Symbian C++ Hello World

This tutorial shows you how to create a skeleton application using the Carbide.c++ IDE and UI designer, and how to get it up and running on the Windows-hosted Symbian Emulator.

This article needs to be updated: If you found this article useful, please fix the problems below then delete the {{ArticleNeedsUpdate}} template from the article to remove this warning.

Reasons: hamishwillee (25 Aug 2011)
Need to verify images and instructions for current version of carbide - this was done for version in ADT and a lot has moved on since then. Fundamentally the article should be OK though, so this is a minor change

Starting Carbide.c++

The Carbide.c++ IDE is the only supported/official IDE for Symbian C++ development. It is installed from here in the Nokia Developer Website Symbian C++ Tools section.

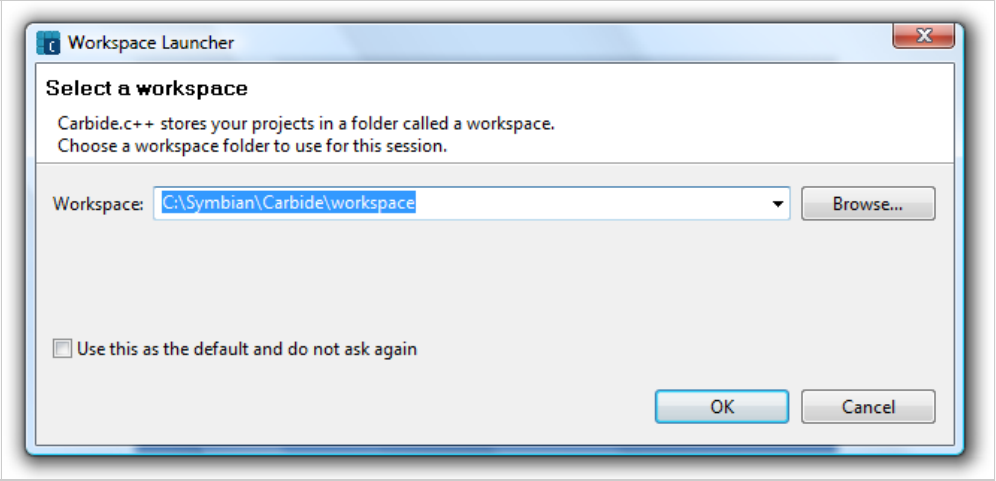
Carbide.c++ is launched from the Windows **start** button:

- **All Programs | Symbian Foundation ADT v**<*ADTVersion*>**| Carbide.c++ | Carbide.c++ v**<*CarbideVersion*>

On start, you will be prompted to select a workspace directory. The workspace directory contains any projects you've already created in the workspace and their common settings - such as code-formatting options (you can define multiple workspaces in order to separate completely different tasks). If this is the first time you've run Carbide.c++ the workspace will be empty.

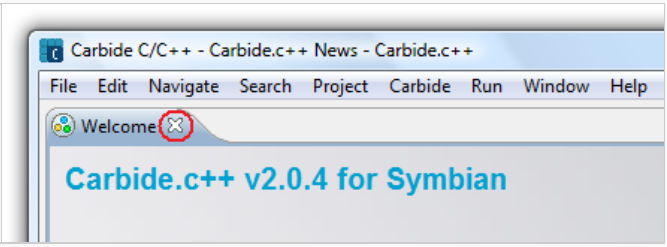
Note: Your Symbian projects must be on the drive where you installed your SDK. You must also ensure that the path name of the workspace does not contain non-alphanumeric characters or spaces. This is because the Symbian toolchain uses command line tools that cannot read special path names.

If you installed the SDK to drive `c:\`, an example of an acceptable workspace path is: `c:\Symbian\development\`.



Specify the workspace location

Once Carbide.c++ has started, close the *Welcome tab* (by clicking the cross shown circled in red below) to see the default workspace.



Carbide welcome screen

Updating Carbide.c++

ADT v1.4 contains Carbide.c++ 2.3. You can update to the latest version of Carbide.c++ (at time of writing 2.6) within the IDE (only). Note, this is not mandatory, but there have been many defect fixes and improvements between versions.

