Beginning .NET Game Programming in VB.NET

DAVID WELLER, ALEXANDRE SANTOS LOBÃO,
AND ELLEN HATTON

Apress®

Beginning .NET Game Programming in VB.NET

ISBN (pbk): 1-59059-401-1

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Technical Reviewer: Andrew Jenks

Editorial Board: Steve Anglin, Dan Appleman, Ewan Buckingham, Gary Cornell, Tony Davis, Jason Gilmore, Chris Mills, Dominic Shakeshaft, Jim Sumser

Assistant Publisher: Grace Wong

Project Manager: Sofia Marchant

Copy Editor: Ami Knox

Production Manager: Kari Brooks

Proofreader: Linda Seifert

Compositor: Dina Quan

Indexer: Rebecca Plunkett

Cover Designer: Kurt Krames

Manufacturing Manager: Tom Debolski

Distributed to the book trade in the United States by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, New York 10013 and outside the United States by Springer-Verlag GmbH & Co. KG, Tiergartenstr. 17, 69112 Heidelberg, Germany.

In the United States: phone 1-800-SPRINGER, email orders@springer-ny.com, or visit http://www.springer-ny.com. Outside the United States: fax +49 6221 345229, email orders@springer.de, or visit http://www.springer.de.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, email info@apress.com, or visit http://www.apress.com.

The source code for this book is available to readers at http://www.apress.com in the Downloads section.

Para Ana: Mi esperanza, mi corazón,
mi tesoro, mi amiga, mi amor.

Contents at a Glance

Contents

Foreword

Back a few years ago I had an idea. What if I could make the power of the DirectX API available to the developers who were going to be using the new set of languages and common language runtime that Microsoft was developing? The idea was intriguing, and opening up a larger portion of the world to DirectX was a goal I was only happy to endorse. Besides, what developer doesn't want to write games?

It seems that at least once a week I am answering questions directly regarding the performance of managed code, and Managed DirectX in particular. One of the more common questions I hear is some paraphrase of "Is it as fast as unmanaged code?"

Obviously in a general sense it isn't. Regardless of the quality of the Managed DirectX API, the fact remains that it still has to run through the same DirectX API that the unmanaged code does. There is naturally going to be a slight overhead for this, but does it have a large negative impact on the majority of applications? Of course it doesn't. No one is suggesting that one of the top-of-the-line polygon pushing games coming out today (say, Half Life 2 or Doom 3) should be written in Managed DirectX, but that doesn't mean that there isn't a whole slew of games that could be. I'll get more to that in just a few moments.

The reality is that many of the developers out there today simply don't know how to write well-performing managed code. This isn't through any shortcoming of these developers, but rather the newness of the API, combined with not enough documentation on performance, and how to get the best out of the CLR. For the most part, we're all new developers in this area, and things will only get better as people come to understand the process.

It's not at all dissimilar to the change from assembler to C code for games. It all comes down to a simple question: Do the benefits outweigh the negatives? Are you willing to sacrifice a small bit of performance for the easier development of managed code? The quicker time to market? The greater security? The easier debugging? Are you even sure that you would see a difference in performance?

Like I mentioned earlier, there are certain games today that aren't good fits for having the main engine written in managed code, but there are plenty of titles that are. The top ten selling PC games just a few months ago included two versions of the Sims, Zoo Tycoon (+ expansion), Backyard Basketball 2004, and Uru: Ages Beyond Myst, any of which could have been written in managed code.

Anyone who has taken the time to write some code in one of the managed languages normally realizes the benefits the platform offers pretty quickly. Using

this book, you should be able to pick up the beginning concepts of game development pretty easily. It takes you through the simple sprite-based games, all the way through a basic 3-D game implementation.

It's an exciting time to be a developer.

Tom Miller
Lead Developer for the Managed DirectX Library,
Microsoft Corporation

About the Authors

Somewhere around 1974, **David Weller** discovered a coin-operated Pong game in a pizza parlor in Sacramento, California, and was instantly hooked on computer games. A few years later, he was introduced to the world of programming by his godfather, who let him use his Radio Shack TRS-80 computer to learn about programming in BASIC. David's first program was a simple dice game that graphically displayed the die face (he still has the first version he originally wrote on paper). He quickly outgrew BASIC though, and soon discovered the amazing speed you could get by writing video games in assembly language. He spent the remainder of his high school years getting bad grades, but writing cool software, none of which made him any money. He spent the next 10 years in the military, learning details about computer systems and software development. Shortly after he left the military, David was offered a job to help build the Space Station Training Facility at NASA. From that point on, he merrily spent time working on visual simulation and virtual reality applications. He made the odd shift into multitier IT application development during the Internet boom, ultimately landing inside of Microsoft as a technical evangelist, where he spends time playing with all sorts of new technology and merrily saying under his breath, "I can't believe people pay me to have this much fun!"

