

Series 60 Application Development

Details

Level:	Intermediate
Duration:	5 days
Development language:	C++
Experience:	Candidates should have attended the Symbian OS Essentials course or have equivalent working experience before attending this course (Active Objects, Descriptors, Client/Server). A good understanding of C++ and OO/UML is required. Familiarity with the essential basics of Carbide.c++ IDE is also beneficial.



Target audience

This course is for anyone who is considering or is currently involved in creating GUI applications for Series 60 using C++. The course introduces software engineers, designers and project managers to Series 60 development through lectures and practical exercises.

The course consists of approximately 50% exercises and practical work and 50% lectures. The exercises include C++ coding and use the Series 60 emulator to verify the results.

Description

The course starts by the examination of the framework architecture behind every Series 60 GUI application and of key elements of application design. We continue with the basics of creating UI controls and handle events like screens taps, key presses, changes of screens orientation (landscape/portrait) and redraw events. The essential Series 60 UI controls such as skins, menus and status panes are explored. The view architecture – a commonly used design for Series 60 application – is explained as well as dialogs, setting pages, forms, notes, editors, queries, lists and grids. Many S60 application will use graphics, animations and bitmaps. The course will present how to draw using a graphics context and how to use fonts and bitmaps. Images will need to be converted to bitmaps before displaying. The course will show how to use the Image Conversion Library to convert, rotate and scale images to bitmaps.

Objectives

- Learn to develop and deploy professional GUI applications for the Series 60 3rd and 5th Edition range of devices.
- Know what can be achieved using standard S60 UI components and what the limitations of a standard S60 UI design will be.
- Use common S60 UI components like menus, dialogs, setting pages, forms, lists, (text) editors.
- Localize S60 applications (L10N) and make applications ready for internationalization (I18N).
- Use the Image Conversion Library (ICL) to display, rotate, convert and scale images.

Tools and platform

During the course we will use the Nokia Carbide.c++ OEM v2 IDE environment. Carbide.c++ can be downloaded from Forum Nokia free of charge.

Exercises and course materials are adapted for the Series 60 5th Edition SDK emulator.

Agenda

Day 1

1. Introduction to Series 60 Platform

The S60 AVKON GUI library builds on top of Symbian OS. The main features of Symbian OS are summarized with focus on recent improvements (support for SMP; real-time kernel; Symbian Foundation). Further this section explains how Forum Nokia supports developers through all stages of the application life cycle. The Nokia Platform approach gives developers choice on which development environment to use including Flash Lite, Python, J2ME, C++ as well as browser technologies. The different releases of Symbian OS are related to releases of Series 60 to provide some historic context.

2. Getting Started

To get started developing for S60 you need to get familiar with the Carbide.c++ IDE. We create and import S60 applications. We will edit source files, change and inspect the INF and MMP files, build the application for the emulator and target as well as run and debug the application from Carbide.c++ in the emulator. Further a signed SIS (SISX) file is created and installed on the device and debugged remotely using Carbide.c++. Carbide.c++ also offers a S60 UI Designer.

3. Series 60 UI guidelines

The Series 60 principal characteristics including application suite, display sizes, user interaction paradigms and hardware requirements are introduced. A short overview of licensee customization options is given.

4. Application Architecture

The application architecture explains the GUI application startup sequence and reviews the source code required for the application, document, app UI and view classes. Applications can be designed using three common application architecture designs:

- Traditional Symbian OS Control Based Architecture
- Dialog Based Architecture
- Avkon View Switching Architecture

Further the role of the application registration file is explained and you will be able to give the application an caption and (SVG) icon.

Day 2

5. Windows and Controls

Controls are the basic elements for user interaction in the user interface. Controls are derived from `CCoeControl` and can accept user input and/or display output. Controls can be simple or compound. A window is a system resource that contains one or more controls. Symbian OS v9 provides a new way of creating controls and gives every control an unique handle. Controls can be window-owning or non-window-owning.

6. Event handling

An application will receive key events when it has focus. `CONE` distributes key events in an application to the appropriate controls which are on the control stack. Controls can consume key events in `OfferKeyEventL`. Applications may handle foreground events and switches between portrait and landscape mode (Scalable UI). Controls will receive system-initiated and application-initiated redraw events too. It is possible for controls to report events to another object. Any class can be setup as a control observer by deriving from `MCoeControlObserver`.

7. Skins

Skins provide a way of changing the appearance of the UI at runtime. Most Avkon (S60) controls are skin aware. You can make your own controls skin aware and receive notifications of skin changes.

8. Resource files

Resource files specify user-visible elements separately from the source code. This section presents a small introduction on how resource files are used in Series 60.

9. Menus

A menu is a window that presents a list of commands to the user. When a user selects a menu item the appropriate command is invoked and handled in `HandleCommandL`. Menus can have submenus (cascading menus) or can be context-sensitive. Menus can be defined in the resource file or dynamically changed in `DynInitMenuPaneL`.

