

Introduction to REALbasic

“Part 2: Getting to Know REALbasic”

by Wally Wang

Last week, “Part 1: The Roots of REALbasic” introduced this four-part series.

To learn how to program any computer, you essentially need to learn three different tasks:

- How to identify a problem and define a solution
- How to use a programming editor and compiler to create a program
- How to use the syntax of a specific programming language

The first, and most crucial task of programming is knowing the right problem to solve. If this sounds obvious, it's not. Government agencies and Fortune 500 companies have all been guilty of spending millions of dollars and years of programming—only to discover that nobody knew what problem they needed to solve in the first place. If you don't know what you want your program to do, you'll never be able to write a program that can do it.

Every program solves a problem by performing a specific task. A missile-control program solves the problem of hitting a target by aiming a missile as it flies. An inventory-management program solves the problem of tracking products stored in a warehouse. Even a video game solves the problem of entertaining someone by displaying animated characters on the screen that you can manipulate.

After you can clearly identify the specific problem to solve, you need to decide how you want your program to solve it. The specific steps that your program follows to solve a problem is called an algorithm.

Here's the secret about computer programming and problem-solving in general—there are literally an infinite number of ways to solve the exact same problem. The first step is to find any way to solve a problem. The second step is finding the simplest way to solve that problem.

For example, suppose your problem is how to add a series of numbers, such as the numbers 1 through 10. One way to solve this problem is to simply add each number in consecutive order, such as:

$$1+2+3+4+5+6+7+8+9+10 = 55$$

A less straightforward, but equally effective way to solve this exact same problem is to use the following formula:

$$(L^2 - F^2 + F + L) / 2$$

where F is the first number of the series (1 in this example)

L is the last number of the series (10 in this example)

So using this formula looks like this:

$$\begin{aligned} &= (102 - 12 + 1 + 10)/2 \\ &= (100 - 1 + 11)/2 \\ &= (99 + 11)/2 \\ &= (110)/2 \\ &= 55 \end{aligned}$$

Notice that this formula calculated the same answer, but in a much less intuitive manner. Once you understand that there is more than one way to solve the same problem, you can choose the best method. This best method may be the one that's the easiest to follow and understand (such as adding all numbers consecutively), or that's easiest to implement in a programming language (such as using a formula).

Once you understand that programming is nothing more than problem-solving, you can start writing your solutions down in step-by-step fashion like a recipe, or what computer scientists call an algorithm. An algorithm simply tells the computer exactly which steps to follow in which order to solve a problem.

After you have identified both your problem and a solution for solving that problem, you're ready to start converting your solution into an actual computer program.

You can write a program in any programming language, just as you can say the phrase, "How are you?" in any human language from Spanish and Arabic to Swahili and Chinese. The specific programming language you use can determine how simply or elegantly you can create your program.

Every programming language is designed to solve different types of problems. The C language is designed for maximum efficiency, which is why the language uses lots of strange symbols and commands that resemble cryptic abbreviations. In REALbasic, the emphasis is on helping you create a program quickly using commands that resemble English, so you can write, edit, and understand them easily. As a result, writing and editing a program in REALbasic is much easier than writing that same program in C.

To write a program in REALbasic, you need to get familiar with the REALbasic IDE (Integrated Development Editor), which is just a fancy term for the REALbasic user interface.

At first glance, the REALbasic interface might look intimidating and confusing, but you'll see that it's actually pretty simple once you understand the purpose of each part, as shown in Figure 1.

