Chapter 1

Why You Want to Write Games in Flash

In This Chapter

- Seeing how to use Flash to write games
- How programming differs from animation
- Exploring basic game design concepts

Computer programming can be a whole lot of fun. That's why I got into it way back when, and it's why I still do it. Truth be told, the main reason I learned how to program was to write games. I couldn't buy much software for my first computer (a TRS-80 Model 1, still in the garage . . . sigh). I wanted to play games, so I had to create them myself. Admittedly, I was pretty bad at it, and I failed a lot, but I kept trying. As I grew up, my programming skills were marketable in the "serious" world, but I never lost my fascination with computer games.

Here are some very good reasons to write games:

- Computer games made more income in 2003 than the movie industry.
- Game programming is technically challenging.
- Making a game is *fun!*

Most other game development books can be divided into two camps:

- Some talk about the game design process, storyboarding, coming up with game ideas, and the visual side of gaming. That's pretty good stuff to know, but it doesn't help you actually make a game.
- Other books assume that you're already good at C++ and advanced math. That's pretty good stuff, too, but you don't need to start there.

I believe that newcomers to programming can master the essential ideas of programming at the same time they're learning to build games. I also feel that those with some programming experience will truly enjoy the uniquely creative aspects of game development. You don't have to know anything about programming or Flash to use this book. (However, if you know these things, you'll still probably see something new.)

In this chapter, I give you an overview of the basics of game designing and planning, writing, and programming in Flash (with ActionScript). Most of all, you're going to have a lot of fun.

Designing and Writing Games

If you've asked around about how to get started in game programming, people have probably told you to learn C++ and take lots of math classes. That's not bad advice, but I have an easier way. The truth is that making games isn't really about any particular computer language. After you learn how to write games, you can transfer those concepts to any environment you wish. There are surprisingly few main concepts behind game development. If you truly understand these ideas, you can translate them to any programming language you want.

In this book, I show you how to program games in Flash. I like Flash because it simplifies the visual side of programming, works on almost every computer made, and has a powerful and reasonably easy programming language. I talk about this more in the upcoming sections, "Game Programming in Flash" and "Game Programming 101."

Too, game programming is different than other kinds of software development. For one thing, games need to be fun. And games are all about communicating with the player as well as providing some sort of immersive world in which the player participates. As a game programmer, you get to be creative and think outside the box.

Making artificial worlds

Typical business programming relies heavily on certain conventions and metaphors. If you're writing a database application, it's *de rigueur* to make your program much like all the other programs users have seen. In game programming, though, you're often trying to "hide" the computer from the player. For example, if you're making a spaceship game, you want the controls to look and feel like spaceship controls. Imagination is a really important part of playing and writing games.

The importance of interactivity

Games need to react to the player. The player should manipulate a virtual presence, and the game should react accordingly. Some games are turn based, and some are in real time, but all require more immediate feedback than traditional types of programs.

Games are about objects

Many games involve objects bonking into each other, shooting each other (with other objects), avoiding each other, and simply milling around. While you're writing a game, you usually think about objects, their characteristics, and how they interact with the player as well as with each other.

Players compete with the programmer

When you play a really great game, you're not really playing against the computer. Rather, you're really engaging in a stylized conversation with the programmers. As a game developer, you get the chance to set up worlds. The players interact with a stored version of your thoughts and imagination.

Game Programming in Flash

Macromedia Flash is a very good environment for learning basic game programming ideas. Here are a number of reasons for starting with Flash:

- **Flash offers robust multimedia support.** Flash, which was designed to support animation on the Web, supports various kinds of images easily. (Think JPEG images and custom drawings.) See how to use the drawing features of Flash in Chapter 9. Flash also has great support for various kinds of audio files, such as MP3 and WAV formats. You incorporate audio into your games in Chapter 8.
- **ActionScript is related to the influential C language.** The ActionScript programming language built into Flash is closely related to JavaScript and ECMAScript, which are two extremely common programming languages. All these languages are based on the C programming language, so the coding conventions you'll master are much like those in other languages.

- **Flash is designed for the Web.** By working in Flash, you have a ready distribution network. Because Flash was designed for the Web, all your games can be easily published on the Web, and anybody with a Web browser and a Flash plug-in can enjoy your games. And you won't have to worry about what operating system your users use. (All the programs in this book have been tested in Windows XP and Fedora Core Linux, but they should work in any OS with a Flash plug-in.)