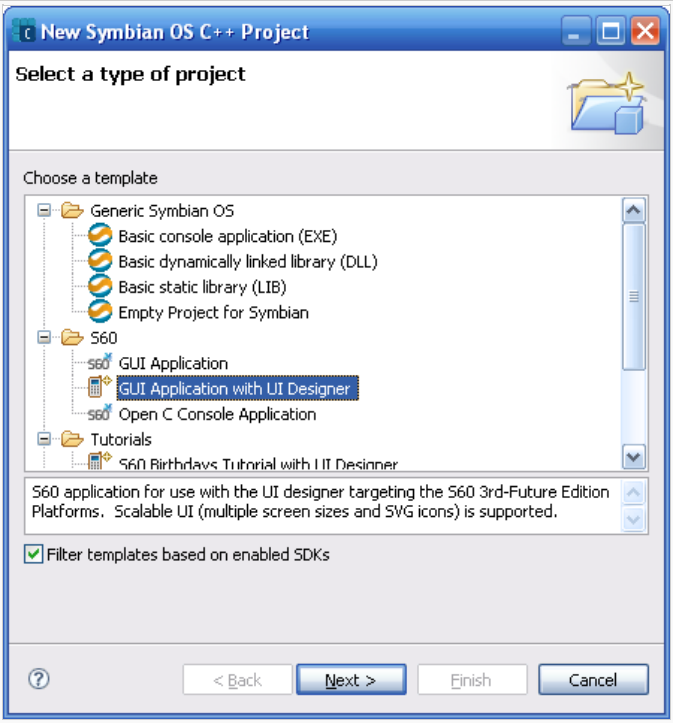
(C) Copyright Nokia 2014. All rights reserved.

1. Click Help > Check for Updates
   - Install all components
   - You can restart the IDE if you wish
2. Click Help > Install New Software
   - Select the Carbide Update in the pull down menu
   - Select the 2.6 ADT Features
   - Restart the IDE

Creating a Project

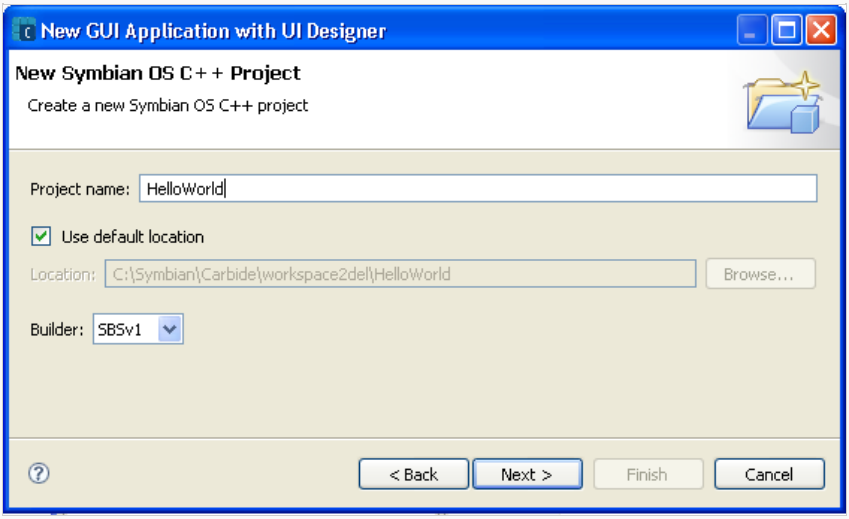
To launch the Carbide.c++ *Create New Project Wizard* select: **File | New | Symbian OS C++ Project**.

Choose the **GUI Application with UI Designer** application template (in the "S60" section). This template creates a runnable GUI application with the *UI Designer* tool enabled (the *UI Designer* allows views to be created "visually" by drag & dropping standard UI components).