The three main parts of REALbasic are:

- The Toolbox
- The Window
- The Properties palette

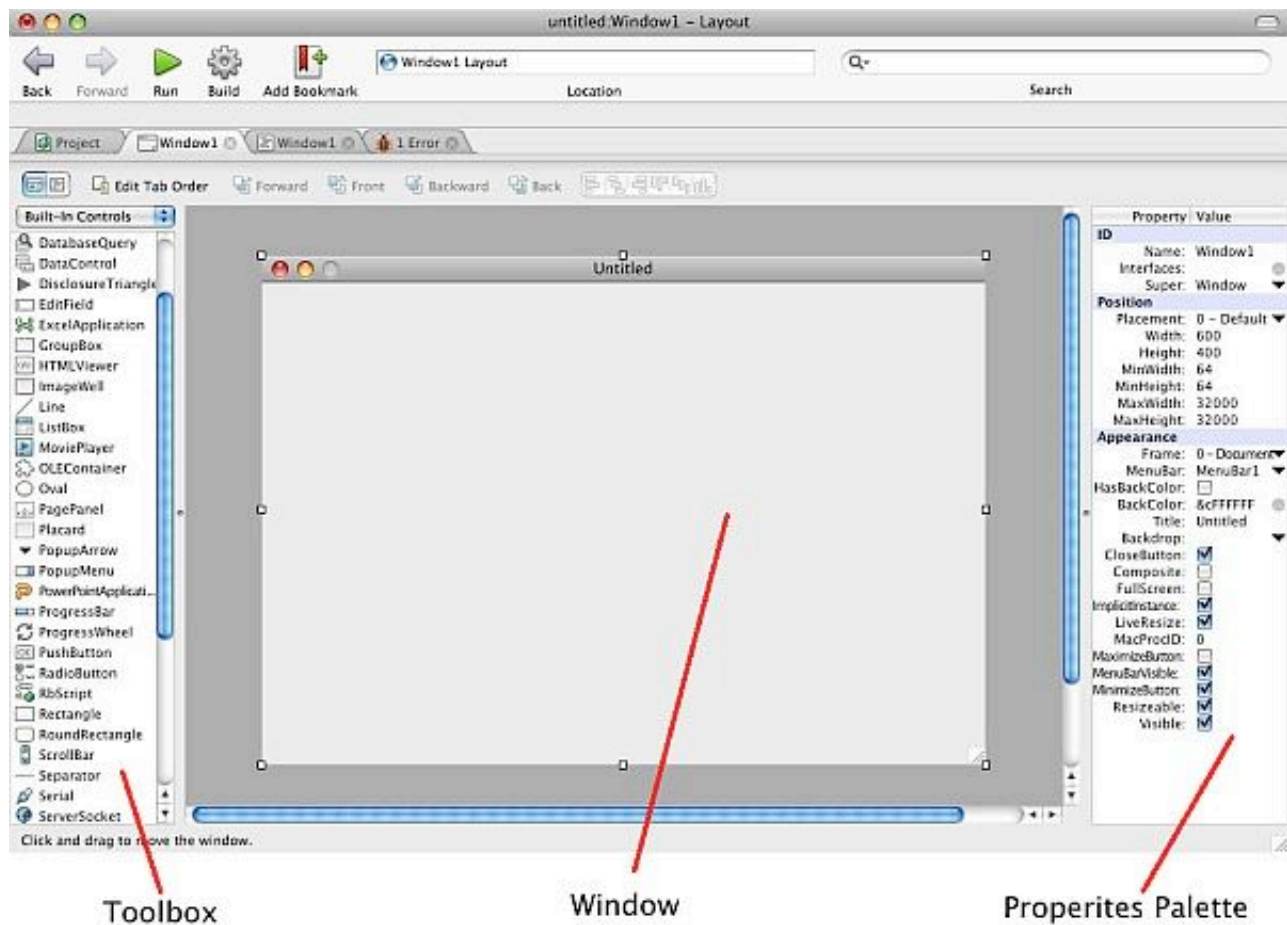


Figure 1. The Toolbox displays the parts of a user interface.

The Toolbox contains icons that represent different parts that make up a typical user interface. Some of the more common user-interface parts are buttons, check boxes, text and scroll bars.

Writing a program in REALbasic involves a three-step process:

1. Design your user interface.
2. Customize your user interface.
3. Write BASIC commands to make your user interface do something.

In REALbasic, the basic idea is that you first create your user interface (all the windows, buttons and other stuff that the user sees on the screen). When your user interface looks nice and pretty, you'll wind up with a generic user interface consisting of buttons or check boxes that you'll need to customize for your particular program.