Alexandre Santos Lobão got his first computer in 1981, when he was 12, and immediately started to create simple games in BASIC. Since then, computers have evolved massively, and so has he. Graduating with a bachelor's degree in computer science in 1991, Alexandre, together with six friends, founded that same year a company that came to be known as a synonym for high-quality services in Brasilia, Brazil: Hepta Informática.

Besides his excellent work in many software development areas, from financial to telecommunication, he never forgot his first passion, and has always worked as a nonprofessional game programmer. From 1997 to 1999 he also worked at Virtually Real (`http://www.vrealware.com`), a virtual Australian amateur game programming company founded by Craig Jardine.

At the end of 2000, Alexandre started searching for new horizons and, leaving the company he helped to create, entered Microsoft as a consultant. Looking at the new and extremely interesting possibilities offered by the .NET Framework, he decided to take everything he's learned over the last decade and apply it to this new development platform.

Ellen Hatton is a computer science undergraduate at Edinburgh University. She was exposed to computers at a very early age and has been fascinated with them ever since. Her first experience with computer games was playing Dread Dragon Doom, at which she quickly excelled at the age of 5. She's been hooked on games ever since.

Ellen is not only interested in computers. She skis frequently, amongst other sports, and enjoys general student life in the bustling Scottish capital, Edinburgh.

As her choice of degree suggests, Ellen still finds computers very interesting and is constantly looking for new challenges. This book is the latest.

About the Technical Reviewer

Andrew Jenks began writing code when his parents bought him a TI 99-4A for a Christmas present. As tape drives were hard to use, and the media resulting was often overwritten by singing siblings, his father brought home their first family computer in 1985. Andrew learned to write BASIC and assembly programs through old Sanyo manuals and whatever he could find in the library. This proved handy when he found himself broke at the Georgia Institute of Technology and discovered that people would pay him to teach computing classes. He went on to act as a developer for an artificial intelligence company, manager for a communication company at the 1996 Olympics, and a technical advisor for several political campaigns. Andrew joined Microsoft as a program manager in 2000 and can currently be found working on MSJVM migration issues when he's not off skiing or diving.

During Andrew's illustrious career as a professional geek, he has written code that caused several graphics cards to make pretty blue sparks, lost one monitor due to a long fall, and set one machine on fire. He is most proud of the fire. That was good code.

Credits

Figure 6-13: Serious Sam® ©2001 is a trademark of Croteam Ltd. All rights reserved.

Figure C-1: Quake® is a trademark of Id Software, Inc. All rights reserved.

Figure C-4: PAC-MAN® ©1980 Namco Ltd. All rights reserved. Courtesy of Namco Holding Corp.

Figure C-5: Super Mario Bros. 2® © 1988 by Nintendo of America Inc.

Figure C-6: GALAGA® ©1980 Namco Ltd. All rights reserved. Courtesy of Namco Holding Corp.

Figure C-7: GAUNTLET® DARK LEGACY™ © 1998–2000 Midway Games West Inc. GAUNTLET DARK LEGACY is a trademark of Midway Games West Inc.

Acknowledgments

Tools and Tunes

To begin with, no development effort can be done without tools. There tools were invaluable to me, and I heartily recommend them as "must have" tools:

- *IDE:* Visual Studio .NET Professional 2003 (`http://www.microsoft.com/catalog/display.asp?subid=22&site=11513&x=30&y=4`)
- *Source control:* SourceGear's Vault (`http://www.sourcegear.com/vault`)
- DirectX 9 SDK (`http://www.microsoft.com/directx`)

I also want to thank those that kept me rocking while typing: Prodigy, Ghetto Boys, Radiohead, Everclear, AC/DC, Christopher Parkening, Elliot Fisk, Jimmy Buffett, Fleetwood Mac, the cast of the movie *Chicago,* Shakira, Norah Jones, Alejandro Sanz, Juanes, and many, many more.

People Who Really Made This Happen

Few authors can write a book completely by themselves, and I'm no exception to this rule. First and foremost, this book could not have been done without the coding wisdom of Scott Haynie. He converted the Spacewar game and wrote the bulk of the code for the Spacewar3D game. In addition, he gladly contributed the 3-D models for the Spacewar3D game. This book would have been very different without his help and ideas, and he has my undying gratitude.

