

Neonode N1

Development Description

Description: White paper describing application development for the Neonode N1 handset.
Author: Johann Gerell, Neonode AB.
Revisions: 2003-02-19, Version 1.
Security level: Public.

Contents

Contents	1
Disclaimer	1
Introduction	1
Updates.....	2
Terminology	2
Technical Overview	3
Device Layout.....	3
Basic Functions.....	3
Mouse Event Resolution	3
Navigation	4
Developing Applications	5
Tools	5
<i>IDE</i>	5
<i>SDK</i>	5
<i>The Neonode Developer Forum</i>	5
Special N1 considerations	5
<i>Touch Screen</i>	5
<i>Navigational Decisions</i>	6
<i>Application Windows and Child Controls</i>	6
<i>Keyboard</i>	7
<i>Additional Application Features</i>	7
<i>File Browser</i>	7
<i>Yes and No Buttons</i>	8
<i>Direction Buttons</i>	8

Disclaimer

The contents of this document are Copyright © Neonode AB 2002-2003.

Introduction

Application development for the Neonode N1 handset is supposed to be fun. If you know how to do it, then it will also be easy — as with anything else in life. Reading and understanding the topics discussed in this document will greatly enhance the probability for your application to be a “killer app”.

Questions on this document or general questions regarding development for the Neonode N1 handset must be directed to the Neonode developer forum found at <http://www.neonode.com>. Private emails to the members of the Neonode crew will not be answered of the simple reason that we want information to reach all interested developers and that goal can only be achieved using the forum consistently. Members of the N1 crew will monitor the developer forum regularly.

The Neonode N1 handset is running the Windows CE .NET operating system from Microsoft. The functional and graphical environment built on top of Windows CE .NET specifically for the N1 is called NeoShell. If an application wants to blend well with other applications on the handset and take advantage of system functionality exposed only through NeoShell, it should conform to some guidelines and specifications imposed partly by NeoShell and partly by technical features of the handset. These issues are described in this white paper.

Updates

It is of great importance to realize that the SDK mentioned in this document is a work in progress. The N1 SDK will be updated with minor releases when Neonode see the need for it, in the same way that the Microsoft Platform SDK is updated every now and then. However, individual APIs will remain untouched in the same major release to keep backward compatibility with applications that already use them.

New NeoShell features that expose functionality usable for general application development can for instance trigger updates, and the availability of the updates will be announced in the Neonode Developer Forum found at <http://www.neonode.com>.

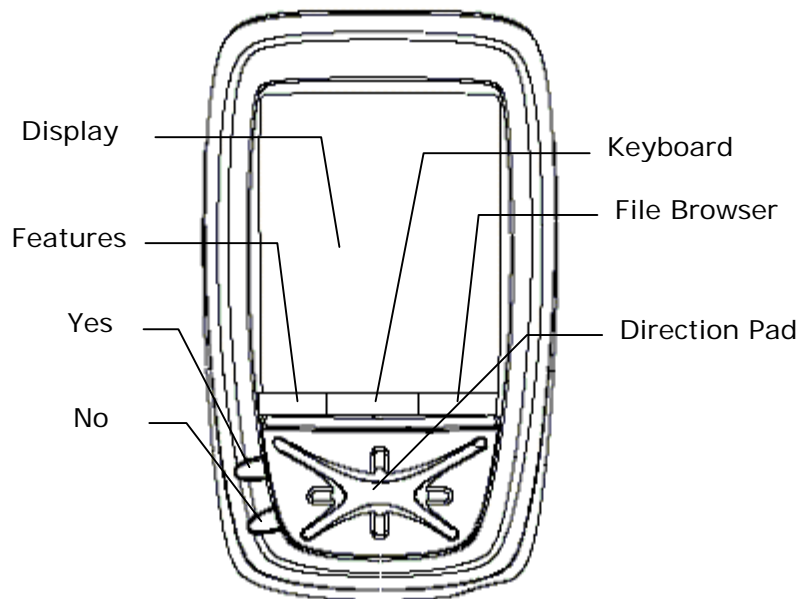
Terminology

To avoid meaningless repetition, the following simplifications will be used henceforth, effective immediately after respective explanation.

- *The N1* refers to the Neonode N1 handset.
- *The SDK* refers to the N1 SDK.
- *NeoShell* is the functional and graphical shell on N1.