The most common way to customize the parts of a user interface is to type the names of commands directly on different parts, such as typing "Cancel" on a button or "Male" or "Female" on two adjacent radio buttons.

After customizing your user interface, the final step is to write BASIC commands to make your user interface actually do something.

Designing a User Interface

The window, which appears in the middle of the REALbasic interface, represents the actual window that your program displays on the screen. Initially, this window will be blank. Since displaying a blank window is relatively useless, you'll need to place objects on that window that let the user do something.

These objects, also called controls, appear in the left side of the screen, grouped together in the Toolbox. By clicking and dragging an icon off the Toolbox, you can place and resize it on your program window.

Some common types of controls in the Toolbox include:

- Check boxes
- Push buttons
- Radio buttons
- Text boxes
- Scroll bars
- Sliders

After you place a control on the window, you can drag the control on the window using the mouse until it appears exactly where you want it. Do this with enough controls, and you'll wind up with a generic user interface, as shown in Figure 2.

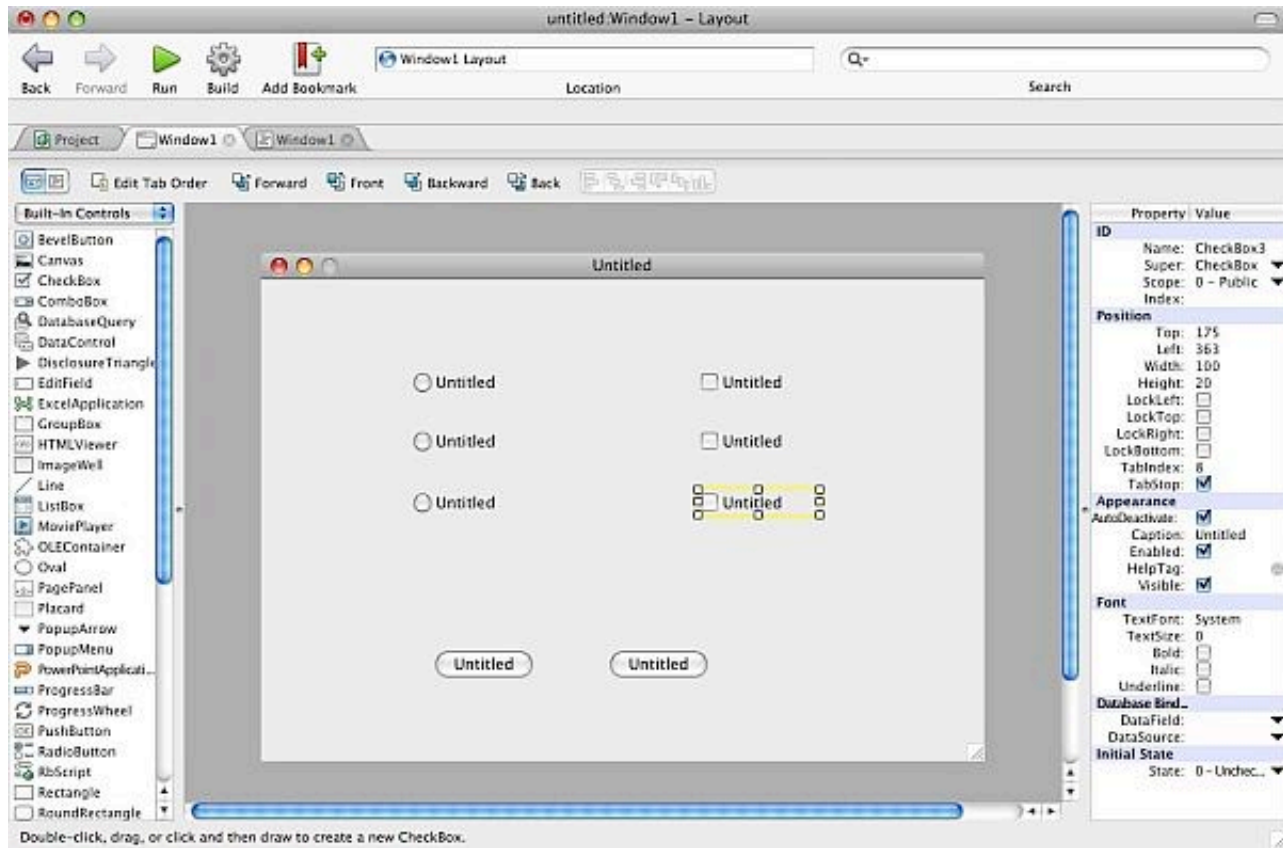


Figure 2. Creating a generic user interface by placing controls on a window.