Select project template

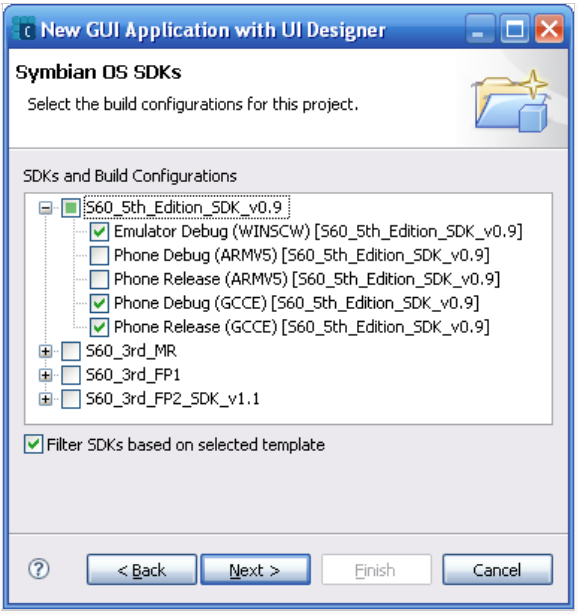
The **Next** page of the wizard is "**New Symbian OS C++ Project**". Define the project name - in this case "HelloWorld". Once again, make sure the project directory is on the same drive as the SDK and does not contain spaces or other special characters.



New project dialog

The **Next** page of the wizard is "**Symbian OS SDKs**". Choose the SDK(s) you want to use for building the project from among those installed to your PC (You can add more SDKs to your project later on). This should include the Symbian^1 SDK (**Note:** this is a copy of the S60 5th Edition SDK v1.0).

(C) Copyright Nokia 2014. All rights reserved.



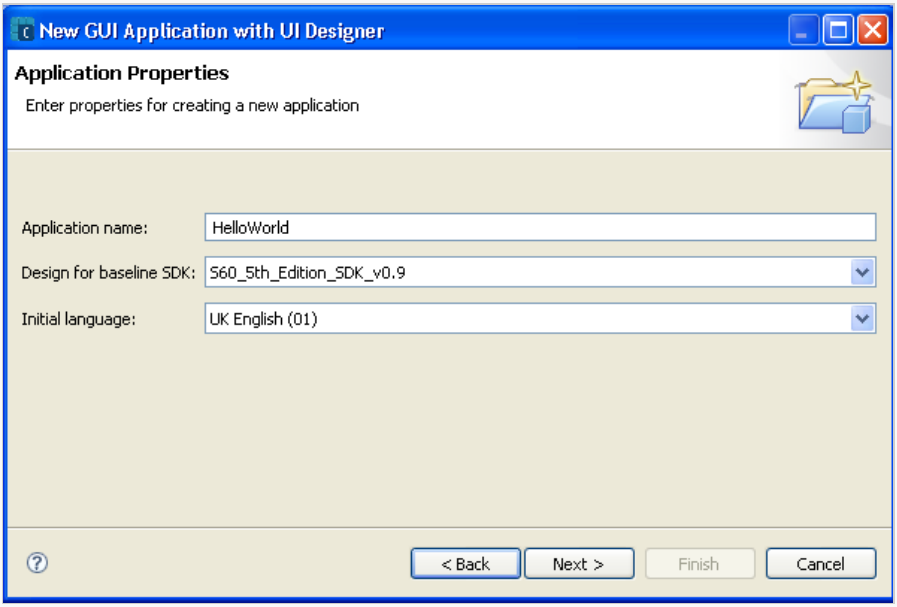
Select target SDK

By default all build configurations will be selected:

- **Emulator Debug (WINSCW)** builds binaries for the Windows-hosted emulator.
- **Phone Debug | Release (GCCE)** builds binaries for the phone using the (free) GCCE compiler that was installed with the SDK.
- **Phone Debug | Release (ARMV5)** builds binaries for the phone using the ARM RealView Compiler (RVCT). RVCT produces code that is a few percent smaller and faster than the current versions of GCCE supported for Symbian C++ development, but must be separately licensed from ARM. RVCT is primarily used by phone manufacturers to build binaries for device ROM.

The Emulator is needed by all developers, and GCCE is sufficient for most third-party development. Therefore most developers should simply de-select the ARMV5 options as shown above.

The **Next** page of the wizard sets the "**Application properties**".



