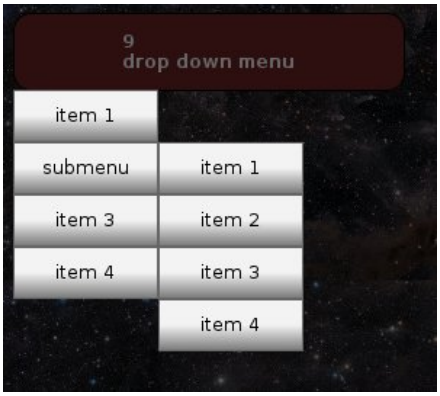
(C) Copyright Nokia 2013. All rights reserved.

Examples

Here some examples which you can find in the source code of File:StarMenu Project.zip in File:StarMenu Project.zip. There are also other useful links like File:Getting Started.txt , File:Overview.txt and a developer File:Components.txt.

How to make a drop down menu

This is the example number 9 (Menu9.qml) in File:StarMenu Project.zip. It will explain how to make a drop down menu with `StarMenu` plugin.



Lets start with a `Rectangle` component with `StarMenu` as a child:

```
Rectagle {
    id: mainButton
    property int itemsW: 100
    property int itemsH: 36
    width: 150; height: 30

    StarMenu {
        anchors.fill: parent
        starCenterVisible: false          // don't display the star polygon
        shape: StarMenu.ShapeLine   // define the shape that entries will assume
        visualParent: mainWindow    // set the sensible area to the root scene item
        color: Qt.rgba(0,0,0,0.6)         // set the color of the sensible area
        delayChangePos: 150            // entries are arranged with a delay
        radiusIncrement: 0                // the entries are arranded with no offset
        startingAngle: -90                 // the line shape direction
        startingRadius: 0                  // offset of the 1st entry
        center.x: mainButton.x+mainButton.itemsW/2
        center.y: mainButton.y+mainButton.height+mainButton.itemsH/2
    }
}
```

in this code we start to define `itemsW` and `itemsH` for the dimensions of the menu entries. `StarMenu` is anchored to `mainButton`, so when it is clicked, the menu will pop up. A plus to the comments in the above code, the `center` property needs an explaination. When `StarMenu` opens, all entries are placed where the user clicks. This behavior can be changed by setting a value to the `center` property. In this case, we want to rise the menu just under `mainButton`. Since this center is also the center of the 1st item, we calculate the x position to be aligned to the left of `mainButton` and y position to be at the bottom of `mainbutton` plus the mid height of 1st item.

Now lets start to add as many `StarItems` as you like:

```
StarItem {
    id: firstItem
    width: mainButton.itemsW; height: mainButton.itemsH
    scale: 0.9
    zoomHoverValue: 1
    Behavior on x  { NumberAnimation { easing.amplitude: 0.3; duration: 800; easing.type: Easing.OutBounce } }
    Behavior on y  { NumberAnimation { easing.amplitude: 0.3; duration: 800; easing.type: Easing.OutBounce } }
    Behavior on opacity  { NumberAnimation { duration: 1000 } }
    Behavior on scale     { NumberAnimation { duration: 200 } }

    Rectangle {
        anchors.fill: parent
        border { color: "#000000"; width: 1 }
        Text {
            anchors.fill: parent
            verticalAlignment: Text.AlignVCenter
            horizontalAlignment: Text.AlignHCenter
            text: "item 1"
        }
    }
}
```

As mentioned before, all menu entries are placed in center before the menu opens. So it's easy to animate the transition to the open state. Also `opacity`, `rotation` and {{Icode|scale} can be animated. In `StarMenu` component we set the `delayChangePos` to 150 (ms). If we want to override this value for a particular entry, we can just add to this `StarItem` its own `delayChangePos` property. Another nice feature of StarItems is the `zoomHoveValue` property. In this case, since there is the text that will be distorted with a scale > 1, we started with a `scale` of 0.9. When the mouse is hover the item, it will be scaled to 1 with an animation. The `StarItem` interaction with the user can be managed by these signals:

```
StarItem::hoverItem()
StarItem::leaveItem()
StarItem::clicked()
```


(C) Copyright Nokia 2013. All rights reserved.

while for `StarMenu` are available these:

```
StarMenu::starMenuOpening()
StarMenu::starMenuClosing()
```

