

✔ This tick symbolizes the featured content on the Forum Nokia Wiki. Featured articles are considered to be good quality articles in the Forum Nokia Wiki, as determined by the [Forum Nokia Wiki administrators](#). Before being listed here, these articles are reviewed for accuracy, neutrality, completeness, and style.

A small tick (✔) in the top-right corner of an article indicates that the article is a featured article.

View by category: [All](#) | [Symbian C++](#) | [Open C/C++](#) | [Java](#) | [Qt](#) | [Python](#) | [Flash Lite](#) | [WRT Widget](#) | [Maemo](#) | [Other](#)

How do I start programming for Symbian OS?

Week 33 - August 9th 2009

This week's featured article, [How do I start programming for Symbian OS?](#), is written by [Sellis](#) which has been enhanced by [Lucian Tomuta](#) and other Wiki contributors over a period of time.



[Symbian C++](#) is the native programming language of [Symbian OS](#) devices. In order to achieve efficient exception handling and memory management in resource-constrained mobile devices, [Symbian C++](#) provides certain fundamental concepts different from standard C++.

Most of the people who want to learn programming with [Symbian C++](#) and have no prior experience at all, find themselves in the situation where they don't know where to start. This article is a step by step guide for developing software applications for the [Symbian OS](#) with [Symbian C++](#) programming language.

This article covers various steps including choosing an appropriate [SDK](#), working with an [IDE](#) and suggesting reference materials. The article has been translated in different languages such as Portuguese, Chinese and Russian.

[Read the article](#) and start creating your applications with the native and powerful [Symbian C++](#) programming language to work on [Symbian OS](#).

SMS Operations

Week 23 - May 31st 2009

This week's featured article, [SMS Operations](#), is created by [Kiran10182](#).



Short Message Service (SMS) is an integral part of the mobile communication. The SMS technology has facilitated the development and growth of text messaging. The connection between the phenomenon of text messaging and the underlying technology is so great that in parts of the world the term "SMS" is used as a synonym for a text message.

This article explains SMS technology from the programming point of view. It explains the basic SMS related operations which form the backbone of the SMS programming; such as sending messages, reading messages,

Featured_Articles:_Symbian_C++

deleting messages, disabling delivery reports etc. with the use of [Symbian C++](#) APIs.

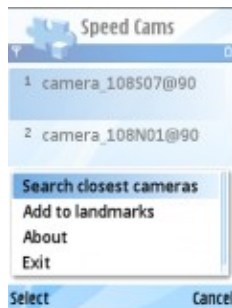
The article extends the engine interface, which is available to download in the article, for different SMS related activities. A step by step explanation is provided which makes it easy for the developer to learn the article.

[Read the article](#) to implement various SMS related operations in your [Symbian C++](#) applications.

Landmarks/web client example using Carbide.c++ and UI designer

Week 10 - March 1st 2009

This week's featured article, [Landmarks/web client example using Carbide.c++ and UI designer](#), is written by [Petrosoj](#).



The article walks through the steps of developing an [S60](#) application that sends an HTTP-query to a web service to fetch landmarks information. Carbide UI designer is used to create the application UI in the article.

The functionality presented in the article is to query the last known position from the default positioning module, sending HTTP query to GPS waypoints client API, where GPS Waypoints is a GPS Community where you can share your GPS POI (Point of Interest) and parsing the query results to create application UI and saving query results to the landmarks database of the device.

The basic knowledge of [Carbide.c++](#) and [Symbian C++](#) is prerequisite to learn this article. A working example and a corresponding installation file is attached in the article.

[Read the article](#) to retrieve the information of the traffic cameras closest to your last known position.

How to create a Symbian C++ project with Unit Tests

Week 7 - February 8th 2009

This week's featured article, [How to create a Symbian C++ project with Unit Tests](#), is written by [Danilo Freire](#).



Unit testing is a process of verifying individual units of the source code, where a unit may be an individual program, function, procedure, etc. The goal of unit testing is to isolate each part of the program and show that

the individual parts are correct.

This article presents a step-by-step guide that explains how to prepare a Symbian C++ project(S60 3rd edition platform) to run unit tests using the SymbianOSUnit Framework. SymbianOSUnit is a free and open source testing framework for Symbian.

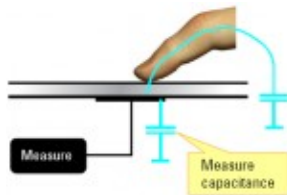
The article explains how to import the SymbianOSUnit framework code into the Carbide.c++ workspace and then how to write test suites for the code. Finally, it demonstrates setting up the project files to build and run test suites.

