UX Checklist

This article is a snapshot of the items listed in this UX Checklist document: UX Checklist for Touch.

Key use cases have been picked from this UX checklist. Here you can find concrete guidance on how to implement such use cases in your application code. Qt, Qt Quick, Symbian C++, and Java ME technologies are covered.

Designing for touch

Basic interaction on Nokia touch devices is based on single tap interaction. This means that 'touch down' and 'touch release' events are detected in a certain short period of time. Since one tap triggers the action the old "focus and select" style is not needed anymore, and visible focus is not shown on the screen by default. However, with every touch down event visual feedback (e.g. highlight with a suitable color, or appearance of a pushed down button) is given to a user to show that the touch has been registered and something is happening. With the touch release the visual feedback disappears. For more information about Feedback, see below.

If your application contains double tap interaction (i.e. focus and select is used), you may want to adapt it to single tap and long press interactions instead.

On Symbian devices that have a keypad and a touchscreen, focus appears on the screen only if a hardware key is pressed. Once the screen is tapped again, the focus disappears. See more information about basic interaction in the Design and User Experience Library: Designing applications for touch UI.

Use case	Qt	Qt Quick	Symbian	Java Me
Implementing a single tap	QWidget Class Reference	QML MouseArea onClicked	Modifying applications to support single-tap	Touch UI
Implementing a long press	N/A	QML MouseArea onPressAndHold	Detecting long tap using CAknLongTapDetector	Gesture API

Layout and colours

Make sure that the text has good contrast with the background - while you're checking, you may want to see if it's accessible for colour blind people as well.

Check the contrast of your colours at Colour Contrast Check.

If the screen orientation can be changed during application use, the application must adapt its appearance accordingly. If only one orientation is supported (such as for games or video), the orientation should be locked to maintain the best possible user experience.

Make sure that the controls in your application are big enough for pleasant finger usage (especially with capasitive screens where a regular stylus cannot be used). The recommended minimum size is 7 x 7 millimeters. Depending on the resolution and the dpi of the device your are creating your application for, that translates into different pixels.

Calculate the size of the pixels of your device by using PX Calc.

For more information about the scale and positioning of the controls, see Scale and positioning of controls.

Use case	Qt	Qt Quick	Symbian	Java Me
Listening for orientation changes	Archived:Listening for screen orientation changes in Qt	Implementing custom orientation changes animation with QML	Layout change events	-
Locking the orientation	Archived:Lock application orientation in Qt		Lock application orientation in Symbian	JAD and JAR manifest attributes

Softkeys

In your application, ensure the following:

- The left soft key (LSK) stands for a positive effect.

- The right soft key (RSK) stands for a negative effect.
- Exit is placed on the RSK in the main view of the application and closes the application.
- RSK is labeled Done in editable forms when LSK is labeled Options. If there is no Options menu, LSK should be Done and RSK should be Cancel.
- In wizards, LSK refers to the next step forward and RSK is used for going backwards.
- If you're not using the soft keys, ensure the users have a way to back step and exit your app either from the top or bottom right corner!

Use case	Qt	Qt Quick	Symbian	Java Me
Soft key positions	Archived:Handling softkeys in Qt	N/A (*)	Control pane	Example: Setting softkey label location
Making of full-screen application	Fullscreen applications on Qt	N/A (*)	How to make a full-screen application using Symbian C++	Archived:Changing the Canvas to full-screen mode on Symbian

(*) Adding softkeys or making a full screen application is not possible to implement with plain QML, however both of these use cases can be implemented by mixing Qt and Qt Quick.

Feedback

Provide visual and tactile feedback on user actions immediately, and for every touch down event on every item that can be interacted with. Visual feedback can be a highlight color (a button changes color when touch down is detected), or a button can look like it's actually pushed down when the user touches this (changing the button graphic completely instead of just changing the color), for example. It's important to remove the highlight or other visual feedback with the touch release event.

When triggering an activity that will take some time, in addition to the normal visual feedback, it's good to use some sort of a spinner to indicate to the user that it's going to take more than a few seconds to get to the next state.

In cases where an error occurs, an error note should be displayed. Error notes must be valid, informative, and easy to understand. They should not refer to system calls but should rather be text that the user can understand.

See more details about providing feedback in the Design and User Experience Library: Feedback.

Use case	Qt	Qt Quick	Symbian	Java Me
Providing haptic feedback effects	Feedback	Feedback QML Plugin	Tactile feedback	Tactile feedback
Providing audio feedback	QSound Class Reference	Multimedia QML Plugin	Audio Client Overview	Playing sampled sound and MIDI

Working with profiles

Make sure that your application does not play sounds if the silent profile is active. Checking the profile once, when opening the application is suitable. Constant checks are not necessary; they are likely to strain the battery in the long run. If it makes no sense to use your application without tones, you can have an additional query when opening in silent profile where the user is asked to allow sounds, open the app without sounds (if at all possible) or exit the application.

Use case	Qt	Qt Quick	Symbian	Java Me
Detecting profile changes	QSystemDeviceInfo Class Reference	QML DeviceInfo Element	How to get notification of profile change	-