BestPrice price comparison - app showcase

BestPrice is an app which allows you to compare prices of items quoted in different currencies or units. This article explains the design of BestPrice and its use of Qt Quick Components.

Introduction

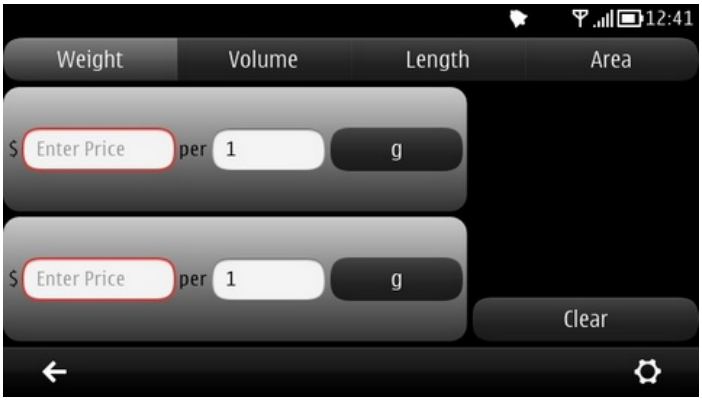
BestPrice was created to simplify the comparison of prices of two items that are not of the same amount or not labelled in the same units. In Canada, for example, price tags for items sometimes list weights in metric units, and sometimes in imperial units. So you might see an item for $1.99 per 100g next to an item for $0.59 per ounce. Determining which price is best isn't always easy in these situations.

BestPrice was designed to be used quickly and then put away. While shopping, a user should not have to scroll through pages or use more than one hand to navigate the app. As such, it has only one main screen, with an "About" page available if the user wants to get more information about the app. A legal disclaimer also appears on first launch of the app, but does not reappear if accepted.

Background

Before working on BestPrice, Minian Software released QuickStocks on the Nokia Store, which was written using Qt Widgets. While Qt Widgets provide a lot of control, we decided that Qt Quick was a better platform for BestPrice. BestPrice was initially created using Qt Quick 1.0 and Qt Quick Components 1.0. After Qt SDK 1.2 was released, the app was changed to use Qt Quick 1.1 and Qt Quick Components 1.1.

Discussion



BestPrice in Landscape Orientation

The app is a Window with StatusBar, PageStack and ToolBar components. These could be updated to a PageStackWindow component (which became available in Qt Quick Components 1.1), however the end result is not substantially different, so I chose to leave it as a Window and spend my time working on other parts of the app.

The main Page consists of a ButtonRow component, for choosing between weight, length, volume, and area units. This was a natural choice since all 4 could be displayed and ButtonRow allows one checked item at a time. If there were more categories of measurements, I would have chosen to use a component that would compact the list of catories to use less screen space. A tumbler component would have been a good choice in that case.

Below the ButtonRow are two custom PriceInfoItem components that are composed of subcomponents. We'll get back to these in a moment.

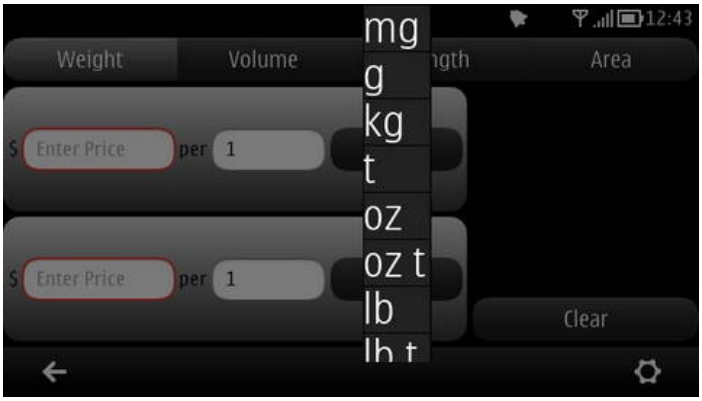
After the two PriceInfoItems are a simple TextRow to display the price difference, and a Button to reset the page.

Let's get back to the PriceInfoItem components, which contain most of the code and functionality. Each of the two PriceInfoItems allows the user to enter a price, the amount, and the units of measurement. The Price and Amount are TextField components. The units of measurement is a rectangle with text in it, which pops up a custom menu component when the user taps the rectangle.

Whenever a PriceInfoItem is updated, a piece of Javascript is called to run the price comparison. The PriceInfoItem that was updated emits a signal valuesChanged(), and the Main application handles this signal using an onValuesChanged handler. This setup is similar to the signal mechanism in Qt Widgets.

The unit selection popup menu is a custom menu based off of the information in the article found here:How to create a Context Menu with QMLThe list items are dynamically updated based on the type of measurement (length, volume, area, or weight) that the user has selected and has some repositioning code to handle the switch between portrait and landscape orientations.

(C) Copyright Nokia 2013. All rights reserved.



Unit Selection Menu

UI Design

The main UI challenge with BestPrice is that the user is on the move and short on time while using it. She is not sitting at home or on the train. She literally wants to spend no more than a few seconds to figure out what item to buy and then move on.

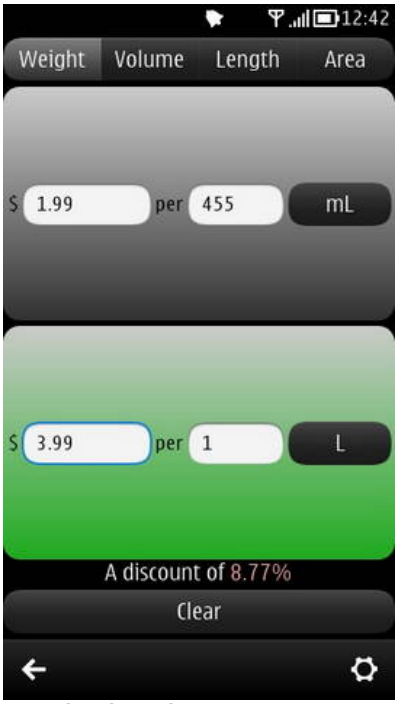
To facilitate that need, everything is single tap interaction and the most important feedback (what to buy) is visual and very obvious (the PriceInfoItem rectangle pulses green for the better-priced item). Some text showing the actual percentage difference in price is also displayed if the user wants to know and the text color changes based on how big the price difference is.

