Prepare your application for Nokia E6

This article contains tips and tricks to help optimise application UIs for the Nokia E6. Most of the advice given is Nokia's standard guidance for writing scalable UIs, however there are other useful tips related to keyboard support and home screen widgets etc.

01 May 2011

Introduction

The Nokia E6 is the first device to feature the Symbian Anna release. The device has a set of features that set it apart from the other Symbian^3 based products, including a touch display with VGA resolution (640x480 px) at high pixel density (326 DPI), a fixed landscape orientation, a monoblock design with keyboard and navikey.

Most applications written using existing Nokia's UI design guidelines related to scalability should display *acceptably* on the new Nokia E6 without change (for example, the screenshots in this article are from scalable applications that run without modification on the Nokia E6). Applications that have made hard coded assumptions about screen resolution, size and pixel density and key event handling may need significant changes to support scalable UIs and other differences between the Nokia E6 and other Symbian^3 devices.

This article has useful tips for developers working in all development frameworks (e.g. Symbian C++, Java, Qt).

Note: These tips were identified when testing 3rd party applications on Nokia E6. Most issues are related to scalability, but there have been some related to key handling and the updated web browser included in the Symbian Anna release.

Native applications

Native (C++) applications should work properly on all devices, regardless of their supported screen resolutions and orientations. This requires that applications correctly scale their UI to fill the available space, along with any associated fonts and graphical resources, and that touchable UI elements aren't scaled below the point where they can no longer be used. In addition, the application must respond to orientation events unless there is a good reason to lock them to a specific orientation.

Symbian C++ developers can use the Scalable UI framework to create applications that work properly on devices with different screen resolutions and at different orientations (see Symbian Scalable UI Framework for an overview of the related concepts).

Qt developers benefit from layout managers that can dynamically position and resize `QWidget` and `QGraphicsView` based components to fill the available space.

Use all of the available screen real estate

Symbian applications almost invariably occupy the whole screen. The application may take control of the whole of the screen to draw its UI, or it may allow the application framework to manage the standard screen furniture (status panes and softkey areas) and draw its UI within the remaining client rectangle.

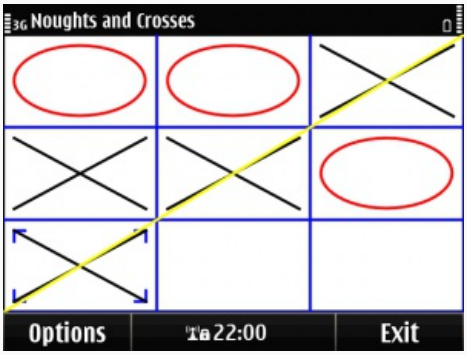
Whatever approach is used, it is important that applications make no assumptions about the available client or screen area, other than of course determining the minimum area needed for the application so that all needed UI elements can fit into the visible area. Failing to allow for screen size variation will result in UI elements placed at the wrong locations (either partially off screen or grouped together in a corner leaving significant on-screen real estate unused).

Recommendations:

- When developing Qt applications, set the MainWindow (or main widget) to show maximized and then use layout managers within this window to position, align and resize the UI elements. If using custom code for layout, do not hardcode assumptions about the screen size.
  - Qt Quick has a built-in anchoring system as well, which provides the mechanism for controlling the positioning and size of the QML components. For more information about managing different screen form factors and resolutions, please see the Qt 4.7 Scalability (snapshot) document.
- When developing Symbian C++ applications:
  - where possible use predefined templates. Avkon UI components including forms, list, toolbars, navigation tabs etc. scale automatically and will be positioned correctly whether the app is run on Nokia 5500's tiny screen or on a device with VGA resolution like the Nokia E6.
  - when using custom UI elements always ensure that they are positioned and sized relative to each device's screen size

(C) Copyright Nokia 2014. All rights reserved.

and client rectangle geometry.



Noughts and Crosses - designed to work on Symbian/S60 3.x and later devices, of any resolution and form factor, touch, non-touch or hybrid. Works perfectly on Nokia E6 as well.

Settings Screen - designed to use stock Avkon controls, which are guaranteed to scale automatically to all resolutions, following the predefined platform layout.

Scale graphic resources

Graphic resources associated with resizable UI components may also need to be scaled. For best results, use scalable vector graphics (SVG) whenever possible, *even if the API can also accept bitmap format*.