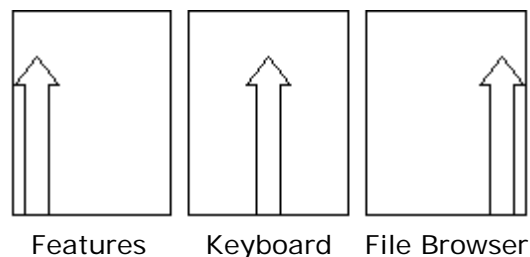
Technical Overview

Device Layout



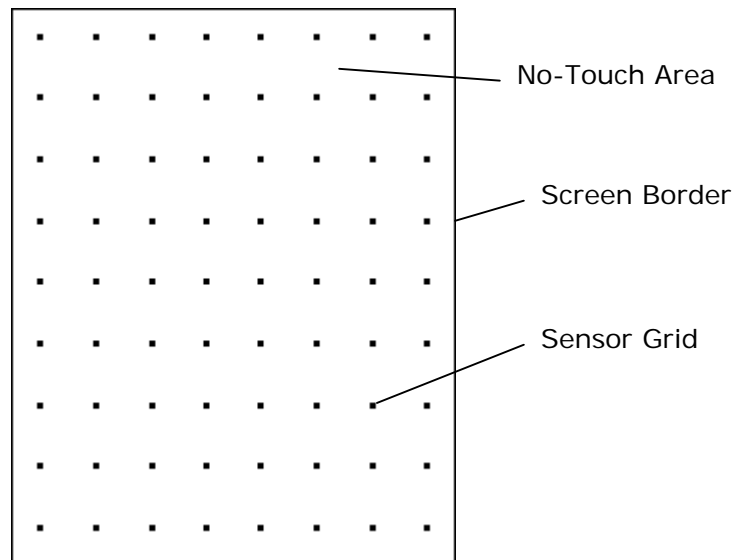
Basic Functions

The N1 has three basic functions from which additional tasks can be performed. These functions can be reached using the three "buttons" below the screen and they are from left to right "Additional Features" (+), Keyboard (⌨) and File Browser (📁). Putting a finger on one of these buttons and sliding upwards will show the respective function:



Mouse Event Resolution

There are 8 sensors from left to right and 9 from top to bottom. This gives at a first glance approximately 22 horizontal pixels touch event resolution and 24 vertical pixels resolution. But, since the touch point is calculated by interpolation if two or more sensors are active, and as there is also a band at the very outer parts of the screen where no touch events at all are registered, the resolution is actually considerably higher. The sensors are placed as shown in the picture below.



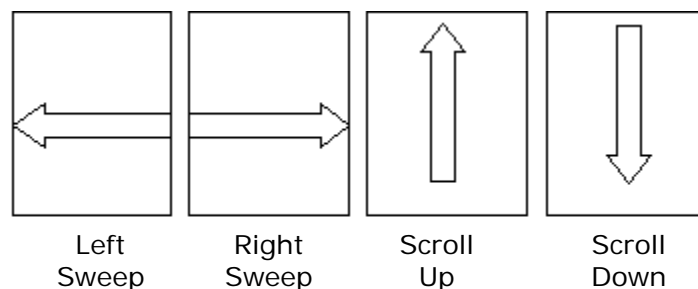
The white area surrounding the sensor grid points shows the no-touch area. The no-touch area closest to the screen border is 12-14 pixels wide, and this leaves about 150 horizontal pixels for the sensors. That gives an interpolated touch event resolution of 9-10 pixels. Vertically we have about 194 pixels to play with, and that means 10-11 pixels touch event resolution.

Navigation

Since the N1 should be easy to use with one hand, two basic means of navigation in an application are defined:

- To "close" the current task, whatever that might be, make a sweep with a finger from the right to the left. This is defined as *left sweep*.
- To "continue" to the next task or perform whatever other operation that would currently fit into the application (like moving up towards the root folder in the File Browser), instead make a *right sweep*, i.e., make a sweep with a finger from the left to the right.

To move something up or down in a window (like an edit control caret or list box selection), *up and down scrolls* should be made. A scroll can be described as a vertical sweep. The sweep and scroll gestures look like this:



Developing Applications

Tools

There are some constraints on the diversity of tools possible to use for N1 development. They are actually limited to a couple of Microsoft IDEs and the N1 SDK must accompany them. However, lots of relevant functionality, both logically and graphically, can be developed without the SDK if some rules are enforced — see *Special N1 Considerations*. A detailed description on how to set up the IDEs to function properly with the SDK will be included in the SDK documentation.

IDE

The two IDEs that support development for the Windows CE .NET platform (not counting Platform Builder, which is used to build the actual platform) are eMbedded Visual C++ 4 (eVC) and Visual Studio .NET with the .NET Compact Framework (VS). With eVC the only supported language is C++, but with VS it is also possible to use C# and Visual Basic.

Downloading eVC is free from Microsoft's web site, but be warned — it is a considerable amount of data to transfer. VS on the other hand is a commercial product.

SDK

Downloading of the SDK can be initiated from Neonode's web site.

The Neonode Developer Forum

The primary sources of information about N1 development are the Neonode Developer Forum found at <http://www.neonode.com>. Detailed SDK reference is bundled with the SDK distribution in standard HTML Help format.

Special N1 considerations

Touch Screen

N1 has an unconventional touch screen where a sensor grid defines the touch point. A screen touch event is registered when a something like a finger activates the sensor, and the screen driver will then send the touch events to the system. The system will in turn generate conventional Windows mouse messages, for the windows available on the screen.

Since the sensor grid is sparse, it is impossible to get pixel resolution in the touch event position. An application must therefore be graphically designed with this in mind and extraordinary solutions might be enforced. For example, restrictions on the size of “things” meant for user interaction in the GUI are imposed. For instance, small buttons can be impossible to press if they are misplaced in regards to a sensor point and touch positioning of the caret in an edit control is virtually undoable of the same reason.