Set app properties

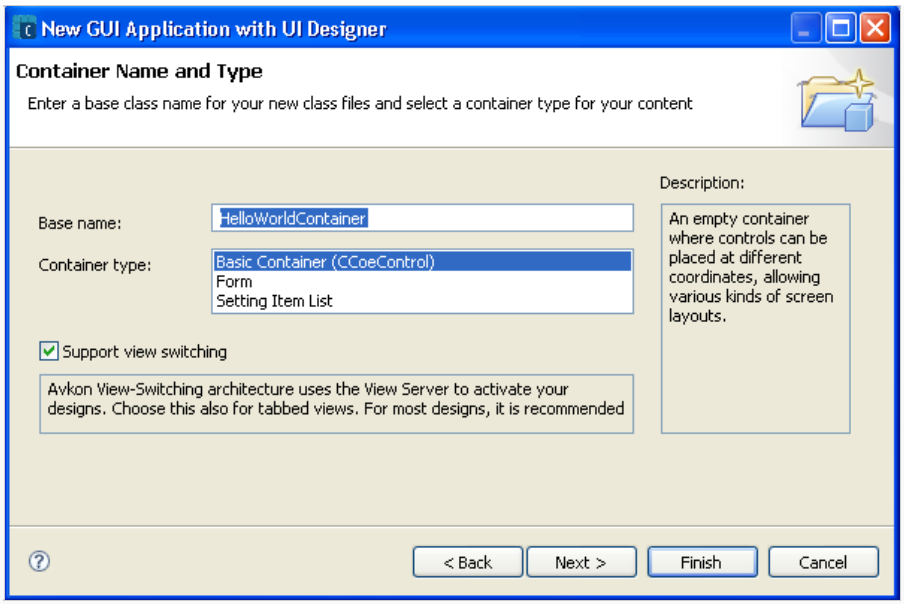
Keep the default values. The **Baseline SDK** defines the common compatibility level of your application. For example, if set to the **S60 5th Edition SDK**, the UI designer doesn't let you add elements that are only available in later S60 5th Edition, Feature Pack 1+ SDKs and devices.

The **Next** page of the wizard allows you to choose an S60 UI design. Keep the **Empty** user interface variant.

The **Next** page of the wizard "**Container Name and Type**" has default values that are usually acceptable.

Note: The **Finish** button is active in this page. The next page is optional but for the purposes of this tutorial you should click **Next**, rather than **Finish**.

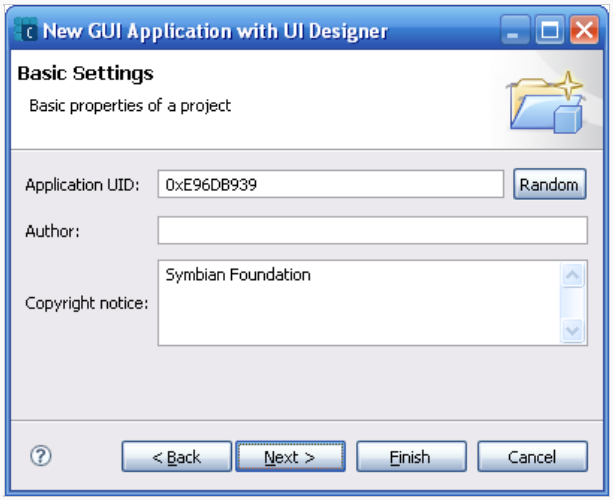
(C) Copyright Nokia 2014. All rights reserved.



Select GUI container type

Even though our application will only use one view, it's usually better to check "Support View Switching", in order to make the application easier to extend later on.

The **Next** page of the wizard "**BasicSettings**" can be used to customize the project with your name and a copyright notice for every human-readable file. It also allows you to specify the application unique identifier (UID).