Read the article to integrate unit testing suites in your projects.

TouchUI: Event from hardware to software

Week 5 - January 25th 2009

This week's featured article, TouchUI: Event from hardware to software, is written by Mahbub s60.



The touch screen technology is widely used in PDAs, smartphones, ATM, information kiosks and in many other devices. This article focuses shortly on the touch UI technology and how the information is flown from hardware to the applications via operating system.

The touch input is detected by a touch sensor. There are two major types of sensors, resistive and capacitive technology. In this article both the types of sensors are well described with suitable graphics and benefits of one over the another.

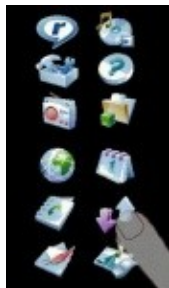
The article explains the role of the controller which filters out erroneous touches and determines which part of the UI was touched, by calculating the coordinates. A window server event is generated by the touch event that is routed to UI framework which is further processed by the application.

Read the article to learn what happens on touching the touch enabled device.

AppLauncher - S60 5th Touch UI

Week 52 - December 21st 2008

This week, AppLauncher - S60 5th Touch UI by Stenlik is selected as the featured article of the week.



Featured_Articles:_Symbian_C++

This article addresses how to use the touch screen functionality for the semi-3D application launcher. The concept of OpenGL functionality (rotating cube with texture) with the finger-touchscreen trajectory detection algorithm is also demonstrated in the [article](#).

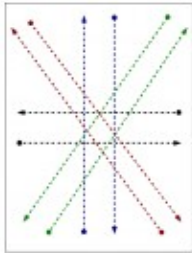
The example illustrated in the article uses mainly the `HandlePointerEvent (const TPointerEvent &aPointerEvent)` method, which gives information about the type of the generated event and the position on the screen where the event occurs. Based on the array of pointers obtained, the pixel-based coordinates are then transformed to the X-Y coordinate system before the angle is computed and the direction is determined. The article contains detailed images to help understand the example. The example is also available for download in the article.

[Read the article](#) which has a very good visual appeal as well.

Using basic touch gestures

Week 51 - December 14th 2008

This week's featured article, [Using basic touch gestures](#), is written by [Damavik](#).



The article explains the user experience aspects to be considered with the application design. With the support of Touch UI, the amount of user actions for the same application functions can be reduced. One of the possibilities is to map basic touch gestures to the application main functions.

After mapping main functions of the application to the touch gestures, we can remove the need for the users to take attentive look at the application. For example, moving to next track in media player can be performed with Left-Right gesture and moving to previous track with Right-Left gesture.

The article demonstrates how to use basic touch gestures with suitable example code. You can also download the example application from the link given in the article.

[Read the article](#) to find out more about touch gestures.

A tour to the S60 Touch UI components

Week 50 - December 7th 2008

[A tour to the S60 Touch UI components](#), the featured article for this week, is authored by [Kiran10182](#).



Featured_Articles:_Symbian_C++

S60 5th Edition is the latest generation of the S60 platform. A new touchscreen capability with tactile feedback, sensor framework, and support for QHD screens brings an unprecedented level of expression and usability to S60 devices. These new features enable application developers to build more features into their UIs, using the extra screen real estate, leverage touch, tactile feedback, and sensors to provide users with innovative ways of interacting with applications.

The article features new APIs being offered by S60 5th Edition Touch UI platform. Among the new APIs featured in the article are Toolbar API, Long Tap Detector API, Stylus Popup Menu API, Tactile Feedback Client API, Adaptive Search Feature, Choice List API, Generic Button API, SingleStyleTreeList with Hierarchical Lists API and SingleColumnStyleTreeList with Hierarchical Lists API. Each API is well described with screenshots, code snippets, and example applications.

Read the article and be a member of the escorted tour towards mobile application development with S60 5th Edition.

Trace Function Enter, Exit and Leave

Week 47 - November 16th 2008

This week's featured article, Trace Function Enter, Exit and Leave, continues the theme (from last week's featured article) of log based debugging.



The article shows how the code can be instrumented so that it will generate logs whenever a method is being executed.

Your application does not work? Apply the logging technique described in the article and you will soon find out what was the last of your methods that was executed, and whether it was successful or not. Once you know that, you can then focus your attention to the faulty method, add more detailed logging if needed, and eventually identify and correct the problem.

The logs collected will show when the execution pointer is entering the targeted method and when it exits, but more than that, it will also show whether the method had a clean exit or whether it was forced to quit by a leave. On WINSCW target (that is, an emulator) the author employs an assembler code based trick which allows retrieval of method's exit code. To further enhance the topic, it would be great if someone could contribute a similar trick for the device targets so that the information can be collected when running on the real phone too, not just on the emulator - such contribution would not go unnoticed!

