Programming The Nintendo Game Boy Advance: The Unofficial Guide

Programming The Nintendo Game Boy Advance: The Unofficial Guide

Jonathan S. Harbour

Project Editor: Estelle Manticas

Copy Editor: Laura Gabler

Technical Reviewer: André LaMothe

Programming Reviewer: Emanuel Schleussinger

Software Reviewer: Peter Schraut

ISBN: 1-59200-009-6

Printed in the United States of America

03 04 05 06 07 BH 10 9 8 7 6 5 4 3 2 1

For selfless support and love, for accepting

the inequity of my preoccupation with games

and fully supporting it, this book is dedicated

to my wonderful wife,

Jennifer Rebecca Harbour

Acknowledgements

I am thankful for the encouragement of my wife, Jennifer, who has been wonderfully supportive of my writing career. I love our life together, and our kids, Jeremiah and Kayleigh, bring me more joy than I thought possible. I also owe a thank you to two pair of friends who have greatly encouraged me on a personal level: Justin and Kim Galloway, and Milford “Wade” and Lindsay Eutsey. I thank God for the blessing of your friendship. I am grateful for a supportive family: my parents, Ed & Vicki; my sister, Joy; my niece, April; in-laws, Dave & Barb Yoder; David; Steve, Andrea, and Stephen; Jason, Autumn, and Tater; and Barbara Jean. On another level, I thank the Lord for giving me the drive to do this sort of work, which is both enjoyable and insanely difficult at the same time. I am thankful for having learned many life lessons from Focus On The Family and Dr. James Dobson, as well as from the extraordinary *Left Behind* series by Tim LaHaye and Jerry Jenkins.

This was perhaps the most enjoyable book I have written, not solely due to the content (which is utterly compelling!), but also due to two events that took place while this book was being written. The first experience was during the Game Developer's Conference 2003, where I met several editors from Premier Press face-to-face for the first time. After the Game Developer's Choice Awards, presented by the International Game Developer's Association (IGDA), Premier Press held a private dinner at a nice restaurant in downtown San Jose, which was a fun meet-and-greet with André LaMothe, Heather Hurley, Mitzi Koontz, Heather Talbot, Todd Jensen, as well as other fellow authors. Thanks to all of you for doing such a great job with this series; I have enjoyed working with all of you. The second experience that made this book memorable was a celebration a few days later on the launch of *DarkBASIC* with fellow author Joshua Smith, our families, and my two favorite freelancers, Cathleen Snyer and Estelle Manticas. Thank you for doing such great work, it's been a great pleasure working with you.

I would like to thank one of the greatest game designers of all time, John Romero, for not only being such a great encouragement and role model for aspiring game developers, but also for agreeing to write the foreword. As Jacopo says, "I'm your man!" Thanks to my good friend, Joshua R. Smith, for loaning me the Flash Advance Linker.

Greets go out to two guys who were invaluable while writing this book, for they provided up-to-the-minute updates to the development tools. Emanuel Schleussinger, for the excellent GBA SDK used in this book, called *HAM*, and to Peter Schraut for the excellent IDE

called *Visual HAM*. These tools saved me literally hundreds of hours of work. Not only did they provide superior development tools to me and the GBA community at no charge, these two were invaluable as proofreaders of the manuscript. Thank you both for getting me out of many dead ends and helping resolve coding problems along the way. This book reflects your efforts.

A book of any size and on any subject involves much work even after a manuscript is done. It takes a while just to read through a programming book once, so you can imagine how much is involved in reading through it several times, making changes and notes along the way, refining, correcting, perfecting. I am indebted to the hard work of the editors, artists, layout specialists who do such a fine job. Really, as far as total hours go, the author's work is only perhaps 40% (or less) of the total time spent on getting a book into print. Thank you especially to Laura Gabler for copy editing the manuscript, to André LaMothe for his technical expertise, and to Estelle Manticas for managing the project.