To keep things out of the way of the user, we chose to use an About screen for branding and version information instead of a splash screen. So the information is available, but doesn't use precious time (the app loads quicker than the time we would want to show a splash screen for).

Relative layout automatically adapts to different screen resolutions. And QML makes this very easy to set up using anchors.

All transitions are animated in the UI to provide a slick feel. QML animations are easy write and quick to test.

Another important UI decision was to optimize the layout for portfolio and landscape orientations. This was handled using States and a simple test of the height to width ratio of the window. You can see the two different layouts below.

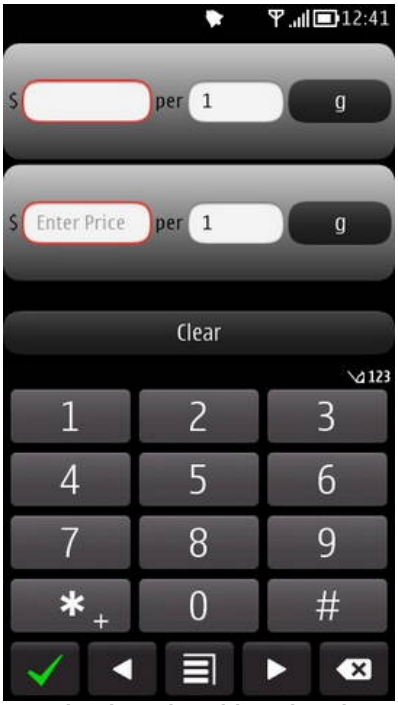


Portrait Orientation

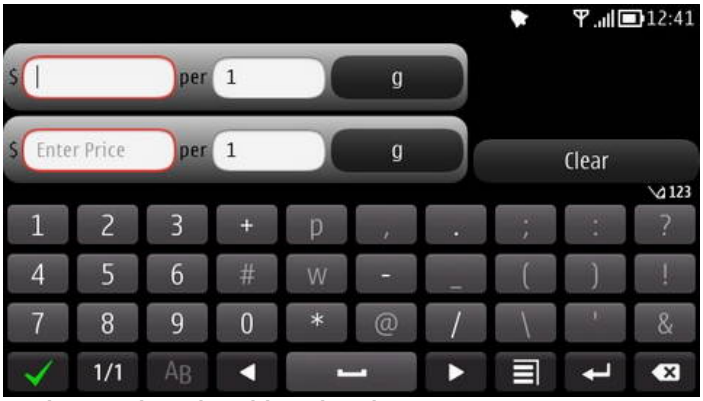
Landscape Orientation

The final element of the UI that I want to discuss was made available with Qt Quick Components 1.1 and that is the InputContext element, which gives your app information on the state of the split-view input. Previously, the keyboard was full-screen, but with Qt Quick 1.1, we have a split-view keyboard which allows part of your app to be visible while the keyboard is on screen. BestPrice uses InputContext to shrink the Main Page to the essential elements so that all of the important parts of the app are still visible while the keyboard is in use. All apps should make use of this to keep the user grounded and informed while typing.

(C) Copyright Nokia 2013. All rights reserved.



Portrait Orientation with Keyboard



Landscape Orientation with Keyboard

Make It Look and Feel Like Symbian

The Symbian Design Guidelines document provides guidance on how to design your app to look and feel like a Symbian app. Using Qt Quick Components gives you a head start because all components are built to follow the design guide by default. It's just a matter of a few lines of code to get a standard Status Bar, and Tool Bar, for instance.

Problem Areas

BestPrice is available for Symbian ^3, Anna, and Belle devices. However, there is still a large installed base of users with Symbian^1 (S60 5th Edition) devices, and unfortunately these devices cannot be targeted if you use Qt Quick Components because Qt Quick Components are not supported for Symbian ^1 devices. This means that if you want your app to run on Symbian^1, you need to write it using vanilla Qt Quick (without Qt Quick Components). For BestPrice, we find that Qt Quick Components are extremely convenient to use, but we would consider re-writing it in order to get on Symbian^1 devices. At the moment I'm still holding my breath that it will one day soon be supported on Symbian^1.

While I love the split-view input, we ran into a problem with it that we haven't been able to solve yet. Let's say the user is entering some text and has the soft keyboard up. Our UI has shrunk down to minimal view while the keyboard is visible. Now let's say the user taps a different text field to enter additional information. This causes the keyboard to slide away for a split second and then re-appear. Meanwhile, our UI begins transitioning between the minimal view to the full view and then back to minimal view. We'd like to see the keyboard remain on the screen if the user goes from one text field to another.

Wrapping It Up

QML with Qt Quick Components was absolutely the right framework for BestPrice. Using a combination of QML, Qt Quick Components, and Javascript, we were able to create an app that has the Symbian Look, performs and functions well, and has an animated UI to delight the customer. And we did it in record time.

Download

BestPrice is available on the Nokia Store for Symbian^3, Anna, and Belle devices:
Web browser: http://store.ovi.com/content/215371
Nokia mobile browser: http://store.ovi.mobi/content/215371

Note: This is an entry in the Symbian Qt Quick Components Competition 2012Q1

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.