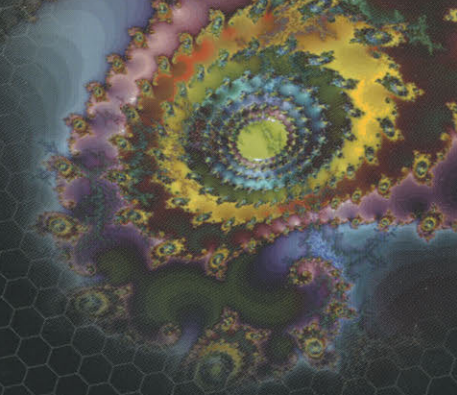
THE EXPERT'S VOICE® IN JAVA



Advanced Java Game Programming

David Wallace Croft

APress Media, LLC

Advanced Java Game Programming

DAVID WALLACE CROFT

APress Media, LLC

Advanced Java Game Programming

Copyright © 2004 by David Wallace Croft

Originally published by Apress in 2004

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN 978-1-59059-123-9 ISBN 978-1-4302-0713-9 (eBook)
DOI 10.1007/978-1-4302-0713-9

Trademarked names may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, we use the names only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Lead Editor: Steve Anglin

Technical Reviewer: Jack Park

Editorial Board: Steve Anglin, Dan Appleman, Gary Cornell, James Cox, Tony Davis, John Franklin, Chris Mills, Steve Rycroft, Dominic Shakeshaft, Julian Skinner, Jim Sumser, Karen Watterson, Gavin Wray, John Zukowski

Project Manager: Sofia Marchant

Copy Manager: Nicole LeClerc

Production Manager: Kari Brooks

Production Editor: Ellie Fountain

Compositor: Gina Rexrode, Point n' Click Publishing, LLC

Proofreader: Susannah Pfalzer

Indexer: Brenda Miller

Artist: Kinetic Publishing Services, LLC

Cover Designer: Kurt Krames

Manufacturing Manager: Tom Debolski

The information in this book is distributed on an "as is" basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

To my parents, Joseph Wallace Croft, Jr. and Libby Ann Stischer Croft.

Contents at a Glance

Contents

About the Author

David Wallace Croft is a Java software architect with a professional background in Java game development. He formerly served as the president of the Silicon Valley Java Users Group and is the founder of the Game Developers Java Users Group (`http://www.GameJUG.org/`). He earned his B.Sc. from the United States Air Force Academy in 1990 and his M.Sc. from the California Institute of Technology in 1995. After a brief career in neural network chip design, he joined an online Internet multiplayer game start-up in 1996 and has been programming in Java exclusively ever since. While writing the book, he taught Java 2D game programming within the Institute of Interactive Arts & Engineering program at the University of Texas at Dallas (`http://iiae.utdallas.edu/`). In 2004, he transitioned from faculty to student and is now pursuing a doctorate in cognition and neuroscience in the School of Behavioral and Brain Sciences at the same university. His contact information is available at `http://www.croftsoft.com/people/david/`.

About the Technical Reviewer

Jack Park gives pretty good google. To do that, he remains very active developing software in the open source arena. His projects have included NexistWiki, an experimental platform that combines topic maps, issue-based information systems, and storytelling, together with wiki behaviors and weblogs. He produced, with technical editorial help from Sam Hunting, and with authors drawn from all over the planet, the book *XML Topic Maps: Creating and Using Topic Maps for the Web* (Addison-Wesley, 2002). In a former life, he built windmills and solar heaters, and created the book *The Wind Power Book* (Cheshire-Van Nostrand, 1981). He is presently employed as a research scientist with SRI International.

Acknowledgments

I would like to thank the following individuals for their contributions to this book.

My parents Joseph Wallace Croft, Jr. and Libby Ann Stischer Croft for their encouragement and financial assistance while I dedicated myself full-time to writing and coding. If it were not for their support, this book would not exist.

My wife Shannon Kristine Croft for enabling and motivating me in multiple ways including the preparation of my coffee thermos for my nightly writing sessions. She also contributed some of the graphics for the demonstration games.