About The Author

Jonathan S. Harbour has been programming games for 15 years, first with Microsoft GW-BASIC and Turbo Pascal, then on to Turbo C, Borland C++, Watcom C++, and 80386 assembler. After finally bridging the gap to Windows programming, he has spent time with Borland Delphi, Visual Basic, Visual C++, and more recently, Visual Studio .NET. Jonathan graduated from DeVry Institute of Technology in 1997 with a B.S. degree in Computer Information Systems, and has since worked for cellular, aerospace, pharmaceutical, education, medical research, healthcare, and game companies. In his spare time, Jonathan enjoys spending time with his family, reading fiction, playing console video games, watching movies, and working on his classic Mustangs.

Contents

Part III
Meditating On The Hardware

Foreword

For over 10 years, programming the Game Boy has been a mystery. Unless you are an approved Nintendo developer, the only way to learn how to develop on the Game Boy family has been to piece together a million scraps of information off the internet, play around with secret register addresses you heard whispered somewhere, and basically just shove lots of crazy values in the register lists you've pieced together to see just what would happen.

But now, that era has passed. Jonathan Harbour has tackled the huge task of compiling all this esoteric and cryptic information into the most useful Game Boy programming book you'll find. Even Nintendo's documentation doesn't read this good (and we're Approved Nintendo Developers, so we know) and the example code and explanations are now understandable to English-reading lifeforms.

You'll find out soon that programming the Game Boy is really not that difficult if you're familiar with C programming and using pointers. We designed our game, *Hyperspace Delivery Boy!*, originally for the Pocket PC computer. When we ported the game to the Game Boy Advance, we could still retain our original cross-platform code architecture and just rewrite the areas of the game that dealt with graphics, sound, and file access (since it's now all just memory in ROM). It's always a great idea to write your game for your target platform and take advantage of its strengths, but coding the Game Boy Advance doesn't totally require you to do that and still create a successful game.

What are you waiting for? Turn the page and get ready to make your own game on a cartridge!

John Romero

Game Designer

http://www.monkeystone.com

Introduction

The Game Boy Advance is a sophisticated handheld video game machine, with a powerful 32-bit microprocessor, 16-bit graphics, stereo digital sound, and yet small enough to fit in your pocket. Millions of Game Boy Advance units have been sold already, with upwards of 100,000 units a week shipping around the world during peak months, making this the highest selling video game system in history, with over 140 million units sold in all. The affordable price of the Game Boys, large library of games, relatively low price of games and accessories, plus portability make this a very compelling video game system. Investing in development personnel and tools is more likely on a prolific system like Game Boy Advance, which has a long life span (like earlier Game Boy models). 14 years, 4 models, and 600 games later (at the time of this writing), Game Boy is the most successful video game franchise in history, outselling all others by a wide margin. Adding icing to the cake, the latest Game Boy fully supports all the game cartridges from the earlier models!

Game publishing companies and development studios like the Game Boy for its relatively low cost of entry and high potential margins. While a Game Boy cartridge might cost more to produce than a CD-ROM or DVD game disc, there are many factors that amount to strong profits nonetheless. One factor is that Game Boy is almost universally considered an accessory device, not a primary video game system. As such, families with one or more children will often have a console (such as the Nintendo GameCube) in the living room, as well as a Game Boy. Families with two or more children are also likely to buy a Game Boy for each child. Although the demographic for Game Boy users is primarily young children, a large percentage of Game Boy owners are young adults (18-24) and older.

Game Boy Development

Until recently, it was nearly impossible for a hobby or student programmer to get involved in the console market. The problem is this: in order to get hired, console game developers require some level of experience with the hardware; however, one can't gain that experience without first being hired. The key word here is *hardware*. Anyone can gain experience writing games for a PC, because the development machine is equivalent to the end-user's machine. But this is not the case with consoles such as the Game Boy Advance, which requires a custom hardware interface, special development tools (the compiler, linker, emulator, link cables, etc). These tools are very expensive, and require a special license with the video game manufacturer (Nintendo, in this case), which also requires a

non-disclosure agreement (NDA). This agreement says that you are not to share the secrets of the console or development tools with anyone who is not an officially licensed developer. In other words, console games are developed safely behind closed doors.