Now, lets say we want a sub menu when clicking an entry. Just add a `StarMenu` component as a child of this entry:

```
StarItem {
 id: secondItem
 [...]
 Rectangle {
            [...]
         }
 StarMenu {
  anchors.fill: parent
                starCenterVisible: false
                shape: StarMenu.ShapeLine
                visualParent: mainWindow
                parentVisible: true
  center.x: mainButton.x+mainButton.itemsW+mainButton.itemsW/2
                center.y: secondItem.y+secondItem.height/2

  StarItem { ...}
  StarItem { ...}
  StarItem { ...}
 }
}
```

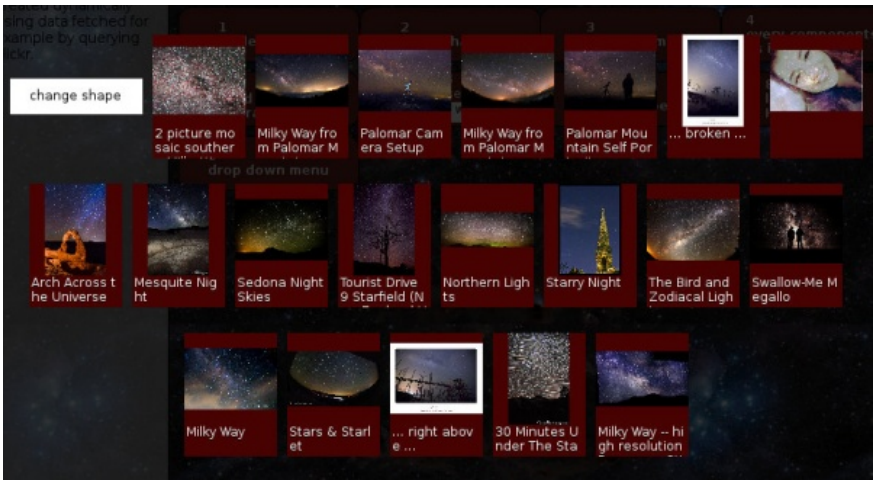
The only thing we should take care for this example, is the `center` property like explained before. Another thing we should consider is the `parentVisible` property. By default when a sub menu is opened, his parent is hidden. With this property set to true, it will stay visible.

About the opening action event by user click, this can be overridden by setting `open` property of `StarMenu` to `true`. So i.e. when the application starts, the menu can be already opened without the user click:

```
Component.onCompleted: starMenu.open = true
```

How to make a dynamic menu

This is the example number 6 (Menu6.qml) in File:StarMenu Project.zip. It will explain how to create menu entries dynamically with `StarMenu` plugin.



We need at first a XmlListModel to get the feed from Flickr:

```
XmlListModel {
    id: modelList
    property string tags

    function commasep(x)
    {
        return x.replace(' ',',');
    }

    query: "/rss/channel/item"
    namespaceDeclarations: "declare namespace media=\"http://search.yahoo.com/mrss/\";"

    XmlRole { name: "url"; query: "media:content/@url/string()" }

    onTagsChanged: {
        if (tags !== "")
            source = "http://api.flickr.com/services/feeds/photos_public.gne?"+(tags ? "tags="+commasep(tags)+"&" : "")+"format=rss2
    }

    onStatusChanged: {
        if (status === XmlListModel.Ready && count>0) {
            repeater.model = modelList;
            starMenu.open=true;
            starMenu.updateStarItemPos();
        }
    }
}
```


(C) Copyright Nokia 2013. All rights reserved.

Then a TextInput component to get the user search words to be used as a Flickr image search:

```
Rectangle {
    id: rectangle1
    x: 0; y: 0; width: 200; height: 40

    TextInput {
        id: textEdit
        anchors.fill: parent
        onAccepted: {
            repeater.model = undefined;
            modelList.tags = textEdit.text;
        }
    }
}
```

We can now use `StarMenu` with, as child, a Repeater. With the delegate a `StarItem` and the `modelList` as the model:

```
    StarMenu {
        id: starMenu

        anchors.fill: parent
        visible: open
        visualParent: mainWindow
        color: Qt.rgba(0,0,0, 0.8)
        center.x: mainWindow.width>>1
        center.y: mainWindow.height>>1
        starCenterVisible: false
        delayChangePos: 60
        shape: StarMenu.ShapeGrid
        columns: 8
        spacingH: 10
        spacingV: 25

        Repeater {
            id: repeater

            delegate: StarItem {
                width:  150; height: 200

                Rectangle {
                    anchors.fill: parent
                    color: "#490202"
                    clip: true
                    Image {
                        anchors.top: parent.top
                        width: 150; height: 200
                        fillMode: Image.PreserveAspectFit
                        source: url
                        opacity: 0
                        onStatusChanged: if (status==Image.Ready) opacity = 1
                        Behavior on opacity { NumberAnimation { duration: 5000; easing.type: Easing.OutQuint } }
                    }
                }
            }
        } // Repeater
    }
```

When the user will press return key to accept, the model of `repeater` wil be reset and the {{Icode|modelList} will be retrieved from Flickr. If all is gone well and the `modelList` is filled, all the entries of this `StarMenu` will be filled with images and it will open itself (without user interaction).