Comparing ActionScript with Animation

Maybe you've used Flash to build Web animations without ever going into its programming features. Many books on Flash (as opposed to ActionScript) focus on the powerful animation features of Flash. These books often mention ActionScript but don't dwell on it heavily. Animation is primarily about creating moving images; user interaction in animations is minimal. When creating an animation, you generally create some sort of visual symbol onscreen and then use a tool called a *motion tween* to indicate where this object should be at a specified point in time. You can also use a tool called a *shape tween* to change the shape of an object over time. You can do this with many objects at the same time to make a complex animation. In order to track all these objects, Flash animators often arrange them into separate *layers.* Thus, a typical 30-second Flash animation might have hundreds of frames of animation in over a dozen layers.

Animation is cool because it allows you to build movies. However, to create games, you must discover how to program.

If you treat Flash as a programming environment (as I do in this book), you see things quite differently. You still use Flash to create objects, but instead of relying on the Flash environment to control what those objects do (via animation), you control the objects directly by writing programming code. The ActionScript programming language built into Flash lets you do anything that can be done with animation — and many things that cannot be done by using animation techniques alone.

In a nutshell, programming is what makes games interactive. You can

- Control what's onscreen, what size it is, where it is, and how it's rotated
- Detect whether two things touch each other
- Accept input from the user

How You Make a Game

The goal of game development can be summarized in one sentence:

Games are stories that use the player as a primary character.

Like any interesting story, a game needs these plot elements:

- A character (at least one)
- A conflict
- A goal

Game play must be compelling, but game elements don't need to be complex. Simple games like *Tetris* and *Pac-Man* have had phenomenal success.

Making a playable game

A good game has a good story, and it also has some form of user interaction. In Flash, the player uses the mouse and keyboard as primary input devices. Although these devices might seem limiting (compared with a more sophisticated joystick or driving console), you can do many things with these basic forms of input.

Although Flash doesn't directly support joystick input, users can easily use modern joysticks with the games you can write in Flash. Most joysticks now come with programs that allow the user to map keystrokes to keyboard commands. In effect, by allowing keyboard input, you also allow rudimentary joystick input.

A game should also look good and sound good, but these things don't matter if the game isn't fun.

Some of the best games ever have incredibly limited graphics and sound. If you've never played *NetHack* (as shown in Figure 1-1), download a copy (free for just about any computer ever made) and play it. At first, you might be thrown by the complete lack of graphics and sounds. *NetHack* uses plain text without any graphics or sound effects, but the game is amazingly absorbing. I bet that you get caught up in the incredible game play and find yourself actually scared of the capital D coming at you.

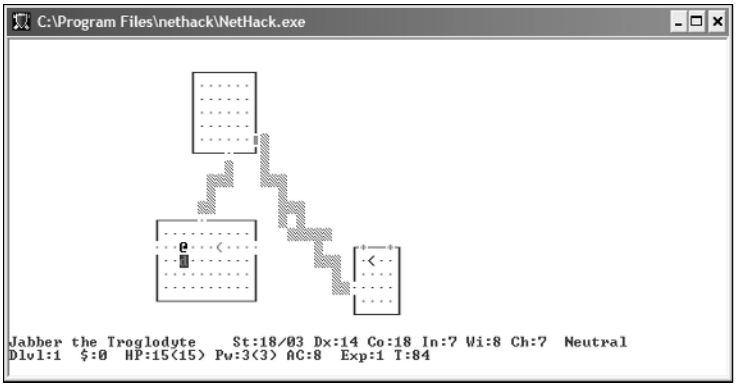

Figure 1-1: A captivating game isn't always about flashy graphics and sound.

Most of all, games should be fun. I can't really tell you how to make a game fun. You need to test a lot for a game that's fun to play.

Starting with a plan

Before you worry at all about the details of your game, come up with a theme.

Think about what you want your game to be about. Outline and define the following components:

- **The main character:** Don't forget the kinds of obstacles this character will encounter. You can read how to build a main character throughout the book, but the topic is covered most deeply in Chapter 9.
- **The overall look of the game:** Consider the setting. What colors will you use? What overall look and feel are you looking for? (Retro? Cartoon? Gothic? Maybe a Gothic retro cartoon?) Chapter 7 describes how to set up the visual feel of a game.
- **The main screens:** Most games have
  - A main play screen (or two)
  - An instruction screen
  - Some sort of introduction
  - A Game Over screen, or maybe two: one for when the player wins and one for when the player loses

  Chapter 7 describes how to build multiple screens for your games.

  Draw these visual elements on paper.
- **The objects on each screen:** You have to build everything in your game. The visual design part is important but relatively easy.