Read the article...

How to use RDebug

Week 46 - November 10th 2008

This week's featured article, How to use RDebug, provides information about a basic debugging technique: logging.



The article shows how to use a Symbian defined API, RDebug, in order to generate logs which can then be captured in real time either by the IDE or by other specialized tools.

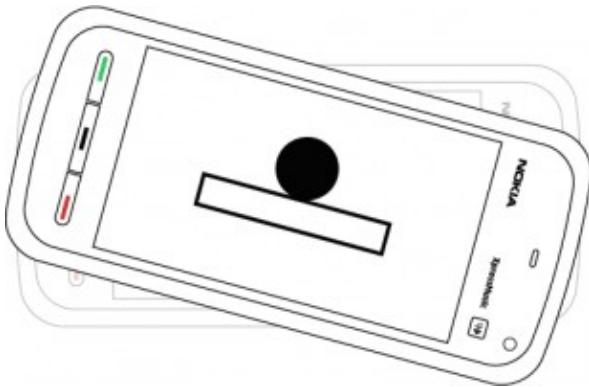
This technique is used intensively in the S60 emulator, where many of the system components are using it to dump potentially useful log data. Unfortunately it gets to the point of having too much log content so that useful information may get lost in a sea of text. This article shows how the IDE can be extended with an existing 3rd party Eclipse plug-in which will help in finding and highlighting your custom log messages.

[Read the article...](#)

S60 Sensor Framework

Week 45 - November 2nd 2008

This week's featured article, [S60 Sensor Framework](#), is written by Dr. Jukka Silvennoinen (better known as [symbianyucca](#)).



One of the most active technology areas for [S60](#) developers is, no doubt about it, [Sensors](#). Although officially supported only by the Nokia 5500 Sport, developers have managed to find resources that enable them to use the "hidden" sensors on some other devices (such as the Nokia N95) and now sites such as youtube.com are filled with application demos. For a list of sensor-based applications, click ["here"](#).

The S60 5th Edition release introduces a platform-level Sensor Framework. This new standardised set of APIs allows developers to write sensor-aware code that can be run on all [S60](#) devices and utilises the sensors that the manufacturers have provided on them. No more fighting to identify "secret" APIs and magic UID values or switching between this vendor's API and the totally incompatible API of the other vendors!

On a related subject, S60 5th Edition also includes the S60 Platform Services framework, which provides various native platform features (including sensors) for [WRT widgets](#) and [Flash Lite](#) applications.

This week's featured article gives an overview of the [S60 Sensor Framework](#), explains how it should be used, and also provides an example application - all you need to get started in using this new set of APIs.

[Read the article...](#)

Using accelerometer - Hourglass

Week 37 - September 7th 2008

[Using accelerometer - Hourglass](#) is this week's featured article. The author of the article is **Diego Soares Lopes**.

The article reminds you of the times when people used to have a hourglass as a means of following the time. Let's go back to these times and watch the time in an hourglass on the Nokia N95 device.

[Using accelerometer - Hourglass](#) is an interesting application using the accelerometer. It simulates a real hourglass in a N95, so every time you change the device's position, an observer updates the acceleration values in each axis, ranging from -360 to 360. The code snippet shows how to detect sensors in the device.

The accelerometer is a device that senses inclination, vibration, and shock. [Using accelerometer - Hourglass](#) uses the [Sensor API](#) and implements the *HandleDataEventL* callback function from the **MRRSensorDataListener** interface. [Read more in article](#).

S60 View Architecture with UI Design

Week 36 - August 31st 2008

[S60 View Architecture with UI Design](#) is featured article of this week. It is created, rather designed by [Olympio](#) with his creative insight and well structured approach. The main concepts of the S60 view architecture are explored in this article using Carbide UI Designer.

[S60 View Architecture with UI Design](#) illustrates how efficiently one can use Carbide UI Designer. The example shown in the article is based on [S60](#) UI View architecture. It shows how to develop a UI application similar to the native ?Settings? application that you can see in every [S60](#) phone.

Using Carbide UI Designer one can rapidly create S60 UI applications. [Olympio](#) has nicely elaborated rapid application development concept using Carbide UI Designer step by step. You can download test application from the article. [Read Article](#).

Wireless Information Sharing Engine

Week 30 - July 20th 2008

[Wireless Information Sharing Engine](#) by [Ebra](#) is available for the maemo and S60 platforms, enables file sharing between number of devices running the service. The number of devices is dynamic.