This menu has a grid shape so the images are shown in a grid with 8 columns, spaced vertically by 25 pixels and horizontally by 10. The `visualParent` is the `mainWindow`. Doing so, a click outside the entries will close the menu.

With `StarMenu` way, it is easy for example, to add a menu for each image. Inserting a {{Icode|StarMenu} inside the {{Icode|delegate}, it will come handy to pop up a menu with entries like "download", "thumb up", "share" and so on. Maybe with a circle shape kind menu or a drop down menu like in the previous tutorial.

A simple dock menu bar

This is the example number 10 (Menu10.qml) in File:StarMenu Project.zip. It will take an overview on how to create a simple dock menu with `StarMenu` plugin.

Lets start with the main qml file. Iside it put rectangle (button) where the menu will be binded to and the dock's background:

```
import QtQuick 1.1
import StarMenu 1.0

Rectangle {
    id: main
```


(C) Copyright Nokia 2013. All rights reserved.

```
    width: 600
    height: 400

    Rectangle {
        id: pressMe
        width: 100
        height: 50
        Text {
            anchors.fill: parent
            text: "press me"
        }
    }

    Rectangle {
        id: mainRect
        x: (main.width-width)>>1
        y: main.height-height+radius
        width: 380
        height: 90
        radius: 13
        visible: menu10.menuOpen

        gradient: Gradient {
            GradientStop {
                position: 0
                color: Qt.rgba(.1,.1,.1, 0.2)
            }
            GradientStop {
                position: 0.450
                color: Qt.rgba(.3,.3,.3, 0.8)
            }
            GradientStop {
                position: 0.500
                color: Qt.rgba(.8,.8,.8, 0.8)
            }
            GradientStop {
                position: 0.600
                color: Qt.rgba(.5,.5,.5, 1)
            }
        }
    }
}
```

`main` is the main view and the background is positioned at bottom center of the main view. It is visible only when the `StarMenu` with id `menu10` is open. Now create `StarMenu` component:

```
StarMenu {
    id: menu10
    anchors.fill: parent

    starCenterVisible: false
    shape: StarMenu.ShapeLine
    startingRadius: -mainRect.width/2+32
    radiusIncrement: 10
    startingAngle: 0
    center.x: mainRect.x + (mainRect.width>>1)
    center.y: mainRect.y + (mainRect.height>>1)-10

    function setStartAnimPosition()
    {
        animate1.starty = si1.y;
    }

    onCenterChanged: if (open) setStartAnimPosition();
    onOpenChanged: if (open) setStartAnimPosition();

    StarItem {
        id: si1; width: 64; height: 64
        onHoverItem: animate1.start()
        onLeaveItem: { animate1.complete();  animate1.stop(); si1.y=animate1.starty; }
        Rectangle { anchors.fill: parent; color: "#ff0000" }
    }

    SequentialAnimation {
        id: animate1
        property real starty
        loops: Animation.Infinite
        running: false
        NumberAnimation { target: si1; property: "y"; easing.type: Easing.OutSine; to: animate1.starty-20; duration: 250 }
        NumberAnimation { target: si1; property: "y"; easing.type: Easing.InSine;  to: animate1.starty; duration: 250 }
    }
}
```

The `StarMenu` properties are defined to `center` the menu into the background rectangle. With `startingRadius` the line shape rotation origin is set to the center of the bounding rect of the menu. With `radiusIncrement` we defined the space between `StarItems` and with `startingAngle` the menu will be horizontal. When `StarMenu` is open or when the mainRect changes position (the main window resizes), the function `setStartAnimPosition()` is called to initialize the animation position. The `StarItem` will catch `onHoverItem` and `onLeaveItem` signals to start and stop bouncing animation. It this example the menu has only 1 item with its own bouncing animation. You can add as many `StarItems` as you like with their own animation and maybe use the `Drop down menu` example to add a pop up menu to the dock's items like in the File:StarMenu Project.zip app.

Downloads

You will find sources of the plugin and the demo in File:StarMenu Project.zip. There are also a deb package for Nokia N9/950 and a sis package for symbian Anna or Belle.

Conclusions

I will be glad if you find this component helpful and I will greatly appreciate any feedbacks or suggestions to improve it. Feel free to contact me or use the Talk page below.

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.