10. Panes

Series 60 provides a set of standard panes that make up the Series 60 application window. The most important pane is the status pane which consists of a title pane (application title), the context pane (icon) and the navigation pane (tabs, labels, images and indicators).

11. View switching

Applications may register their screens (Avkon views) with a system wide View Server. The View Server coordinates activation and deactivation of views.

Day 3

12. Standard dialogs and forms

S60 provides a base class (CAknDialog) and resources to create simple and multi-page dialogs. Dialog data is validated and saved in OkToExitL(). The dialog is defined using the DIALOG resource. Flags and softkeys need to be defined here. Further the dialog class needs to be written and a RunLD method needs to be provided to construct and execute the dialog. Controls are initialized with data within PreLayoutDynInitL. Using CreateCustomControlL() can be added into a standard dialog. Forms are like dialogs, but can have many fields displayed in a list. The user can scroll up and down through the list of fields.

13. Notes

S60 offers wrapped notes for display of information notes, warnings, errors and confirmations. A wait note is used to inform the user that processing is taking place, but no indication is given of the time needed to complete the processing (a wait bar is shown). Progress notes are similar to wait notes, but a progress bar gives the user visual feedback of what portion of the processing is still left to do. Global notes are displayed even if the application that owns it is not in focus.

14. Queries

S60 provides a set of standard queries to ask the user a question. A response to a query may require the user to enter some data, select one or more items from a list or simply confirm an action. The confirmation query, data query, list query and global query are covered in this section.

15. List dialogs

List dialogs are available as a selection list dialog and a markable list dialog. A list dialog is defined in a DIALOG resource. Icons can be added dynamically for each item in the list.

16. Lists and grids

List basics cover the basic characteristics of lists within Series 60. S60 provides selection lists, menu (popup) lists, markable list (single selection) and multiselection lists. To search for items in a list a find pane can be added. List items have use a number of list style consisting of icons and one or two labels. Lists can be defined in the resource file or dynamically created from C++. Grids are similar to lists, but display items in rows and column. Settings lists are used to group together the user configurable settings for an application.

Day 4

17. Text editors

Text editors are feature-rich editors that can comprise of single or multiple lines of text. They have numerous configurable attributes such as the dimensions, input mode, input case, numeric keymap, special character table and other special properties. There are three main types of text editors:

- Plain editor
- Global editor (global formatting of characters and paragraphs)
- Rich text editor (individual formatting of characters and paragraphs)

18. Other editors

S60 supplies three different editors for numeric data:

- Integer editor
- Floating point editor
- Fixed point editor

Secret editors are used to securely obtain PINs or passwords from the user. Multi-field numeric editors (MNFE) are used for entry of multiple numeric fields into a single editor, e.g. IP address editor, range editor, time, date, time and date, duration and time offset editor.

19. Internationalization (I18N) & Localization (L10N)

The goal of internationalization is to make it easy for your application to transcend not languages exactly, but locales. A locale consists of a base language plus differences like currencies and date formats. All user visible text in the resource files need to be localized for the languages supported by the application. Languages can be bi-directional like Arabic or Hebrew and may require Unicode (e.g. Chinese).

Day 5

20. S60 Graphics Architecture

The graphics architecture focuses primarily on interactions between the Window Server, Font and Bitmap Server and Multi Media Server.

21. Basic Drawing

Basic drawing is provided by the base class `CCoeControl` for each control. The system graphics context performs various drawing functions and has attributes like pen and brush. The graphics context is used to draw shapes like lines, rectangles, ellipses, arcs, pies and polygons. The graphics context is also used to draw text. Each control has a font provider. Font have measurements like base line, ascent, descent, height, width, left and right adjust as well as styles. Using text effects we can draw text in different colors or orientation.

22. Bitmaps and animations

Bitmaps and SVG images are stored in a multi-bitmap file (.mbm or .mif) can be loaded using `CFbsBitmap` and draw on the screen using `DrawBitmap()` or `BitBlit()`. Bitmaps have a mask to indicate the transparent areas of the bitmap. S60 provides a DLL framework for animations. These animations can run as a high-priority thread within the Window Server. An alternative approach to animation is to use double buffering and off screen bitmaps and perform the animation using client-side code.

23. Direct Screen Access

For games and other graphics intensive applications Series 60 provides Direct Screen Access (DSA). The application can request authorization from the Window Server to draw directly into a certain screen area. It is essential to only apply Direct Screen Access when the application has moved into the foreground.

25. Image manipulation

Images are not always available as bitmaps. Before displaying the image it needs to be converted to a bitmap. This module looks at decoding and encoding using the Image Conversion Library and active objects. Series 60 provides classes for rotating and scaling the image before conversion if needed.