When providing a background image to your UI elements (views, lists, etc.) you should ensure that the respective resource is either large enough to cover all possible resolutions or that you scale the available resource accordingly. The best approach is for the application to use both solutions, and have a high resolution image that can be used at native resolution or can be scaled-up when needed. SVG resources could be considered for this use case as well.

Platform Scalable Screen Drawing Example v1 4 en.zip.html
Scalable Screen-Drawing - an application that scales to multiple display resolutions in portrait and landscape orientations supported by the Symbian platform. Targeted especially at games and multimedia developers who prefer developing a custom UI instead of using an automatically scaling UI, such as Avkon components. A large raster image (800x500) provides the background on which various scalable UI elements (SVG stars, text) are sized and positioned dinamically.

Finger-touchable UI elements

Nokia's devices are expected to be finger touch friendly, which simply means that UI elements must be big enough for the user to be able to tap their active areas. The Symbian UI style guide indicates that active elements for touch functions should be at least 7x7 mm in size, with at least 1 mm spacing in between them. See Scale and positioning of controls in the Design and User Experience Library (v2.2)

Note that not all devices have the same display parameters. 7x7mm translates in different sizes in pixels, depending on screen's DPI parameter. Applications should take this factor under consideration as well, and dynamically calculate the minimal UI sizes of the finger touchable UI elements. The components can then be allowed to scale up from the given minimal value for each device,

(C) Copyright Nokia 2014. All rights reserved.

but not below it.

Orientation changes

Most Nokia devices will change display logical orientation when the device is tilted, adapting to device's current position. Applications are expected to handle the orientation changed notification and adapt their application UI to the new client area.

Note however that not all devices have orientation change enabled and that, at for certain form factors, the screen orientation is not always in sync with device's orientation. For example, on Nokia N8 the screen has the same orientation as the device while on the Nokia E6 the screen is in landscape mode when the device is in portrait mode and the orientation does not change when the phone is tilted.

Applications should avoid hard coding the phone's orientation (e.g. always switch to landscape), leaving it a decision to be made based on reading the phone's actual parameters.

Font size

The DPI parameter of the screen can also have a significant impact on the size of the fonts used by the application. At higher DPI the font would appear to be smaller, therefore the application must also dynamically manage the font size.

The recommended way is to use the logical fonts provided through the UI Framework Utilities API (*AknLayoutUtils* ...)

With Qt/QML, the Qt Quick Components offer the Style QML element which provides device specific constants similar to Avkon's logical fonts, which allow applications to use fonts of predefined sizes defined to match the DPI and overall look and feel of the targeted device.

```
 import com.nokia.symbian 1.0
...

Text
{
    font.pixelSize: platformStyle.fontSizeSmall
    text: "Small font on all Symbian devices although the actual pixel size varies
function of screen's DPI"
    color: platformStyle.colorNormalLight
 }
```

Keyboard handling

Symbian touch devices may also have a physical keyboard, and this may only be active in certain operation modes (foldable keyboard can be exposed or hidden) or may be active all the time (monoblock design with touch screen).

Applications should be designed to allow both touch and keyboard operation. For example when a list of options is presented to the user he/she may choose to make the selection by tapping on the screen or by using a navikey or directional keys from the keyboard.

In particular for Nokia E6, the device does not have a dedicated camera key. Applications should therefore be ready to handle `EKeyOk` as a camera trigger button.

The virtual keyboard in split view

With the new Symbian Anna release more of the in-built applications are taking advantage of the split view input and end-users will be expecting to see the same in 3rd party applications. While not particularly useful for the Nokia E6, which has an always-on physical qwerty keyboard, adding the new functionality is rather easy (once you have taken care of the above discussed scalability issues described above). For an example of how to implement the split view input, see Split view input in Symbian C++ applications.

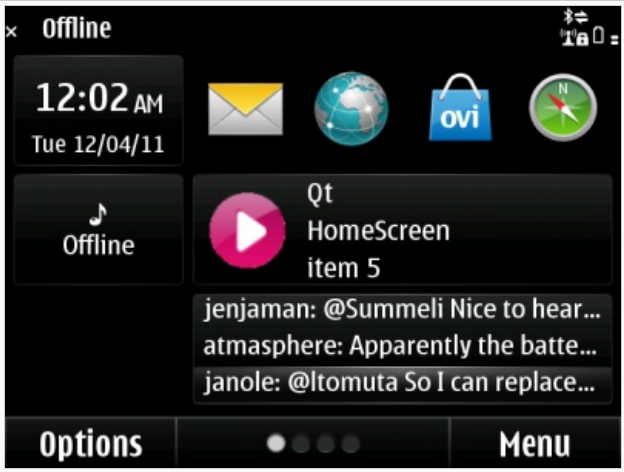
Home Screen widgets

Symbian^3 does not provide a public API for publishing content to home screen widgets. However, the Nokia N97 device specific API Home Screen Publishing API can also be used on the Nokia E6.