In addition, other people helped by contributing code or offering suggestions. Tristian Cartony (.Nettrix), Stephen Toub (.Netterpillars), Carole Snyder, and Franklin Munoz. For anybody else who contributed that I forgot to call out by name, please accept my apologies in advance.

There are two other people I'd especially like to thank: Tom Miller, the principal developer of the Managed DirectX libraries, graciously whacked me over the head several times saying, "What were you thinking?!" Without his (if you'll pardon the pun) direct input, we might have taught some beginners some very bad Managed DirectX habits. And, of course, Sofia Marchant, the project manager for this book, who did a great job of being my "velvet-gloved taskmaster" as well, making sure I was staying on schedule to get this book done on time.

Lastly on the list are the people who have quietly (or not-so-quietly) influenced this book:

- My godfather, Charles Plott, who opened up my eyes to the world of computers and computer games.

- My high school math teacher, Duane Peterson, who let me take a computer programming class in spite of not knowing enough math—the result of which inspired me to get a degree in computer science with a math minor.

- My mom and dad, who put up with my intense passion for computers during my adolescence, in spite of not having enough money to buy me the mainframe system I wanted to put in our garage.

- My kids, Erich and Gretchen, and their mother, Nancy, who patiently tolerated my passion for computer games for many years.

Lastly, I want to thank my girlfriend Ana, who has made some very gloomy days for me much brighter, and who gave me all the support she could, even though she was 2000 miles away most of the time.

—David Weller

Preface

I approached Alexandre about a year ago to offer him comments on his first book, *.NET Game Programming with DirectX 9.0*. After presenting him with a rather long list of what I would have done differently, Alex graciously suggested collaborating on a new book. We decided early in the process to reuse some of the game examples from his book (specifically .Nettrix and .Netterpillars), although some parts have been heavily modified. We did this for two reasons:

- The games are good, simple examples that can stand the test of time when it comes to learning game programming. There was no sense creating a different game just to convey the same concept.

- Writing different games from scratch would take time away from adding newer games at the end of the book that challenged the beginner.

Of course, my youthful memories of the early computer games influenced me to choose a space theme for the later games, leaning on the well-known games of Asteroids and Spacewar. But I wanted to take things a step further, to show how 2-D gaming knowledge can quickly scale into 3-D games. I had never seen a book take such a step, and was frankly worried that it couldn't be done effectively. However, the book you're holding is the best attempt I can put forward, and hopefully you'll find the progression simple as well as instructional.

Due to my distaste for gaming books that double as gymnasium free weights, I wanted to create a book that avoided the long, pointless chapters that explained Visual Basic .NET (henceforth referred to as "VB"), object-oriented programming, how to use Visual Studio, etc. This book gets right to the games, and assumes you have a rudimentary knowledge of VB. If you need to get up to speed on VB, we recommend Matthew Tagliaferri's *Learn VB .NET Through Game Programming* (Apress, ISBN 1-59059-114-3), which makes an excellent companion book to this one.

For developers who are already familiar with programming and basic gaming concepts, this book will serve well as a high-speed introduction to Visual Basic .NET and, in later chapters, Managed DirectX. If you're already intimately familiar with DirectX game development and are looking for a book focused directly on Managed DirectX, I recommend *Managed DirectX Kick Start* (SAMS, 2003) written by Tom Miller. Of course, I would love for you to buy this book as well, but I'm more interested in getting you to write games in Managed DirectX than I am in making a buck or two by convincing you to buy this book.

The whole book is designed to be read in a continuous way. In Chapter 1, we start by creating a very simple game while presenting the basics of collision detection. Chapter 2 shows how to build a new game, using the concepts presented in Chapter 1 and adding new explanations and examples about artificial intelligence in games.

In the following chapters, we continue to build new games and explore new topics relating to game programming, such as the basics of sprite creation, multiplayer features, 3-D graphics, porting a game to Pocket PC, and much more. We start with the basics and increase the complexity as we go along, so that by the time you come to the advanced topics, you have all the background you need to gain the most from them. Near the end of the book, we stick our toes in the deeper DirectX waters by investigating point sprites. I have yet to see a book that discusses point sprites in a good, introductory style, so even intermediate game developers should find this part interesting.

Please keep in mind though that this book isn't intended to provide a route to the professional game programming world, because we don't go deep enough into some essential aspects professional game developers need to know. However, you can think of this book as a first step into this world, since we do provide insights into important concepts such as the need to create a good game project and organizing the game's team, as well as appendixes written by professionals from the game industry that serve as guides to game creation.

—David Weller