Chapter 1

Introduction to SketchUp Scripting

Chapter Topics

- SketchUp scripting in brief: explanation and motivation
- SketchUp and the Ruby programming language
- Organization of this book

There are many three-dimensional modeling tools available, and the debate still rages over which provides the best features. But when it comes to ease of use and sheer *fun*, Google SketchUp® wins hands-down. Its user interface is so intuitive that most users can learn the basics in a matter of minutes. Its learning curve is so gentle that novices become experts in a matter of hours.

Most modeling tools are aimed at specific audiences. AutoCAD® attracts engineering types while animators and graphic artists prefer Maya®, Blender®, or 3ds Max®. But SketchUp has a broader point of view. This is reflected by its slogan and *raison d'être*: "3D for Everyone."

By all accounts, SketchUp lives up to its goal. Not only have architects and engineers embraced the tool, but also woodworkers, graphic artists, animators, mathematicians, and other creative types in a variety of fields. And if SketchUp can't meet your particular need, you can augment its functionality with *plugins*—a topic this book explores in great depth.

Despite SketchUp's egalitarianism, this book is *not* for everyone. We're going to put aside the friendly point-and-click interface and focus on interfacing SketchUp with text-based commands. As we progress, we'll store these commands in files called *scripts*. With the right scripts, you can accomplish everything you normally do in SketchUp and a great deal more.

1.1 Ten Reasons to Learn SketchUp Scripting

SketchUp's user interface is one of its primary strengths, so it may seem odd, even anachronistic, to have a book devoted to SketchUp's text-based commands. But there are many reasons why learning SketchUp scripting is a good idea. In no particular order, the following list presents ten of scripting's main advantages:

1. **Complex design management** - SketchUp's components and groups make it possible to create hierarchies of design elements. A large-scale design, such as a shopping mall, may comprise thousands of components, sub-components, and even sub-sub-sub-components. It's far easier to manage these hierarchies using scripts than the design window. Scripts also make it easier for other developers to review and analyze your design.

2. **Plugins** - A SketchUp plugin adds functionality to the overall application, such as new menu items, new tools in the toolbar, new dialog boxes, and many other possibilities. As we'll see, a plugin is just a SketchUp script in the right directory, and Chapter 10 explains precisely how these plugins are coded and deployed.

3. **Access design objects without the mouse pointer** - In SketchUp, your ability to find and select objects is only as good as your manual dexterity. This isn't a problem when your design is made up of simple shapes, but it becomes unwieldy for large, complex models like polygon meshes. With a script, it's easy to iterate through every element of the design and operate on only the ones you're interested in.

4. **Draw anything, anywhere** - Normally, you can draw objects only within your field of view, and you have to rely on construction points, color-coded axes, and the value-control box. For a new design, you can only create shapes in the x-y, x-z, or y-z planes. With SketchUp scripts, none of these restrictions apply. You can draw whatever you like, wherever you like.

5. **Animation** - SketchUp provides a number of ways to create designs that move. You can change the design viewpoint or create slideshows with multiple pages. With skeletal animation, you can animate a complex hierarchy of objects using straightforward rotations. To see what I mean, skip ahead to Chapter 12 and take a gander at the dancing robot.

6. **Free, friendly support** - Google has an online group specifically for SketchUp support and announcements: **http://groups.google.com/group/google-sketchup-developers**. All questions are welcome, and most receive responses within 24 hours. Further, the forums at **http://www.sketchucation.com** provide a wealth of support and information.

7. **Automation** - One of the primary advantages of storing SketchUp commands in a script is that you only need to type the commands *once*. After that, you can execute the script repeatedly. To make changes, simply edit the script in a text editor. You can also cut and paste commands between scripts.

8. **Web access** - As discussed in Chapter 13, SketchUp makes it possible to create WebDialogs that access pages on the World Wide Web. These dialogs can also serve as a bridge between JavaScript® code and your SketchUp design.

9. **Abundance of available scripts** - The SketchUp developer community is active and flourishing, and if your project calls for new functionality, it's likely that someone has already coded a script to suit the purpose and has made it available for download or purchase.

10. **Fun!** - Of all the software development tasks I've worked on, none provide the same sense of fulfillment as watching a new three-dimensional model come to life. And the more intricate the design, the greater the enjoyment.

1.2 Obtaining and Installing SketchUp

In case you haven't already installed SketchUp, let me explain the process. First, visit the web site **http://sketchup.google.com/download**. On the right, you'll see links for downloading regular SketchUp (free) and SketchUp Pro ($495 at the time of this writing). SketchUp Pro provides professional layouts and styles, advanced file operations, and full technical support. However, the free version still provides a ton of features. SketchUp runs natively only on the Microsoft Windows® and Mac® operating systems, but Linux users can access SketchUp through the WINE emulator. Click one of the links, make your way through the user agreement, and you'll have the option of downloading a file (*.exe on Windows, *.dmg on Mac) to your system.