My brother Steven Morris Croft for technical review of the earliest drafts and for contributing some of the game graphics.

My children Ada Beth Croft, Thomas Edward Croft, and Benjamin Wallace Croft. Young children make the best game testers.

My friend Jack Park (`http://www.thinkalong.com`), editor of *XML Topic Maps: Creating and Using Topic Maps for the Web* (Addison-Wesley, 2002), for his much appreciated editorial comments as technical editor of this book.

Dr. Sridhar V. Iyer, founder of the virtual reality collaborative learning startup Whoola (`http://www.whoola.com`), for providing free office space to write the book, for encouraging me to create educational Java games, for supporting the Open Source development of reusable code libraries, and for suggesting that I teach a course on the same subject while I write the book.

Dr. Thomas E. Linehan, Director of the Institute of Interactive Arts & Engineering at the University of Texas at Dallas (`http://iiae.utdallas.edu/`), for sponsoring my teaching of an undergraduate course on the subject matter of this book and for his encouragement.

My friend John Hattan of the game development company The Code Zone (`http://www.thecodezone.com/`) for recommending free and inexpensive game development tools and multimedia libraries.

My friend Sam L. Robertson IV (`http://www.slriv.com`) for volunteering to host a server to demonstrate the server-side game code.

My friend James Gholston, president and general partner of the game development company Dimensionality (`http://www.dimensionality.com`), for releasing some of his professional graphics to the Public Domain so that they could be included in the example games.

Jenn Dolari (`http://www.dolari.org/`) for creating alternative graphics for one of the example games.

Ari Feldman (`http://www.arifeldman.com/`), author of *Designing Arcade Computer Game Graphics* (Wordware Publishing, 2000), for allowing game programmers to use his sprite graphics library for free.

John Zukowksi (`http://zukowski.net/`) for prompting me to write this book and for serving as its initial editor. I have enjoyed his books and online articles on the subject of Java since the earliest days, so I was quite impressed and inspired by his initial suggestion that I might write a book on this topic.

Craig Berry, the second editor, for his recommendations for improvement.

Steve Anglin, the third editor, for the kind words and giving the green light to go to press.

Susannah Pfalzer, the proofreader, for replacing "which" with "that."

Sofia Marchant, the project manager, for introducing me to the process of writing a book.

Introduction

An ounce of wit that is bought, is worth a pound that is taught. —*Benjamin Franklin*

This book is for experienced Java programmers who want to create sophisticated 2D computer games for the Web and the desktop using the latest high performance techniques. Each chapter builds upon the previous by gradually introducing a reusable animation library. The source code for each new library class is described in detail where necessary to document and explain the topics. Example games that demonstrate the classes in actual use are presented. Royalty-free licensing encourages readers to adopt and adapt the library code and the example games for their own use.

Purpose

An alternative title for this book might have been *Modern Java Game Programming*. Many of the techniques and topics covered in previously published Java game programming books have been obsoleted by the advance of the Java programming language and its libraries. These include areas such as thread management, event management, graphical user interfaces, network communications, persistence, and deployment. Where appropriate throughout the book, I have noted these changes and briefly contrast the techniques.

While it has always been possible to create high speed animation in Java by minimizing the average number of pixels painted in each frame, new classes introduced recently in version 1.4 of the Java programming language now offer direct access to video hardware in a portable manner. As documented within this book, it is now possible to achieve fast frame rates even when the number of pixels updated in each frame is very large. An example program is documented herein that has been demonstrated to successfully synchronize full-screen animation to a monitor refresh rate of 75 Hz at a high resolution in true color.

Even though the primary focus of this book is on games, the reader should keep in mind that techniques covered here can be used for other applications such as advertising and simulation. Animated banner ad applets embedded in web pages leap to mind. The fact that banner ads can be written using a powerful general-purpose programming language such as Java substantially increases their potential for additional functionality beyond animation, including live data connects. In the realm of simulation, students and scientists benefit from being able to visualize. When these models are presented using Java, users can interact with the simulations, change the parameters, and see new results.