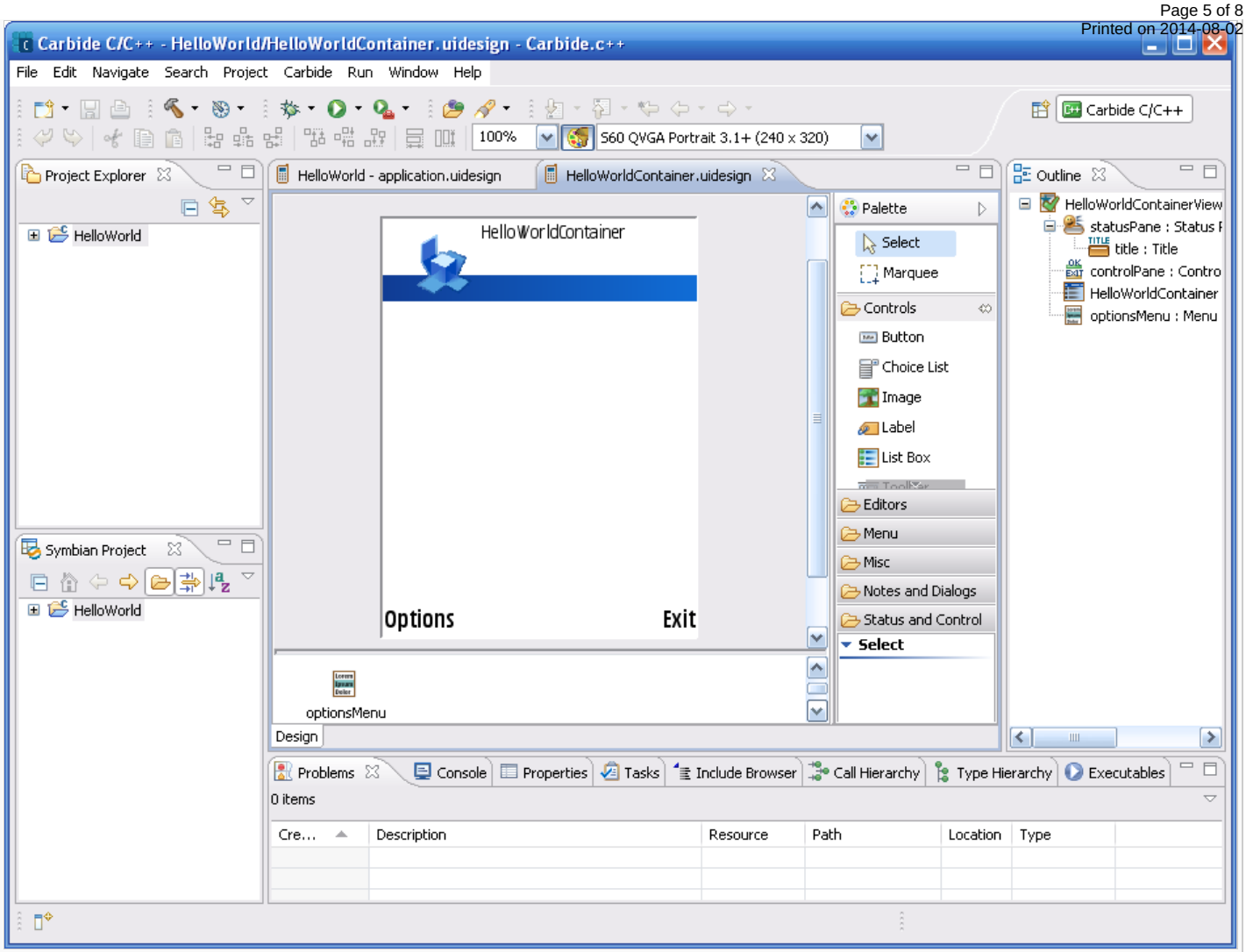
Basic Settings

Carbide.c++ generates a random UID value for you starting with `0xE`, which is the range of UIDs reserved for internal development and testing. If you want to release your application to the public, you need to get your own unique UID allocated by Symbian Signed. As we do not intend to release our Hello World application to the public, we'll simply continue to use the value Carbide.c++ assigned us from the development range (you can change the UID later on, although you must be careful to change every instance of it found within your project.

Select **Finish** to close the wizard and create your application (there is no need to go to the **Next** (last) page of the wizard as the default values are always acceptable).

That's it, your workspace should look similar to the screenshot below.