Note: The directions in this section, like those throughout this book, are directed toward SketchUp 7.1 and the SketchUp 7.1 application programming interface (API).

Installation on the Windows Operating System

On Microsoft Windows, installing SketchUp is simple. Double-click the executable and click Next when the installer dialog appears. Accept the license agreement, click Next twice, and click Install. After installation, click Finish and SketchUp will be ready to run. Figure 1.1 shows what the application looks like in Microsoft Windows if you choose the Engineering template.

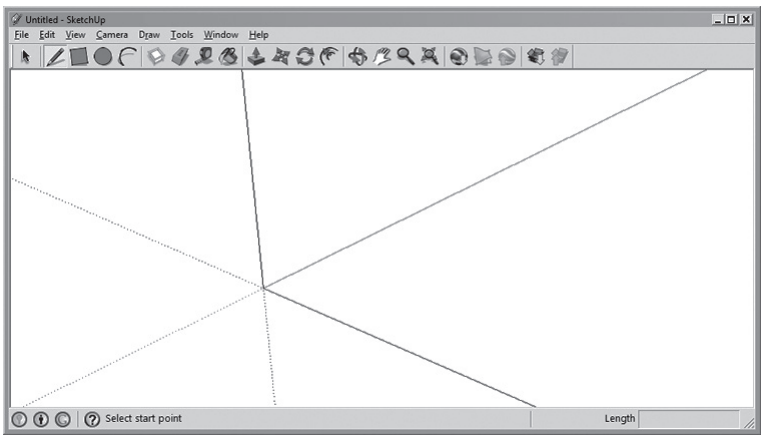

Figure 1.1: The SketchUp User Interface in the Windows Operating System

Mac OS X Installation

In Mac OS X, open the *.dmg file with the DiskImageMounter and open the resulting *.mpkg file with the Installer. Click Continue in the installer dialog, read the license agreement and click Continue and Agree. Next, select a volume where you'd like to install SketchUp and click Continue. Click Install to perform a standard installation, and after the installation, click Close. Figure 1.2 shows what the SketchUp user interface looks like.

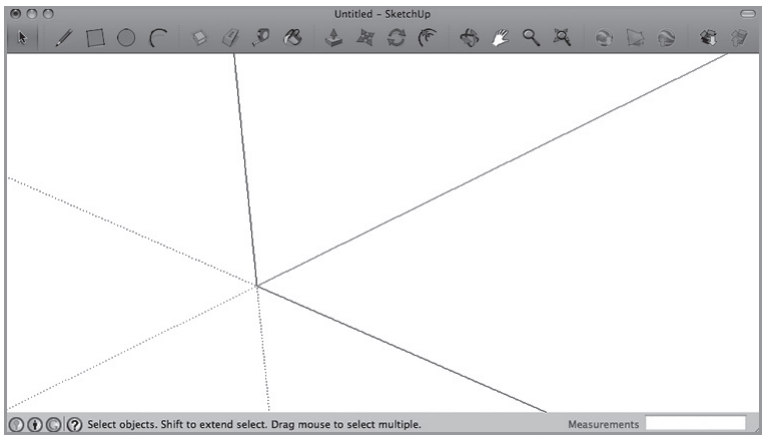

Figure 1.2: The SketchUp User Interface in Mac OS X

1.3 A Brief Example

I know, I know. We're still in the first chapter and this book is already getting into technical details. I apologize, but before you continue further, I want you to have an idea of what a SketchUp command looks like. If you're feeling adventurous and would like to execute the command, I recommend that you start SketchUp, open the Window menu, and select the Ruby Console entry. This is where all of our SketchUp commands will be entered.

Let's say you want to draw a line from the origin, [0, 0, 0], to the point [5, 5, 5]. Normally, this requires three mouse clicks: one click on the Line Tool in the SketchUp toolbar, one click on the origin, and one click when the mouse pointer reaches [5, 5, 5]. But you can also draw a line with the following command:

```
Sketchup.active_model.entities.add_line [0,0,0], [5,5,5]
```

To execute the command, enter this text in the Ruby Console window and press Enter. Figure 1.3a shows what this looks like on the Windows operating system and Figure 1.3b shows what the window looks like in Mac OS X.