One solution for the aspiring programmer was usually to find employment with a game developer as a game tester (also called QA, or quality assurance person), and then gradually learn the ropes by working with the programmers while testing games, with the hope of being promoted to a development position. Another possibility is to find employment with a PC game studio, and gain experience working on PC games before moving on to consoles. Experienced developers are usually easy to train on console hardware, so they are often considered for employment. The PC market is much easier to break into than the console market. Either of these solutions requires patience, determination, drive, and passion for writing games and learning the tricks of the trade, and no one is guaranteed to succeed. Most find enough satisfaction out of hobby programming that they are content in that respect. But the video game industry (like the computer game industry) is growing by leaps and bounds, so the need for highly motivated and talented programmers is greater today than at any time before.

So You Want To Be A Console Programmer?

There is now a solution to the circular problem of finding a job as a console game developer. The Game Boy Advance uses a popular microprocessor (the ARM7 CPU) for which there are software tools available, because this chip is used in hundreds of consumer electronics products: VCRs, DVD players, TVs, satellite receivers, MP3 players, DSL and cable modems, Pocket PCs, GPS transceivers, in addition to the Sega Dreamcast and Nintendo Game Boy Advance. In case you are wondering, the Dreamcast featured a powerful 3D chip in addition to its ARM CPU, and the Dreamcast CPU and Game Boy Advance CPU are similar (though the Dreamcast is faster).

The result is that public domain assemblers and compilers have been written for ARM processors, and the ARM Corporation itself has released tools into the public domain for use with these products. These tools were quickly adapted for Game Boy Advance, and talented programmers were creating emulators and writing Game Boy Advance programs before the handheld was even officially released by Nintendo. (The same was true for the Dreamcast as well).

Software Development Kits

You are probably wondering at this point: how does one write Game Boy programs, let alone compile them and run them on an actual Game Boy? The development tools have matured in the two years (at the time of this writing) since the Game Boy Advance was released. There are now numerous public domain (or freeware) compilers, assemblers, and emulators vying for market share in the Game Boy development community.

The software development kit I selected for this book is a complete integrated development environment (IDE) for writing, compiling, and running Game Boy Advance programs, right on the Windows or Linux desktop. That package is called *HAM*, and was developed by Emanuel Schleussinger. The IDE is called *Visual HAM*, and was created by Peter Schraut. In order to test the programs, HAM comes with VisualBoyAdvance, a top-notch emulation program that runs compiled Game Boy Advance ROM images in Windows (or Linux). That's it! HAM is all you need to learn the art and science of programming the Game Boy Advance, and this wonderful development tool package is free.

Although there are tools available for Linux, I have not included any information in this book about using the Linux version of HAM. Since the same Game Boy Advance code will compile under Windows or Linux, you may wish to download the Linux version of HAM from Emanuel's Web site at http://www.ngine.de. The same ROM images that you will compile in each chapter of this book will run just as well in Linux as in Windows, because the same emulator is available for Linux.

Now, what happens when you have written a really cool game or demo and want to run it on a real Game Boy Advance unit? Well, there are basically two options. First, there is a multi-boot cable device that will download and run compiled ROM binaries on the Game Boy Advance, using the link cable port. Second, there is a more elaborate option. You can write to your own flash cartridge. This might sound difficult and expensive, but it is neither. A flash linker is a small device that plugs into your PC's parallel or USB port and reads/writes the flash cartridge. The process is similar to using a reader for SmartMedia, CompactFlash, and MMC/SD cards, with which you might be familiar if you have a digital camera. Once written, you can plug the flash cartridge into the Game Boy Advance just like any other game and power it up.

Obviously this is not a quick process, and requires a minute or two at least, and adds wear and tear to the Game Boy Advance and the cartridge over time. While it is fun to show your games and demos to friends, the multi-link cable is much faster, because it doesn't require

a cartridge. In fact, you leave the cartridge slot on the Game Boy Advance empty, plug in the multi-link cable, power up the Game Boy Advance, and the compiled program is downloaded and run directly in the little machine. Visual HAM and the HAM development kit comes with the emulator and the various download programs, ready to go. You are literally able to compile and run your own code on your Game Boy Advance. The multi-link cable and flash linker are covered in Chapter 4, "Starting With The Basics," along with details on where you can get one. One of the most popular Web sites for ordering these hardware items, as well as aftermarket Game Boy Advance accessories, is http://www.lik-sang.com.