Once you've created your user interface, you'll need to customize it so it displays information specific to your program, such as buttons that display labels like OK or Cancel. To modify a control, you'll need to change that control's properties using the Properties palette.

Properties define how a control appears. One common property is the Caption property, which displays text on a control. When you first place a control on a window, its Caption property will be Untitled. If you want your control to display the word "OK," you'll need to change its Caption property.

Every type of control has different properties, but most controls share some of the following properties:

- Caption – defines the text that appears on that control
- Name – defines the name of the control
- Top – defines the position of the control from the top of the window
- Left – defines the position of the control from the left side of the window
- Width – defines the width of the control
- Height – defines the height of the control

By default, every control uses a generic name, such as PushButton1 or RadioButton3. The name of a control is important when you start writing actual program commands to make your user interface do something.

Although every control offers a half dozen or more properties, you don't need to change all (or any) of them. However, you'll probably want to change the Caption property to display unique text other than Untitled.

Writing BASIC Commands

After modifying the properties of your controls, the next step is to write actual BASIC commands to make your program work. Right now, your program might look nice but it won't do anything. If you run your program and click on any controls, nothing will happen.

To make your program work, you need to write BASIC commands. In the old days, writing a program meant creating a massive list of commands. In REALbasic, you simply write short programs, called event procedures, which tell that particular control what to do when an event occurs on it. The most common event is called the Action event, which occurs when someone clicks the mouse over that control.

To create an event procedure for any control, you must double-click on that control to open a code editor for that control, as shown in Figure 3.

