K

Test Driving a C++ Program on OS X

In Section 1.10, we showed how to run and interact with a C++ application on Windows and Linux, respectively. Here, we introduce how to run and interact with the same C++ application on OS X. You'll begin by running an entertaining guess-the-number game, which picks a number from 1 to 1000 and prompts you to guess it. If your guess is correct, the game ends. If your guess is not correct, the application indicates whether your guess is higher or lower than the correct number. There is no limit on the number of guesses you can make. [*Note:* For this test drive only, we've modified this application from the exercise in Chapter 6, Functions and an Introduction to Recursion. Normally this application randomly selects the correct answer as you execute the program. The modified application uses the same correct answer every time the program executes (though this may vary by compiler), so you can use the *same* guesses we use in this section and see the *same* results as we walk you through interacting with your first C++ application.]

We'll demonstrate running the C++ application in Xcode on OS X. You'll run the application and enter various numbers to guess the correct number. The elements and functionality that you see in this application are typical of those you'll learn to program in this book. We use fonts to distinguish between features you see on the screen and elements that are not directly related to the screen. We emphasize screen features like titles and menus (e.g., the **File** menu) in a semibold **sans-serif Helvetica** font and emphasize filenames, text displayed by an application and values you should enter into an application (e.g., `GuessNumber` or `500`) in a `sans-serif Lucida` font. For the figures in this section, we point out significant parts of the application.

Running a C++ Application Using Xcode on OS X

For this test drive, we assume that you know how to copy the examples into your `Documents` directory. Please see your instructor if you have any questions regarding copying the files to your system.

1. ***Checking your setup.*** It's important to read this book's Before You Begin section to make sure that you've copied the book's examples to your hard drive correctly.
2. ***Opening Xcode.*** Open Xcode (located in the `Applications` folder) on your Mac. If you don't have Xcode installed, go to the Mac App Store where you can download it for free. You can find the App Store shortcut on the dock at the bottom of your screen. If you're running Xcode for the first time, it will ask you to download additional software components, which you should complete before proceeding.

3. *Creating a Project.* In Xcode, select **File > New > Project** to display the **Choose a template for your new project** sheet (Fig. K.1). In the **OS X** section at the left side, select **Application** then select **Command Line Tool** in the right side of the sheet. A command line tool is an app that you can execute from a **Terminal** window. For demonstration purposes, we'll execute this app directly in Xcode. Click **Next**.

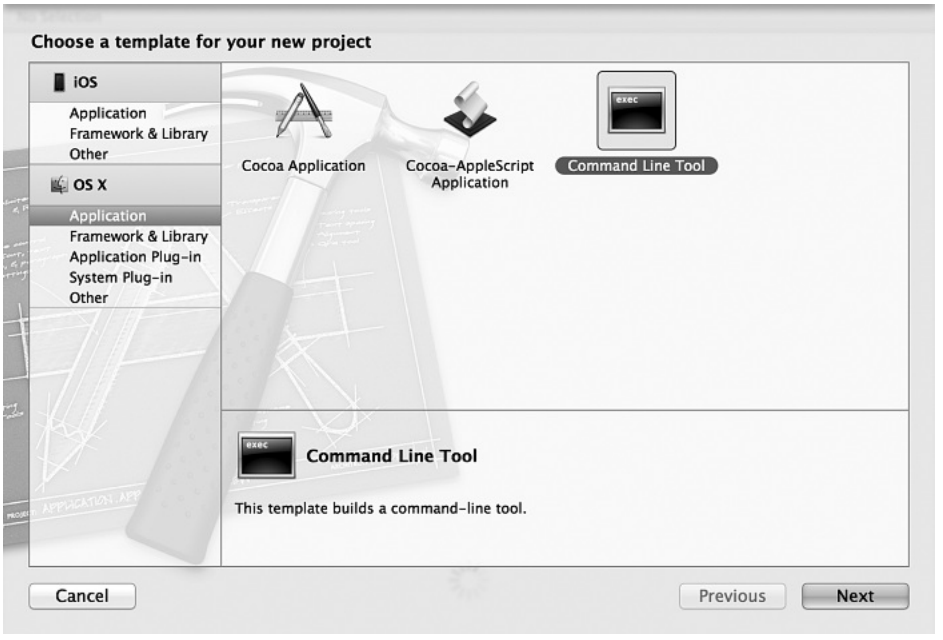


Fig. K.1 | **Choose a template for your new project** sheet.