(C) Copyright Nokia 2014. All rights reserved.



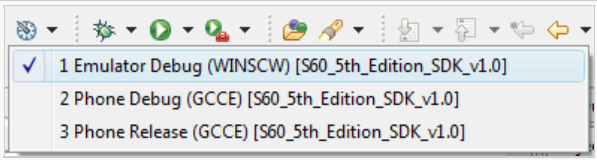
Workspace view

Targeting the Emulator

Normally you'll start by building for the emulator; you can use the emulator for most of your development work (it is possible to access the Internet through the emulator, and even simulate GPS).

Building for the Emulator

- First set the active build configuration. You can do this by clicking the **Manage configurations for the current project** icon in the toolbar or by selecting menu: **Project | Build Configurations | Set Active** and select **Emulator Debug**.



Select active build configuration

- Then build the current configuration using the **Build** icon in the toolbar or through the menu: **Project | Build Project** (You can also select a particular configuration to build from the Build icon selector).

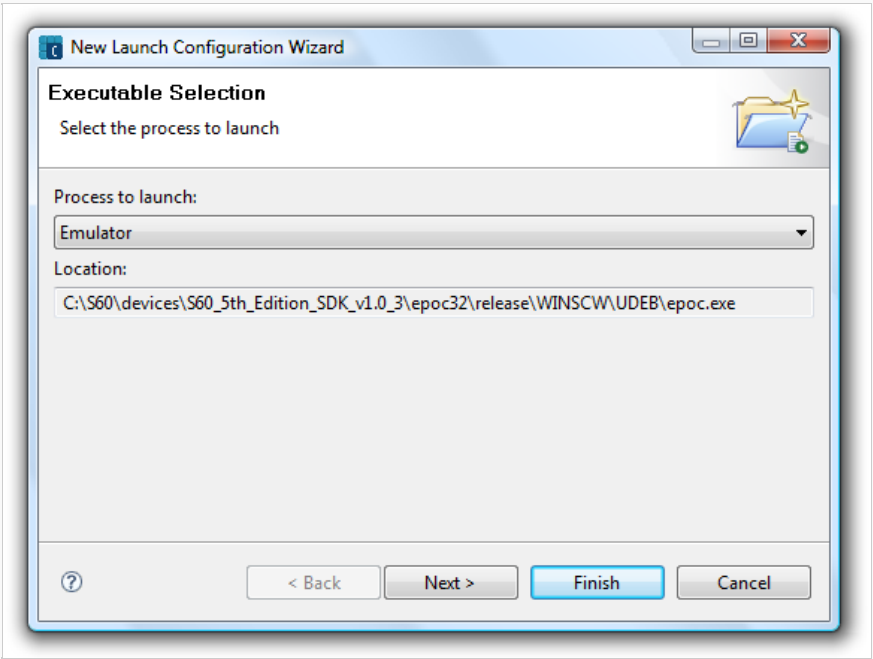
Warning: If you get an error message similar to "**WARNING: EPOCROOT does not specify an existing directory**", you did not place your workspace/project on the same drive as the application development SDK (which is installed by default to C:\). Delete the project and start again.

Expand the thumbnail for how this might appear in Carbide.c++:

Running on the Emulator

(C) Copyright Nokia 2014. All rights reserved.

If your application built successfully, click on the **Run** button (Ctrl + F11). Upon the first launch of your project, Carbide.c++ will ask you which executable you want to launch:



Executable selection

- If you choose *HelloWorld.exe*, the the emulator will be launched and your application started automatically. The emulator will close once you exit your application.
- If you choose *Emulator* the emulator (epoc.exe) will be launched and you will need to navigate to the application and start it by clicking on the icon (just as you do when starting an application on a device). We'll explain how to find the application shortly.

It may sound more difficult to choose the second method, but it has some advantages. You can leave the emulator running if you are only doing small edits in your source code – simply close your application in the emulator, recompile and restart your app through the emulator's menu. You'll also see any error messages that may be shown when you exit the application, because the emulator will not shut down instantly after you exit Hello World in the emulator. Those error messages are also visible in Carbide.c++'s console window.