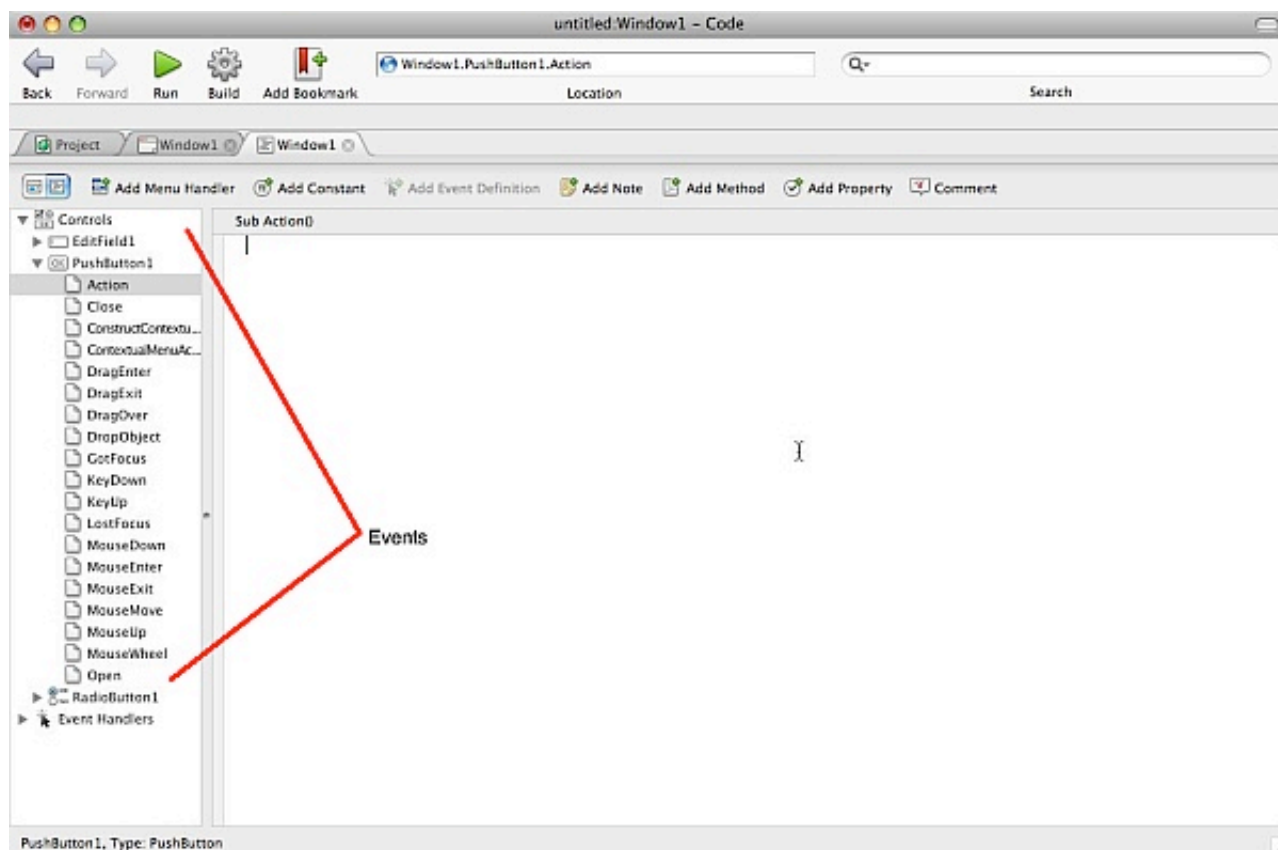


Figure 3. Double-clicking on a control opens a code editor for writing BASIC commands.

The left side of the editor window displays a list of different events your chosen control can respond to. For a push-button control, the most common event is the Action event, which occurs whenever

the user clicks on that push button. Other types of events include DragOver (when the user drags something over a control) and MouseEnter (when the user moves the mouse pointer over a control).

If you want to switch to a different event procedure, just click on that event, such as clicking on MouseDown or Open. The top of the editor window displays the name of the event.

The general idea behind REALbasic is to design your user interface, customize it, and then write BASIC commands to make your program do something. Next week, we'll go through the steps in designing the user interface for a simple program and customize that user interface using the Properties palette.

Last week, "Part 1: The Roots of REALbasic" introduced this four-part series.

In the early days, before Wally became an Internationally renowned comedian, computer book writer, and generally cool guy, Wally Wang used to hang around The Byte Buyer dangling participles with Jack Dunning and go to the gym to pump iron with Dan Gookin.

Wally is responsible for Microsoft Office 2007 for Dummies, Breaking Into Acting for Dummies, Beginning Programming All-in-One Reference for Dummies, and Mac All-in-One Reference for Dummies from www.dummies.com, as well as, Steal This Computer Book 4.0, Visual Basic Express 2005: Now Playing, and My New Mac from www.nostarch.com. He is also the co-author of Strategic Entrepreneurism from www.selectbooks.com.

Every Saturday morning from 9:00 am - 10:00 am in San Diego, you can hear Wally with fellow co-hosts Dane Henderson and Candace Lee, on the radio show CyberSports Today (www.cybersportstoday.com), which covers the video gaming industry on ESPN Radio 800 AM. Wally covers the military history side of the video game industry.

When not performing stand-up comedy or writing computer books, he likes to paper trade stocks with the video game Stock Reflex (www.plimus.com/jsp/download_trial.jsp?contractId=1722712&referrer=wwang).

Wally can be reached at wally@computoredge.com.

Send mail to ceeditor@computoredge.com with questions about editorial content.
Send mail to cwebmaster@computoredge.com with questions or comments about this Web site.
Copyright © 1997-2009 The Byte Buyer, Inc.

ComputerEdge Magazine, P.O. Box 83086, San Diego, CA 92138. (858) 573-0315

COMPUTOREDGE[®] ONLINE

www.computoredge.com

04/10/2009