4. *Setting the Project's Options.* In the **Choose options for your new project** sheet (Fig. K.2), specify `GuessTheNumber` for the **Product Name**, your name for the **Organization Name** and `self.edu` for the **Company Identifier**, then select C++ from the **Type** drop-down list and uncheck **Automatic Reference Counting**. The settings we've asked you to provide here are for testing purposes. (Proper settings for placing applications in the Mac App Store are beyond the scope of this book.) Click **Next**.
5. *Saving Your New Project and Examining the Workspace Window.* Choose where you'd like to save your project on your computer, then click **Create** to display the new project's window—known as a **workspace window** (Fig. K.3)—which is divided into four main areas below the toolbar: the **Navigator area**, **Editor area**, **Utilities area** and the **Debug area** (which is *not* initially displayed). In the **Navigator** area, the **Project** navigator ( ) is displayed by default—it shows all the *files and folders* (folders are known as *groups* in Xcode) in your project. When you select a C++ source code file in the **Project** navigator, it will be displayed in the **Editor** area so you can edit the code. At the right side of the workspace window is the **Utilities** area, which displays **inspectors** that allow you to view and edit information about items displayed in the **Editor** area. By default, the top half of the **Utilities** area shows

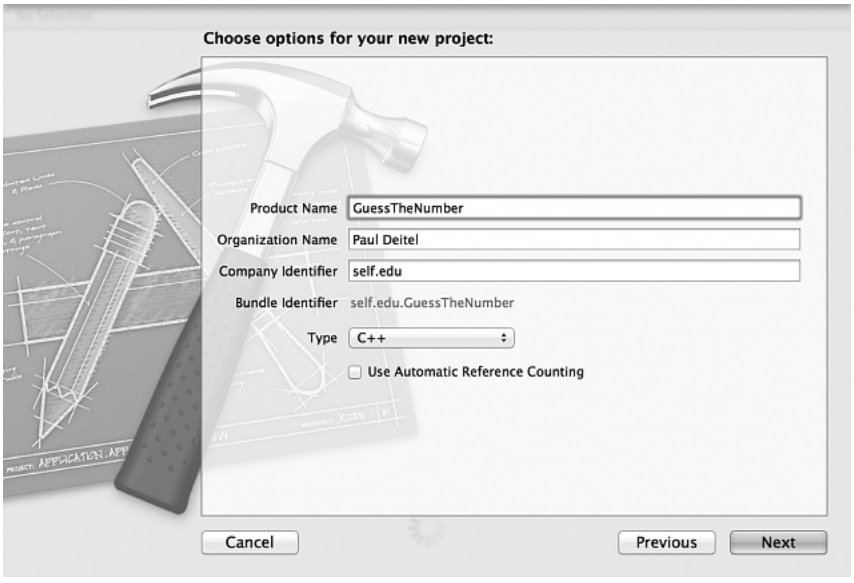
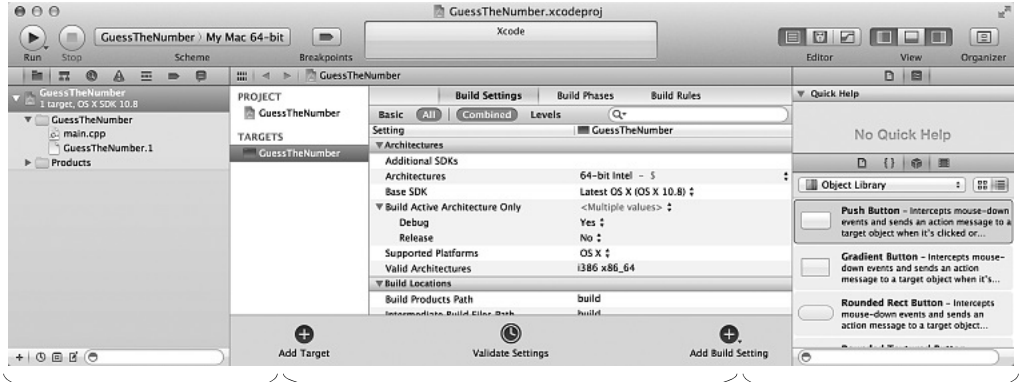


Fig. K.2 | **Choose options for your new project** sheet.

either the **File** inspector ( ) or the **Quick Help** inspector ( ). The **File inspector** shows information about the currently selected file in the project. The **Quick Help inspector** provides *context-sensitive help*—documentation that's based on the *cursor position in the source code*. For example, *clicking in a C++ Standard Library function name* shows a *description* of the function. When displayed, the **Debug** area (discussed in Appendix J) appears at the bottom of the editor area and provides controls for *stepping through code*, *inspecting variable contents* and more.