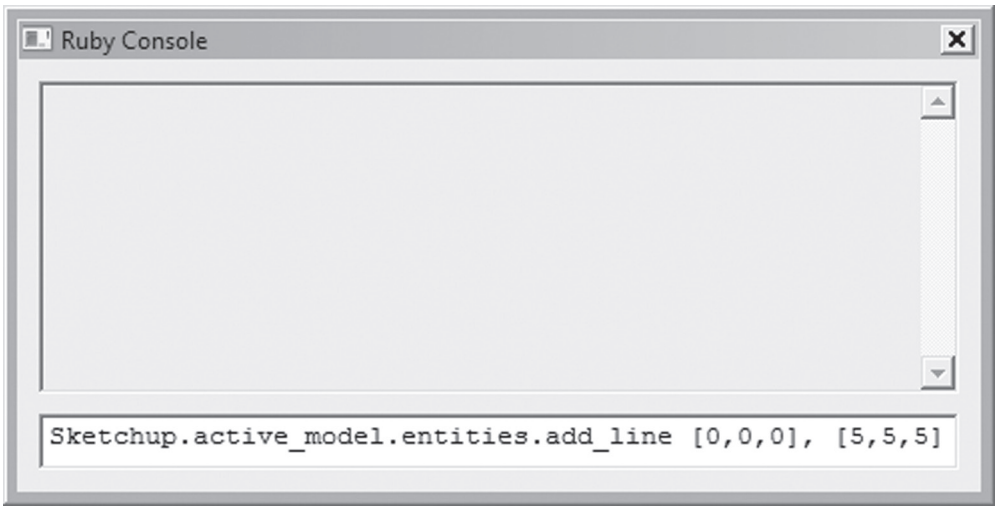

Figure 1.3a: The SketchUp Ruby Console Window in Windows

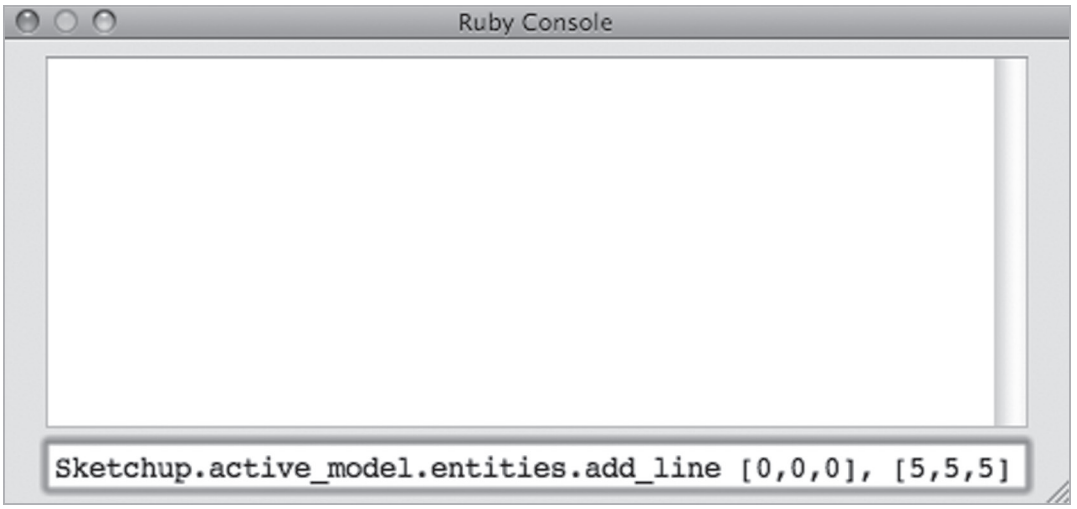

Figure 1.3b: The SketchUp Ruby Console Window in Mac OS X

When the command is executed, SketchUp draws a line between [0, 0, 0] and [5, 5, 5], just as though you'd used the Line tool. It takes more time to type the command than it does to point and click, but if you store the command in a file, you can execute it (and other commands) *automatically*. This isn't a great benefit when you're drawing only line segments, but when your design calls for parabolas, helices, and non-uniform rational B-splines (NURBS), scripts come in very handy.

Automation can be complicated, though, and many users will always prefer points and clicks. There's nothing wrong with this—I can't stand automatic car washes or automatic check-out lanes. If the idea of interfacing SketchUp with commands makes you nervous, that's fine. If you feel that way, I strongly recommend *The Google SketchUp Cookbook* or *Google SketchUp for Dummies*. Fine books both.

However, if the previous command sparks your curiosity and whets your appetite to learn more, this is the book for you. Not only will you see how to create every conceivable type of SketchUp design element, you'll also be able to add custom features to SketchUp, including new tools, pages, menu items, and plugins.

1.4 SketchUp Scripting and Ruby

This book contains a great deal of technical terminology, but the underlying language is English. Similarly, SketchUp commands are composed of SketchUp-specific terms, but the underlying language is *Ruby*. Ruby is a programming language whose primary use has been in the development of web applications (Ruby on Rails). Though relatively new, Ruby has acquired a passionate following, and a casual web search will provide many resources for instruction and support.