When the emulator starts for the first time you might have to wait for several minutes before it is completely ready for use. Successive starts will be a lot faster, because Windows caches most of the emulator DLLs.

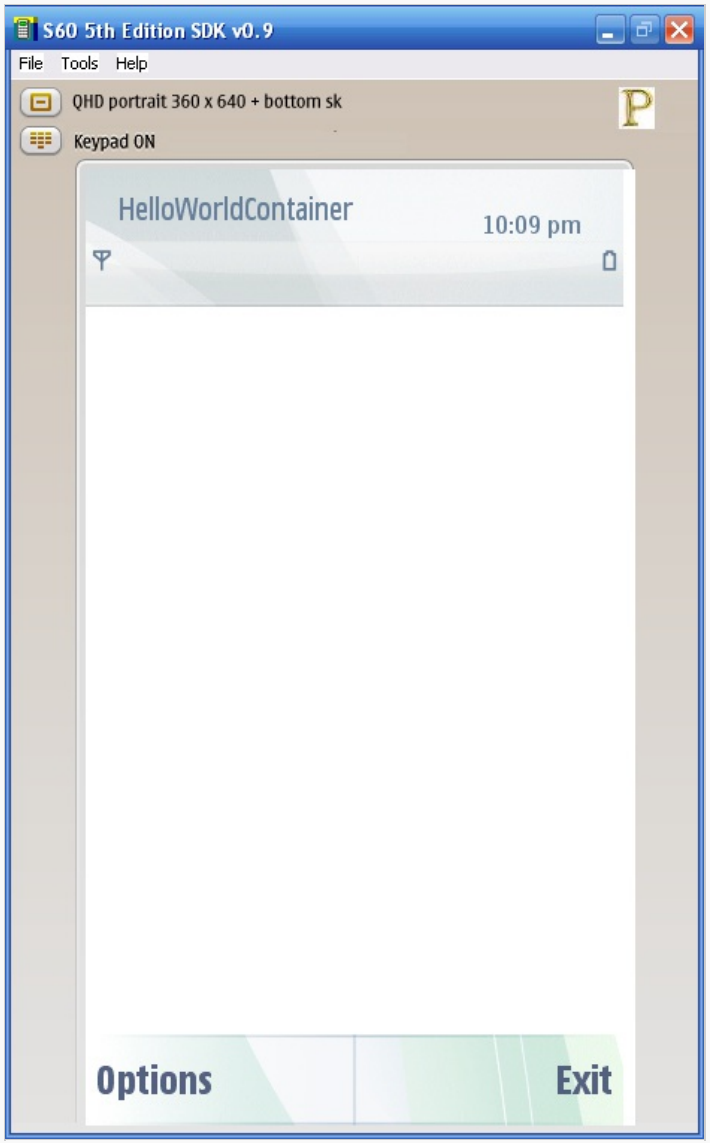
The GUI "Menu" symbol

If you decide to launch the emulator and navigate to your application: First, open the menu through the **S60** menu symbol on the bottom left of the screen. Your own application will be located at the bottom of the **Applications** folder; use your mouse to navigate in the emulator's menus.

When you launch your application, the main pane will be empty. As you can see, the example application the wizard has created

(C) Copyright Nokia 2014. All rights reserved.

already supports a full user interface and handles input from the left and right softkeys at the bottom of the screen.



Hello World main screen

Debugging on the Emulator

The Emulator is the default debug target - you simply click the **Debug** button .

Debugging on the Emulator is not covered further in this tutorial. See **Carbide.c++ User Guide > Debugging projects** for extensive information on debugging using Carbide.c++.

Summary

This tutorial demonstrated you how to create a basic "Hello World" application using a Carbide.c++ IDE wizard, and how to get it up and running on the Symbian Emulator.

The next step on the trail is deploying the application to a device.

© 2010 Symbian Foundation Limited. This document is licensed under the Creative Commons Attribution-Share Alike 2.0 license. See http://creativecommons.org/licenses/by-sa/2.0/legalcode for the full terms of the license.

Note that this content was originally hosted on the Symbian Foundation developer wiki.

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.