Note however that the higher pixel density on the Nokia E6 means that their size is increased from 312x82 to 436x115 pixels. While content is automatically scaled and existing widgets will continue to work, images may become pixelated. Using

(C) Copyright Nokia 2014. All rights reserved.

larger/scalable graphic resources, matching the new VGA CSS, will produce better results. The image below shows two unmodified widgets running on VGA layout: the Qt Homescreen (using the *threerows* template) and Gravity (with a *threerowtext* template).



Screenshot showing two Home Screen widgets implemented using the native API provided for Nokia N97. Despite the lager size of the widget template itself, the content (image, text) is scaled up so that to the user the widget will look just as it did at lower resolution.

The CSS of the published templates has been updated as well, see Home Screen widget template CSS for VGA devices

If you are developing a widget with a custom template, using Symbian's platform XML API, you must create a new variant for VGA layout, with the manifest defining the family as *vga_tch*.

Java applications

The considerations presented above regarding application scalability remain valid. They are reflected in Java Developer Library's Scalability chapter.

Do not use Nokia-MIDlet-App-Orientation

Starting from S60 5th Edition, Java applications can force UI to either portrait or landscape orientation by using JAD attribute *Nokia-MIDlet-App-Orientation*.

This JAD attribute should not be used with devices with VGA display resolution because:

- Since VGA display resolution (640 x 480) stands for having landscape orientation as the default and only orientation for application UI, setting landscape orientation mode by using the JAD attribute is not required.
- Setting portrait orientation mode by using the JAD attribute can result in unexpected and undesired outcome on placing the layout of application UI on the display.

Scale to VGA resolution

In Symbian starting from S60 3rd Edition, *Nokia-MIDlet-Original-Display-Size* and *Nokia-MIDlet-Target-Display-Size* attributes can be used for graphics scaling of full screen Canvases.

In case a Java applications which targets a VGA device and which has been originally set to use these JAD attributes for some other resolution than VGA, redefining the values of the attributes needs to be done in order to optimize the application specifically for VGA display resolution.

The JAD attributes can be used for scaling the Java application for VGA resolution as follows:

- For Java applications which are designed and targeted only for devices with VGA resolution, it is sufficient in most cases to define only Nokia-MIDlet-Original-Display-Size attribute which will enable the application to use full screen automatically while maintaining the original aspect ratio (although that might leave black edges on the display area).
- The attribute *Nokia-MIDlet-Target-Display-Size* should only be used in special cases, for example when application resolution needs to be limited to a certain size or aspect ratio. If the attribute is not set, the MIDlet is scaled without changing aspect ratio. For VGA resolution, application will be scaled to fill the whole screen but the aspect ratio changes and thus the result can be visibly different (which, depending on the use case, can have impact from user experience point of view).

More information

(C) Copyright Nokia 2014. All rights reserved.

More about graphics scaling and related JAD attributes can be found from Nokia Developer Java Developer's Library.

- Graphics scaling for Canvas
- JAD and JAR manifest attributes

Web and Web Runtime applications

In addition to the considerations imposed by the new screen resolution, Web and Web Runtime developers will also benefit from an updated browser and improved widget security prompts user experience. For details, please see Symbian Browser and Web Runtime 7.3 for Developers

Ovi App Wizard

Applications generated using the Ovi App Wizard tool will be automatically scaled to VGA resolution.

Testing your application

To test your application on a Nokia E6 device you can use the free Remote Device Access service, provided by Nokia Developer. The current releases of the Qt SDK 1.1 and Symbian^3 SDK 1.0 have simulation/emulation support for VGA screen resolution as well.

VGA resolution model in the Qt Simulator. Can be used for testing application's scalability, in special when targeting the Nokia E6 device.

The Symbian^3 emulator running in VGA resolution mode.

(C) Copyright Nokia 2014. All rights reserved.