Archived:Creating a gradient background for a QPushButton with style sheet

Archived: This article is archived because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

Qt Quick should be used for all UI development on mobile devices. The approach described in this article (using C++ for the Qt app UI) is deprecated.

Overview

This example shows you how to create a `QPushButton` with a gradient background using a Qt style sheet.

Preconditions

- Qt is installed on your platform.
  - S60:
    - Download Qt release from here: Qt pre-release
    - Install Qt: How to Install Qt
    - Check this link for installation guide: How to install the package
  - Maemo:
    - More information about Qt on Maemo can be found here: Qt4 Maemo port

QPushButton gradient example

Style sheet

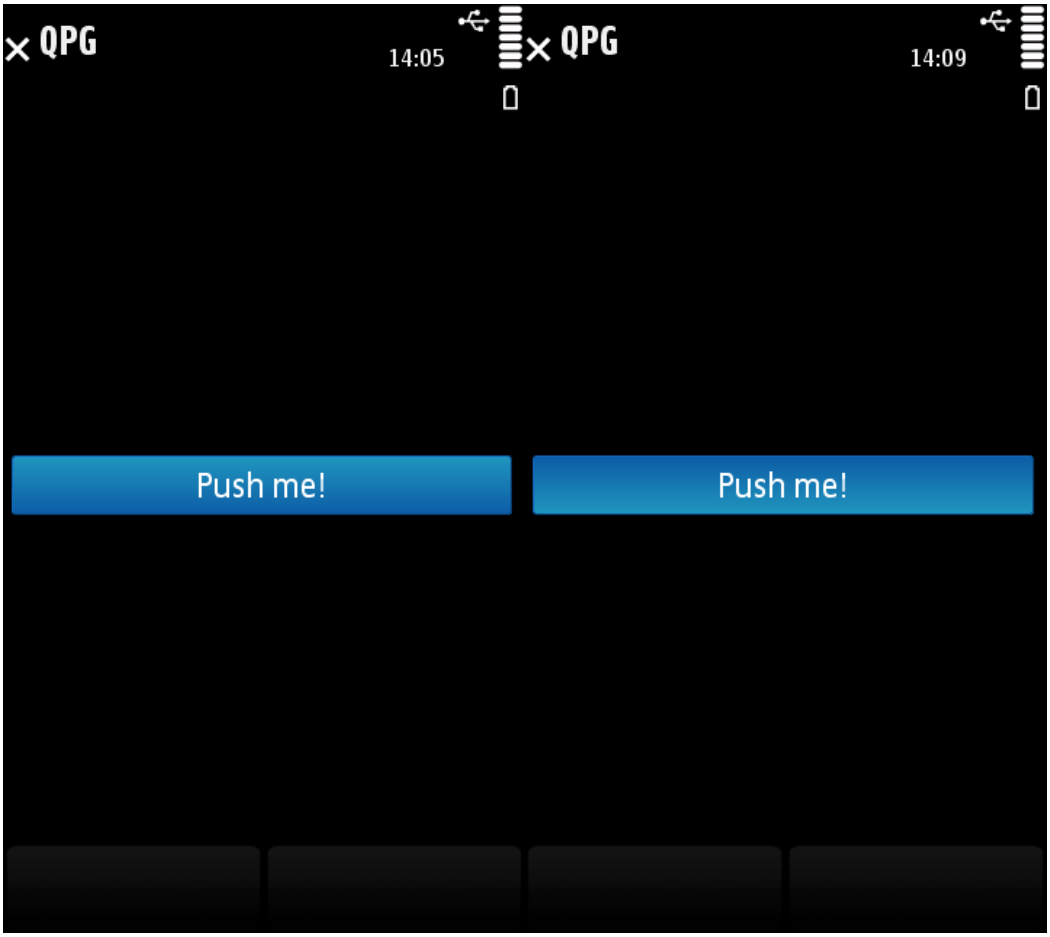
```
QPushButton {
    /* Let's make the size of the button 1,5 times of font size. */
    min-height: 1.5em;
    /* Font size just 1.*/
    font: 1em;
    /* Margins so that we get a little space on the left and right. */
    margin: 0 1px 0 1px;
    /* The font color */
    color: white;
    /* Here's the background gradient with start point, end point,
       stop "percentage" and color, stop percentage and color. */
    background-color: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
                                      stop: 0 #2198c0, stop: 1 #0d5ca6);
    /* We'll round the borders. */
    border-style: outset;
    /* Round radius will be 3px */
    border-radius: 3px;
    /* Border is only one pixel */
    border-width: 1px;
    /* Border color is now set */
    border-color: #0c457e;
}

/* This is style when button is pressed */
QPushButton:pressed {
    /* We'll just invert the gradient by changing the colors around. */
    background-color: qlineargradient(x1: 0, y1: 0, x2: 0, y2: 1,
                                      stop: 0 #0d5ca6, stop: 1 #2198c0);
}
```

Next you need to apply the style to a component. You can apply it on application level (see Archived:Applying a Qt style sheet to an application via QApplication) or on single widget level with `QWidget::setStyleSheet(const QString& style)`.

Postconditions

You've now created this:

(C) Copyright Nokia 2013. All rights reserved.



See also

- Qt style sheet documentation
- Customizing a QPushButton Using the Box Model

(C) Copyright Nokia 2013. All rights reserved.

 (C) Copyright Nokia 2013. All rights reserved.