Is This Book For You?

My goal with this book is to get you a job as a Game Boy Advance programmer. Although you may be a hobby programmer, or perhaps a professional video game developer already, my basis and assumption for this book is that you are an aspiring game programmer. The result is that I don't waste any time talking about subjects that won't help you in that respect. This is a very focused book that stays close to the subject matter, and as a result it is not as large as some books. However, most game programming books try to cover as much as possible. Those are books about writing PC games. This is a console programming book!

Some things are similar between consoles and PCs, but consoles are at a far lower level than PCs when it comes to game development. For one thing, at least on the Windows platform, all commercial games use the DirectX library, a monstrously large runtime for interfacing with computer hardware. In a sense, DirectX turns the hordes of different PCs into a single console platform. While console programmers are guaranteed that the machine will run exactly the same for every person, that is not the case with PCs. DirectX helps to homogenize all the PCs so programmers don't go insane trying to support all the different brands, with different CPUs, graphics chips, and so on (which is how it used to be back in the MS-DOS era).

You will need to be proficient in the C language in order to follow along in this book. Remember the assumption! If you want to be a game programmer, you must know C already. If you don't know C at all, you will definitely need a primer before getting into the later chapters of this book. There are many beginner books on C available, such as *C Programming for the Absolute Beginner* by Michael Vine. You needn't even buy a recent book on C, because Game Boy Advance programs follow the ANSI C standard, which has been around for decades. For example, one of my textbooks in college was *C: An Introduction to Programming* by Jim Keogh, et al, and I used this book as a reference while

writing Game Boy Advance code. C is an easy language to learn, but very difficult to master: that is part of the mystery of this language, and the source of it's widespread use. C is a very low-level language that is only a few degrees above assembly language.

I don't cover C++ because it is overkill for Game Boy Advance development. Most of the important aspects of programming the Game Boy Advance (such as sprites) are handled by the built-in hardware routines, and don't require extensive programmer intervention. For example, a sprite blitter is a given on the Game Boy Advance, so you will definitely not have to write one yourself! It's built into the hardware (which means, that functionality is provided by the manufacturer in the design of the system). However, I'm not going to argue the pros and cons, because I personally love the C++ language. C is simply easier to understand while learning the basics of Game Boy Advance programming. If you wish to use C++ yourself, you may do so, because the HAM toolkit does support C++.

This book does not cover C++, but you may still use C++ for writing Game Boy programs if you like, since the HAM SDK includes a C++ compiler.

System Requirements

The development tools required to write Game Boy Advance programs have very low system requirements, and will probably run just fine on any Windows-based PC. Even the lowliest old Pentium 133 will probably work, as long as DirectX is installed (for the emulator). However, here is a realistic minimum PC:

* Windows 95, NT, or later
* Pentium II 300 MHz
* 128 MB memory
* 200 MB hard drive space
* 8 MB standard video card

Book Summary

This book is divided into four parts:

Part I: The Zen of Getting Started

This first section of the book includes the introductory information you will need to get started in Game Boy Advance development. Included is an overview of the console industry,

the Game Boy Advance hardware, and how to install and use the HAM SDK (including Visual HAM and the VisualBoyAdvance emulator).

Part II: Being One With The Pixel

This section is dedicated to getting pixels on the Game Boy Advance screen, including chapters that cover all six video modes (three bitmap-based, and three tile-based), coverage of sprites, and also working with backgrounds, including special effects like panning, rotating, zooming, and alpha blending.

Part III: Meditating On The Hardware

This section focuses on the hardware aspects of the Game Boy Advance other than the video system, such as using interrupts and timers to retain a consistent frame rate, converting and playing sound files, checking for button presses, and using low-level assembly language to speed up code.

Part IV: The Mother of All Appendixes

This final section of the book includes the appendices, such as the ASCII chart, book and Web site listing, a Game Boy Advance hardware reference, answers to the chapter quizzes, and instructions for using the CD-ROM.