Navigator area occupies the left column

Editor area occupies the center column

Utilities area occupies the right column

Fig. K.3 | `GuessTheNumber.xcodeproj` workspace window.

6. *Deleting* `main.cpp`. In this test drive, you're going to use an existing C++ source code file named `GuessNumber.cpp`. For this reason, you'll delete the `main.cpp` that Xcode provided by default. To do so, right click `main.cpp` in the **Project** navigator and select **Delete**. In the dialog that appears, click **Move to Trash** to delete the file.
7. *Adding* `GuessNumber.cpp` *to the Project.* To add `GuessNumber.cpp` to the project, right click the folder `GuessTheNumber` in the **Project** navigator and select **Add files to "GuessTheNumber" ...**. In the dialog that appears, navigate to the `Documents/examples/ch01/GNU_Linux` folder, select `GuessNumber.cpp`. Ensure that **Copy items into destination group's folder (if needed)** is selected. In the **Add to targets** area of the dialog, ensure that **GuessTheNumber** is checked. Click **Add** to add the `GuessNumber.cpp` source code file to your project. The file should now appear in the **Project** navigator.
8. *Compiling and Running the* `GuessNumber` *application.* To run an application, you must first compile it. When you click the **Run** (▶) button on the workspace window's toolbar, Xcode compiles the code. If there are no errors, Xcode then runs the application. Because this application was set up as a **Command Line Tool**, it executes in the **Debug** area (Fig. K.4) below the **Editors** area of the workspace window. The application displays `"Please type your first guess."`, then displays a question mark (?) as a prompt on the next line.

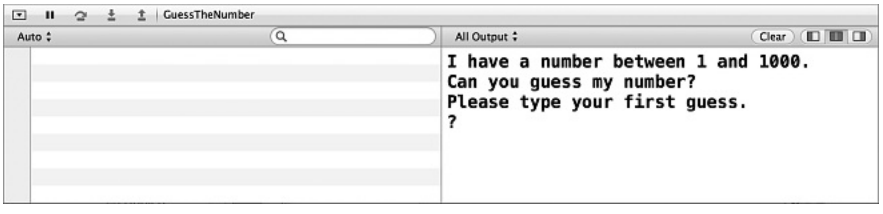


Fig. K.4 | Application running in the **Debug** area.

9. *Entering your first guess.* At the ? prompt, enter **500** (Fig. K.5) and press *Return*.

```
I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
? 500
Too low. Try again.
?
```

Fig. K.5 | Entering an initial guess.

10. *Entering another guess.* The application displays `"Too low. Try again."`, meaning that the value you entered is less than the number the application chose as the correct guess. At the next prompt, enter **750** (Fig. K.6). The application displays `"Too low. Try again."`, because the value you entered is less than the correct guess.

```
I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
? 500
Too low. Try again.
? 750
Too low. Try again.
?
```

Fig. K.6 | Entering a second guess and receiving feedback.

11. ***Entering additional guesses.*** Continue to play the game (Fig. K.7) by entering values until you guess the correct number. When you guess correctly, the application displays `"Excellent! You guessed the number."`

```
I have a number between 1 and 1000.
Can you guess my number?
Please type your first guess.
? 500
Too low. Try again.
? 750
Too low. Try again.
? 875
Too high. Try again.
? 813
Too high. Try again.
? 781
Too low. Try again.
? 797
Too low. Try again.
? 805
Too low. Try again.
? 809
Too high. Try again.
? 807
Too low. Try again.
? 808

Excellent! You guessed the number!
Would you like to play again (y or n)? n
```

Fig. K.7 | Entering additional guesses and guessing the correct number.

12. ***Playing the game again or exiting the application.*** After you guess the correct number, the application asks if you'd like to play another game. At the `"Would you like to play again (y or n)?"` prompt, entering the one character **`y`** causes the application to choose a new number and displays the message `"Please type your first guess."` followed by a question mark prompt so you can make your first guess in the new game. Entering the character **`n`** ends the application. Each time you execute this application from the beginning (i.e., *Step 8*), it will choose the same numbers for you to guess.