- **The role/behavior of each object:** Decide these details upfront for each object:
  - How the object moves
  - Whether it's controlled by the user or the computer
  - Whether it does something when it interacts with other objects
  - Whether it makes sounds
  - What happens when it leaves the screen

After you finish defining these objects, convert your sketches into reality. This sounds like a pretty easy step, but it's the one that might cause you a lot of grief. You probably know exactly what you want all the screen objects to do, but a computer is incredibly stupid.

You have to convert your clever ideas to statements so clear that even an idiot computer can understand them.

Learning to code

Mountain climbers train before they scale the big mountains:

- **Learn:** They learn the tools of the trade, practicing on small hills and isolated, safe areas before testing their skills on actual mountains.
- **Pace:** When they're ready to climb Mt. Everest, climbers don't go for the top in one day. They build a solid base camp at the foot of the mountain. Then they create another camp higher up, and another even higher.
- **Progress:** At each camp, the ultimate goal is still the summit, but the intermediate goal of the next camp is the task at hand.

  A mountain climber concentrates on the next step.

The same advice is really good for all programmers, beginning or advanced:

- **Master the tools of the trade.** There's no getting past the fundamentals (which this book shows).

  You need to know both
  - The basic ideas of programming
  - The principles of game development
- **Know the goal.** That's why you start with a written description of your program.

- **Use small steps.** Concentrate on mastering one task at a time.

  In this book, chapters show you specific skills and apply those skills in simple games, so you can

  - Learn the skills you need for complex games.
  - Practice these skills in isolated programs.

- **Enjoy the view.** Game programming is supposed to be fun.

  Celebrate your progress! When you succeed at a viable chunk of code, do a little Hampsterdance. (If you don't know what I'm talking about, visit `www.hampsterdance.com`.)

- **Pace yourself.** Your first program won't be the next version of *Quake,* but there's plenty of fun in writing games that are a little less ambitious. Eventually, you'll build your skills so you can write something way better than *Quake.*

Game Programming 101

Game programming is a process. All the programs in this book use the Flash environment, but the details of Flash programming aren't the most important factor. When you want to make a game, you need to choose an environment that will work as well as decide a strategy for creating the game.

Selecting a language

If you're reading this book, you've chosen Flash as your environment. Excellent choice!

- **Flash is an ideal environment for beginning game creation.**

  Flash makes a lot of the implementation easier, so you can concentrate on the content of your games instead of all the details of memory management, image drawing, and reading the input devices. (Fancier environments make you put a lot of work into such details instead of mastering the craft of game development.)

- **Most commercial games are written in 2-D, using C++ and graphics engines like DirectX or OpenGL.**

  Those are really great environments, but they aren't necessarily what you need while you're learning the process of game development.

If you want to be a racing champion, you don't just show up in Indianapolis with a helmet. Starting your driving career in a high-performance machine is foolhardy and dangerous. You begin your career racing karts and then advance through more challenging vehicles. That's why you should start programming with Flash and ActionScript:

- C++ is like a Formula 1 car — fast and difficult to handle.
- Flash and ActionScript are like a go-kart (albeit a souped-up, Internet-enabled go-kart that outperforms any computing environment NASA had during the moon program).

Planning tasks

Game programmers prepare by planning several parts of the game:

- **Encapsulating objects onscreen:** All the things that move around on the computer screen are called *sprites* by game programmers.

  Chapter 6 shows you what need to know:

  - What a sprite is
  - How to create a sprite with Flash tools

  

  Flash has a great object called a *movie clip* that can easily be used as a basis for sprites. Chapter 6 shows how to use it.

- **Accepting input from the user:** There are all kinds of user input devices, but Flash games concentrate on the mouse and the keyboard.

  Chapters 2 and 6 show how to get information from the mouse, and Chapter 8 describes reading the keyboard in Flash.

- **Moving things realistically:** Game programmers must

  - Understand the physics properties of position, velocity, and acceleration, and attach these characteristics to a sprite so that it moves in a realistic fashion.
  - Simulate such useful physical properties as gravity and friction.

  

  Chapter 6 shows how to manage basic motion in Flash. Read about more sophisticated motion in Chapter 9 and see how to create realistic vehicles in Chapter 12.

- **Dressing up the user's experience:** Graphics, sound, and animation matter. Chapter 9 shows you how to use graphics, and Chapter 8 shows you to use sound effects.

- **Keeping the action fun:** Every game must be able to adapt to the user's ability level.

  Find places to adapt the computer's ability so the computer always gives users an interesting challenge. As I describe each game in the book, I give hints how you can make the game easier or more difficult.