Prerequisites

I have written *Advanced Java Game Programming* for experienced Java developers who are ready and eager to convert their unique game design ideas into deployed realities. For these readers, the novelty of learning the Java programming language and implementing basic infrastructure code has long since been surpassed by a desire to refine novel and interesting game-specific logic.

At the same time, these advanced Java developers want to understand in detail how the optimization choices in a reusable library may impact the performance of their games. In describing the source code for the game library and the example games documented in this book, I assume the reader has already been introduced to basic topics such as the Java programming language, Object Oriented Programming (OOP), Graphical User Interface (GUI) component libraries, and, to some extent, design patterns.

Scope

I introduce briefly without depth those technologies that are required knowledge but are not specific to Java game programming. These include popular Open Source development tools and common standards and application programming interfaces (APIs) that many experienced Java game programmers will have already used. In these cases, I direct those readers unfamiliar with such technologies to other resources for further reading.

Most of the reusable libraries that I introduce take advantage of the latest APIs available in the core Java platform, currently the Java 2 Platform Standard Edition (J2SE) version 1.4. I do not cover outdated APIs such as the Abstract Window Toolkit (AWT). I instead delve into the details of applying modern APIs such as Swing for animation in Java game programming.

With the exception of the Java 2D and the Image I/O APIs, most of the Java Media APIs are not covered within this volume, as I have reservations concerning their use within game development. Using either the Java 3D, Java Media Framework (JMF), or Java Speech API in games forces the players to download and install an optional package and its native code implementation. I believe that this extra step will discourage many potential players, especially those that encounter installation problems or download delays. The Java Sound API, although installed as part of the J2SE core, requires a soundbank file that is included by default in the Java Runtime Environment (JRE) distributions for some operating systems but not for Windows. The Java Shared Data Toolkit (JSDT) API, although of possible use in network games, appears to be discontinued. I could find no obvious use for the Java Advanced Imaging (JAI) API within game programming.

Use of the Java Native Interface (JNI) API to mix Java with platform-specific code is neither covered nor encouraged. All of the code documented in this book is written in pure, portable Java and should run on any platform with a Java virtual machine. As a developer, I find that I can achieve high performance without using custom native libraries. As a player, I prefer games written in pure Java because I do not have to fret about security risks and I know that the games will run on my favorite platform, whatever it may be.

This book does not cover Java game development using the Java 2 Micro Edition (J2ME) platform. I expect that rapid advances in hardware capabilities as expressed by Moore's Law are going to make J2ME obsolete soon. The hand-held PDAs of today now use microprocessors that are as powerful as those in the best desktop computers of three years ago. It is now possible to install J2SE and run Swing applications on a PDA. I would not be surprised to see Java 2 Enterprise Edition (J2EE) running on wristwatches within a few years.

In later chapters on multiplayer networking, this book does require some knowledge of J2EE. This is limited, however, to development with the Servlet API subset. The use of more advanced J2EE APIs such as Enterprise Java Beans (EJBs) is not covered in any depth within this volume. All of the example network games will run within a simple servlet container, and installation of a full-fledged application server is not required.

Overview

In Chapter 1 "Development Setup," I introduce the reader to the code library that will be used throughout the book. The library layout is described and instructions for compiling the example games are given. Additional sources of code, graphics, and audio that may be used in game development are identified and popular tools for software development are introduced. An example game that demonstrates the basics of Java game programming is provided as a template.

In Chapter 2 "Deployment Frameworks," the focus is on deploying your games within a framework that can be installed on as many different types of platforms as possible. A widely used framework interface, the applet lifecycle, is discussed within the context of animation thread management techniques. After a review of different deployment options suitable for game distribution, a source code example of an abstraction layer that permits games to be deployed in different environments without modification is given.

In Chapter 3 "Swing Animation," a Swing-based animation library is described that will be used throughout the book. The primary consideration in this chapter is animation performance and flexibility. The reader is briefed on the optimization trade-offs as the source code of the core animation engine classes is reviewed.