There are many advantages of basing SketchUp commands on the Ruby language. Three of the most important are as follows:

1. **Wide breadth of capabilities** - SketchUp scripts would be very limited if you could only draw lines, surfaces, and similar geometrical objects. But with Ruby's high-level features—control structures, iteration loops, conditional statements, and object-orientedness—your SketchUp scripts can perform all the processing tasks expected of modern software, from high-level network access to low-level graphics optimization.

2. **Plenty of available code** - Let's say you need to specify a position using spherical coordinates instead of traditional rectangular coordinates. This requires computing sines and cosines, which can be painful to code from scratch. But thanks to Ruby, you just have to call `Math.sin` and `Math.cos`. Need to read a file, parse a string, or telnet to a remote system? With the Ruby Standard Library, it's no sweat. If you visit the free documentation site at **http://www.ruby-doc.org**, you can explore the entirety of Ruby's capabilities online.
3. **Object-oriented** - The word "object" is going to get a lot of mileage in this book. On one hand, it may refer to a *thing* that you create as part of a SketchUp design, such as a line or circle or cube. On the other hand, "object" may also refer to a specific type of data structure. If this makes you nervous, don't worry; all you need to remember for now is that object-orientedness is an important priority in modern software development—not only does object-oriented programming lead to maintainable, well-structured code, using it makes you a better, wiser person.

Of course, if you've already programmed in Ruby, the greatest advantage of SketchUp's Ruby interface is that you're already familiar with the language. In that case, you can leap past a great deal of the introductory material in this book. I'd recommend going straight to Chapter 3 as soon as you get the opportunity.

But because of its recent origin, Ruby experience is rare even among professional programmers. So don't worry if you've never written Ruby scripts before reading this book. I'm going to assume you've never heard of Ruby. In fact, I'm going to assume *you've never coded before at all*.

This presents a challenge: I want this book to be as friendly as possible to newcomers without being too boring for experts. The following section explains how this book is organized to suit the needs of both audiences.

1.5 Organization of This Book

A solid introduction to Ruby requires at least three chapters, and at first, I'd planned to devote Chapters 2–4 to this purpose. But SketchUp scripting is so much fun that I decided to intersperse the Ruby chapters between those that concentrate on SketchUp. Therefore, Chapters 2, 5, and 8 focus solely on the study of Ruby—if you're already an expert coder, feel free to skip them.

As for the SketchUp-specific chapters, I've done my best to organize the presentation in an intuitive manner, from the simple and general to the complex and specific. The chapters in this book are divided into three parts:

- Part 1, consisting of Chapters 2–6, provides a basic introduction to Ruby and SketchUp. Once you're finished with these chapters, you'll be able to create any type of shape in the SketchUp design window. You'll also be able to configure the position, scaling, and appearance of each shape you create.
- Part 2, consisting of Chapters 7–9, explains how to create and manage SketchUp design hierarchies and attach information through attributes, options, and observers. These capabilities aren't necessary for simple designs, but become vitally important as your models grow in complexity.
- Part 3, consisting of Chapters 10–13, discusses advanced SketchUp operations that you can't perform using the traditional design window. These chapters discuss plugins and all the different ways you can augment the SketchUp user interface. The last two topics are the most interesting and tie together all of the book's subject material: animation and WebDialogs.

Throughout this book, example SketchUp scripts will be provided to clarify concepts and show how commands are used in code. To download the sample scripts, visit **http://www.automaticsketchup.com**. This site also provides access to errata and updates to the book's content.

1.6 Conclusion

At first glance, SketchUp may look more like a child's drawing application than a professional modeling tool. But as you become more familiar with SketchUp, you can appreciate its amazing breadth of functions and capabilities. And this is just the tip of the iceberg. If you really want to make the most of SketchUp, you need to know how to write scripts.

This chapter has discussed the merits of SketchUp scripting at length, but I'd like to focus once more on the aspect of *fun*. I don't usually enjoy programming, but SketchUp scripting isn't like regular coding. When executed, a SketchUp script can instantly create objects, models, materials, and even animation. For a clumsy nonartist such as myself who can barely draw stick figures, this is pure joy.

SketchUp scripts are based on the Ruby programming language, and in the chapters that follow, I'll do my best to explain Ruby in sufficient depth as to make SketchUp's capabilities comprehensible. If you'd like more information on Ruby, I recommend that you visit the **http://www.ruby-doc.org** site and the free online book *Programming Ruby: The Pragmatic Programmer's Guide* located at **http://www.rubycentral.com/book/**. For further support with SketchUp scripting, you can't do much better than **http://www.sketchucation.com** and **http://groups.google.com/group/google-sketchup-developers**.

As you progress through this book, I strongly recommend that you experiment with new commands and scripts. Don't just look over the example listings, but actually get your hands dirty and try out new modeling ideas. This will not only give you a better understanding of the material, but also a greater enjoyment of SketchUp scripting in general.