Disco lights in QML- A Random Color Changing Application

This QML code example changes the screen colour randomly at a fixed time interval, creating a "disco light" effect. This is my first QML application,and was very easy to build.

Overview

Press the **start** button to cause the colour to start changing. The colour will start to change at certain interval of time.

The colour used is randomly selected using JavaScript defined in (**main.js**). This function is initiated from QML file (**main.qml**). It returns the random color which is then displayed on screen.

Logic with JavaScript

This JavaScript code shown below is the main logic of application. Here six arrays are implemented as hex value for color has six digit in it. A random Function `[Math.floor(Math.random()*hex1.length)]` of JavaScript is used to select a random letter from every array and return the color random color value.

main.js

```
function randomBg()
{
var hex1=new Array("0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E", "F")
var hex2=new Array("0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E", "F")
var hex3=new Array("0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E", "F")
var hex4=new Array("0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E", "F")
var hex5=new Array("0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E", "F")
var hex6=new Array("0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B", "C", "D", "E", "F")
var bg="#"+hex1[Math.floor(Math.random()*hex1.length)]+hex2[Math.floor(Math.random()*hex2.length)]+hex3[Math.floor(Math.random()*hex
return bg
}
```

QML UI

So here is the starting point to make a Button Bar

Button.qml

```
import QtQuick 1.0
Rectangle {
     id: container
```


(C) Copyright Nokia 2013. All rights reserved.

```
    property string text: "Button"

    signal clicked

    width: buttonLabel.width + 30; height: buttonLabel.height + 15
    border { width: 1; color: Qt.darker(activePalette.button) }
    smooth: true
    radius: 8
    gradient: Gradient {
        GradientStop {
            position: 0
            color: "#000000"
        }

        GradientStop {
            position: 0.87
            color: "#ffffff"
        }

        GradientStop {
            position: 0.84
            color: "#ffffff"
        }

        GradientStop {
            position: 0.88
            color: "#ffffff"
        }

        GradientStop {
            position: 0.99
            color: "#000000"
        }
    }

    // color the button with a gradient

    MouseArea {
        id: mouseArea
        anchors.fill: parent
        onClicked: container.clicked();
    }

    Text {
        id: buttonLabel
        anchors.centerIn: container
        color: activePalette.buttonText
        text: container.text
    }


    states: State {
                name: "pressed"; when: mouseArea.pressed == true
                PropertyChanges { target:container; opacity:0.4}
            }
}
```

This seems to be easy enough,I have created a Custom button code where the width and height of button depends of Button label And with a fix radius and with a state to show a kind of effect when button is pressed. A Signal is implemented to confirm the pressing of button.

Now coming to the main screen.

Main.qml

In this code a rectangle is implemented with respect to screen height and width, a system palette is implemented to give native look to the buttons in the application.Another rectangle is implemented with a button bar having two buttons **Start** and **Exit**. A boolean property is set for enabling and disabling, on pressing the start will check for screen enable, if it is enable the timer is triggered and will call `randomBg()` function is our javascript file(**main.js**) which will return the random colors on screen.

```
import QtQuick 1.0

import "main.js" as Main // Importing a Javascript file

Rectangle {
    id: screen

    width: 490; height: 720
    property bool enabled : false
    SystemPalette { id: activePalette }

    Item {
        width: parent.width
        anchors { top: parent.top; bottom: toolBar.top }

    }

    Rectangle {
        id: toolBar
        width: parent.width; height: 40
        color: activePalette.window
        anchors.bottom: screen.bottom

        Button {
            id: startButton
            width: 69
            height: 32
            anchors { left: parent.left; verticalCenter: parent.verticalCenter }
            text: "Start"
             onClicked: screen.enabled = !screen.enabled  //Check for screen Enable
        }
        Button {
            id: stopButton
            anchors { right: parent.right; verticalCenter: parent.verticalCenter }
            text: "Exit"
            onClicked: { Qt.quit();}
        }
        Timer {
                interval: 300; running: true; repeat: true;
                onTriggered:{if(enabled) screen.color= Main.randomBg()} // Returns the colors hex value and set the color on main sc
```


(C) Copyright Nokia 2013. All rights reserved.

```
        }
        states: State {
                name: "enabled"; when: enabled
                PropertyChanges {target: startButton; text:"Stop"}
            }
    }
}
```

Source Code

The result is here : File:Colorchangerquick.zip

References

Below are documentation and tutorials I found useful:

- QtQuick: http://doc.qt.nokia.com/4.7-snapshot/qtquick.html
- Getting started: Getting Started with Qt Quick and the Qt SDK v1.1
- Tutorial: http://developer.qt.nokia.com/wiki/Qt_Quick_Tutorial (seems to be still in development)
- QML Elements Reference: http://doc.qt.nokia.com/4.7-snapshot/qdeclarativeelements.html
- QML presentations in : Qt Developer Days Videos and Qt Developer Days Talks

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.