In Chapter 4 "Animation Library," a collection of classes for common animation tasks is documented. These classes provide functionality, such as scene management and sprite physics. They also serve to demonstrate how game-specific code can be developed that will interoperate with the core animation engine classes.

In Chapter 5 "Advanced Graphics," I describe advanced graphics techniques such as hardware-accelerated graphics, multi-buffering, and full-screen exclusive mode. Reusable classes that facilitate the use of these techniques are provided and example games that use them are demonstrated. For each of these techniques, I discuss considerations regarding their use.

In Chapter 6 "Persistent Data," mechanisms for loading and saving game data within the various deployment frameworks are compared. A library of reusable data persistence classes suitable for most game programming needs is documented in the process. Alternatives for more advanced persistence requirements are considered.

In Chapter 7 "Game Architecture," the merits of an object-oriented software architecture suitable for Java game development are discussed. An example game that uses this architecture is offered as a template for the development of new games. Data-driven design is promoted.

In Chapter 8 "A* Algorithm," an implementation of one of the most popular and commonly used artificial intelligence (AI) algorithms within the game industry is provided. An example of using the A* algorithm for path-finding around obstacles is demonstrated.

In Chapter 9 "HTTP Tunneling," a networking library that can operate within the security restrictions common to most Java game deployment environments is introduced. An example is provided in which data is transferred between a client and a server.

In Chapter 10 "HTTP Polling," the networking library is extended to support online multiplayer games. Polling is used to synchronize the game state on the client with the server. The flexibility of the recommended software architecture is demonstrated in the conversion of a single player game to a multiplayer networked game.

In Chapter 11 "HTTP Pulling," event-driven messaging is recommended as an alternative to polling. Using the classes documented in the previous two chapters, I demonstrate how this can be accomplished within the security restrictions of an unsigned applet. Recommendations for further development using techniques beyond the scope of this volume are briefly considered.

Web Site

A book with the title *Advanced Java Game Programming* should remain advanced. For this reason I have set up a web site dedicated to the book. From there you will be able to subscribe to the electronic mailing list for book-related announcements including supplemental tutorials and errata. You will be able to test and play the example games described in the book and download the Open Source code library and Public Domain multimedia files that were used to create them. For teachers, I have released under the terms of the Creative Commons Attribution License the syllabus, slides, and assignments that I use in the game development course that I lecture. I also provide my contact information so that you may send your comments and suggestions for future editions.

Before proceeding to the first chapter, please visit `http://www.croftsoft.com/books/ajgp/` and subscribe to the *Advanced Java Game Programming* announcements mailing list.

CHAPTER 1

Development Setup

A good example is the best sermon. —Benjamin Franklin

In this chapter, you will learn how to configure your development environment to create your own games using the game library and examples described in this book.

Upgrading to 1.4

To compile and run the example games in this book, you need to download and install the Java 2 Software Development Kit (SDK), Standard Edition (J2SE), version 1.4 or later. "Java 2" refers to the Java 2 Platform. "Java 1.4" refers to the specific version of the Java 2 Platform. You can blame the marketing folks at Sun Microsystems for the confusion.

Version 1.4 is required because the game code depends upon new accelerated graphics classes in that version of the J2SE. These new classes greatly increase animation performance. At the time of this writing, J2SE v1.4 is available on multiple platforms including Linux, Mac OS X, Solaris, and Windows. If you do not have version 1.4 or later installed, you can download it from `http://java.sun.com/` for the most common platforms besides Apple. Apple developers can download the latest version of Java from `http://developer.apple.com/java/`.

The game code should run without modification equally well on all platforms that support J2SE v1.4. Having said that, I must now warn you that I have only tested the code on Linux and Windows. I will depend upon the feedback of readers with Mac OS X and Solaris systems to let me know how the code fares on those platforms.

Sticking to the Core

The J2SE defines a standardized application programming interface (API) library with thousands of classes grouped into scores of packages that are considered *core*. The core classes are guaranteed to be pre-installed on any J2SE-compatible system. This makes the life of a developer much easier because it means less code that needs to be downloaded and installed to the client platform. An example of a core package is `java.lang`, which contains the basic classes needed in most Java programs.