Navigational Decisions

The sweep and scroll events are defined by the start and stop positions of the finger movement together with a time frame. NeoShell does the decision whether a movement is actually a sweep, a scroll or nothing. The SDK contains a helper window implementation encapsulated in a C++ class, CNeoWindow, which developers are encouraged to derive from for their own application windows or simply use as it stands for basic functionality. CNeoWindow will generate sweep and scroll events in the form of virtual functions, which should be implemented by a subclass if the described behavior is desired. The SDK includes a sample application *Testapp*, which uses CNeoWindow and some NeoShell functions.

Developers using languages other than C++ may have to implement the decision making themselves, using the sweep- and scroll criteria defined in the SDK documentation.

Application Windows and Child Controls

Full Screen Windows: Only full screen windows are allowed, i.e., 176x220 pixels, unless we are talking about child window controls. NeoShell do have means to prohibit other window styles and sizes than the stipulated, so even if windows are created with custom sizes or styles, it is not for certain that they will look as anticipated, since NeoShell will try to alter things “back to normal”. The reason for this behavior is that we want all N1 applications to be part of a complete user experience. Individual styles and sizes can and will disrupt the overall N1 impression.

Main Windows: Main windows must not have standard title bars or captions, since they occupy too much valuable screen real estate.

Borders and Backgrounds: Avoid 3D borders (3D is so... Windows 95 — do you not agree?) on controls. Flat, borderless controls imposed on a snazzy background image are preferred.

Edit and Button controls: Avoid single line edit or button controls (or similar) using the standard system font size if they require user interaction, since it is hard to actually hit them considering the limited touch event resolution.

Dialog boxes and modal windows: Do not use the system dialog boxes created by the Win32 API DialogBox... or CreateDialog... function families. In fact, you cannot use them at all, because they are disabled on the N1. This makes the Microsoft dialog resource editor useless for N1 development — all controls in a window must be created in the code.

If you feel you cannot live without dialog windows, please reconsider your application architecture, because everything can actually be done without them, it just takes some thought. Instead, the SDK provides a modal window component helper implementation encapsulated in a C++ class, to be used when synchronous user interaction is required — not as an application main window. The name of this class is CNeoModalWindow. Developers using languages other than C++ must implement their own solution to handle modality. Finally, remember this statement: *modal dialogs are not needed*.


Message Box: The SDK provides a custom message box component that can be used instead of the conventional Win32 API message box. The function

arguments are the same, so it will be an easy task to cut and paste in existing code, if necessary. The standard system message box will be re-implemented in terms of this component, which is actually derived from CNeoModalWindow.


Looks: Avoid the standard Windows system colors. As mentioned earlier, snazzy images that do not draw attention from the task at hand are preferred. Please avoid messy creations that make the brain twist. And do not clutter the application main window — keep it simple and airy.

Keyboard


The user normally triggers the N1 keyboard, after first setting focus to something editable in an application window. But an application can also force the keyboard to show if there is no question that when something is toggled or getting focus, the user wants to edit text of some sort.

The scheme is like this: When a “thing” (an edit control, a list item, etc.) is activated in an application, the application sets up a data structure with basic information, like which window the keyboard should work with, if the keyboard should be in digit or alphabetical mode and the current text length constraints. This structure is then sent to NeoShell using a function from the SDK. The next step is either to immediately show the keyboard from the code using another SDK function or to wait for the user to bring it up manually using the device keyboard button marked . When the keyboard is shown, the previous window will be deactivated and the keyboard data must then be reset by yet another SDK function, so that no dangling data is present if the keyboard is later shown to make a simple call or something like that.

Additional Application Features

The leftmost device special function icon  will show an “additional features” menu that is tied to an application in such a way that the application places a full-screen image divided into six or less parts (two columns and three rows) in its installation folder on the device. When the application starts, it tells NeoShell, using a SDK function, that it will support such a menu and that the path to the image is this or that. When the features icon is toggled, the menu will be shown only if the current application has registered itself as having such a menu. When one of the fields in the menu is pushed, the key events F1 to F6 are sent to the application and it is then up to the application to deal with them as it see fit, like showing settings or other features central for the main view. For instance, an image viewer application could change the name of a picture when F2 is received and email the picture when getting F3. F1 should always be reserved for a help text of some kind.

File Browser

The rightmost device special function icon  will show a file browser, with which the file system can be browsed, files can be opened and applications launched. A task manager listing the current running applications is also available to allow killing of unwanted or hung processes. The topmost items in the browser are configurable to allow easy access to commonly used items.

Yes and No Buttons

Use the action buttons *Yes* and *No* when interaction of the type *Yes/No*, *OK/Cancel*, *On/Off* etc. are required. These generate *Return* and *Escape* key events to the system. Never present the user with 3 choices, but rather break down the problem in smaller pieces. For back/forward type of navigation, left and right sweep should be used. NeoShell remaps the action buttons when an incoming call etc. is received.

Direction Buttons

The four-way keypad below the screen will generate ordinary left, right, up and down key events. These could be used in for instance games, but it is ultimately up to the developer to find the proper usage model for these keys in his own application.