Preface

My career has been built on a long history of making "stupid" choices and accidentally being right. First, I went to Microsoft's DOS, instead of the wildly popular CP/M. Later, I recall, friends counseled me that Windows was dead, and too hard to program for, and that OS/2 was the future (you couldn't lose by sticking with IBM, they'd say).

Just got lucky, I guess.

There were a few other minor wrong turns that just happened to fortunately have me pointed away from some other collapsing industry segment, but my next really big stupid decision was writing the first edition of this book. I had already built a nice comfortable career out of fixing SQL database problems, and was making the transition to large-scale enterprise IT solutions in the healthcare industry. A book on OpenGL? I had no idea what I was doing. The first time I read the official OpenGL specification, I had to all but breathe in a paper bag, my first co-author quit in disgust, and the whole project was very nearly canceled before the book was half-finished.

As soon as the book came out, I had some meager credibility outside my normal field of expertise. I was offered a job at Lockheed-Martin/Real3D doing "real" OpenGL work. My then-current boss (God bless you, David, wherever you are!) tried really hard to talk me out of throwing my career away. Everybody knows, he insisted, that whatever Microsoft does is going to be the way the industry goes, and Microsoft's Talisman graphics platform was going to bury OpenGL into obscurity. Besides, there was only one other book on OpenGL in existence; how big a thing could it possibly be?

Eleven years have passed, and as I finish yet the fourth edition of this book (and looking at a shelf full of OpenGL books), the number of people reading this who remember the short-lived hype of Talisman would probably fit in the back of my minivan. An OpenGL engineer I used to know at IBM had in her e-mail signature: "OpenGL. It's everywhere. Do the math." This has never been truer than it is today.

OpenGL today is the industry-leading standard graphics API on nearly every conceivable platform. This includes not only desktop Windows PCs and Macs, but UNIX workstations, location-based entertainment systems, major game consoles (all but one), hand-held gaming devices, cellphones, and a myriad of other embedded systems such as avionic and vehicle instrumentation.

Across platforms, OpenGL is the undisputed champion of 3D content creation applications, 3D games, visualization, simulation, scientific modeling, and even 2D image and video editing. OpenGL's widespread success can be attributed to its elegance and ease of use, its power and flexibility, and the overwhelming support it has received from the

developer and IHV communities. OpenGL can be extended as well, providing all the benefits of an open standard, as well as giving vendors the ability to add their own proprietary added value to implementations.

You have probably heard that programmable hardware is the future of 3D graphics programming, and of graphics APIs. This is no longer true. Programmable hardware is no longer in the future; it is here now, today, even on the lowest cost motherboard embedded 3D chipsets. It is not a fluke that this edition follows the last at the closest interval of the series. The pace of evolving graphics technology is simply staggering, and this edition brings you up-to-date on the now-latest OpenGL version 2.1.

We have reinforced the chapters on fixed-pipeline programming, which is not going away anytime soon, and have affectionately deemed them "The Old Testament;" still relevant, illustrative, and the foundation on which the "New Testament" of programmable hardware is based. I find the analogy quite appropriate, and I would refute anyone who thinks the fixed pipeline is completely dead and irrelevant. The rank and file of application developers (not necessarily cutting-edge game developers) would, I'm sure, agree.

That said, we have still trimmed some dead weight. Color Index mode is ignored as much as possible, some old paletted rendering material from the Windows chapter has been pruned, and we have eliminated all the old low-level assembly-style shader material to make room for updated and expanded coverage of the high-level shading language (GLSL). You'll also find a whole new chapter on OpenGL on hand-held systems, totally rewritten Mac OS X and Linux chapters, and a really great new chapter on advanced buffer techniques such as offscreen rendering, and floating-point textures.

Another big change some readers will notice is that the OpenGL SuperBible has been acquired and adopted into the Addison-Wesley Professional OpenGL series. I can't begin to express how grateful I am and humbled I feel by this honor. I myself have worn out the covers on at least one edition of every volume in this series.

One of the reasons, I think, for the longevity of this book has been the unique approach it takes among OpenGL books. As much as possible, we look at things through the eyes of someone who is excited by 3D graphics but knows very little about the topic. The purpose of a tutorial is to get you started, not teach you everything you will ever need to know. Every professional knows that you never reach this place. I do occasionally get some criticism for glossing over things too much, or not explaining things according to the strictest engineering accuracy. These almost never come from those for whom this book was intended. We hope for a great many of you that this will be your first book on OpenGL and 3D graphics. We hope for none of you that it will be your last.

Well, I did make one really "smart" decision about my career once. Once upon a time in the early 1980s, I was a student looking at a computer in an electronics store. The salesman approached and began making his pitch. I told him I was just learning to program and was considering an Amiga over his model. I was briskly informed that I needed to get

serious with a computer that the rest of the world was using. An Amiga, he told me, was not good for anything but "making pretty pictures." No one, he assured me, could make a living making pretty pictures on his computer. Unfortunately, I listened to this "smart" advice and regretted it for over ten years. Thank God I finally got stupid.

As for making a living "making pretty pictures?" Do the math.

Oh, and my latest stupid decision? I've left Windows and switched to the Mac. Time will tell if my luck holds out.

—Richard S. Wright Jr.

Preface to the Previous, Third Edition

I have a confession to make. The first time I ever heard of OpenGL was at the 1992 Win32 Developers Conference in San Francisco. Windows NT 3.1 was in early beta (or late alpha), and many vendors were present, pledging their future support for this exciting new graphics technology. Among them was a company called Silicon Graphics, Inc. (SGI). The SGI representatives were showing off their graphics workstations and playing video demos of special effects from some popular movies. Their primary purpose in this booth, however, was to promote a new 3D graphics standard called OpenGL. It was based on SGI's proprietary IRIS GL and was fresh out of the box as a graphics standard. Significantly, Microsoft was pledging future support for OpenGL in Windows NT.

I had to wait until the beta release of NT 3.5 before I got my first personal taste of OpenGL. Those first OpenGL-based screensavers only scratched the surface of what was possible with this graphics API. Like many other people, I struggled through the Microsoft help files and bought a copy of the *OpenGL Programming Guide* (now called simply "The Red Book" by most). The Red Book was not a primer, however, and it assumed a lot of knowledge that I just didn't have.

Now for that confession I promised. How did I learn OpenGL? I learned it by writing a book about it. That's right, the first edition of the *OpenGL SuperBible* was me learning how to do 3D graphics myself…with a deadline! Somehow I pulled it off, and in 1996 the first edition of the book you are holding was born. Teaching myself OpenGL from scratch enabled me somehow to better explain the API to others in a manner that a lot of people seemed to like. The whole project was nearly canceled when Waite Group Press was acquired by another publisher halfway through the publishing process. Mitchell Waite stuck to his guns and insisted that OpenGL was going to be "the next big thing" in computer graphics. Vindication arrived when an emergency reprint was required because the first run of the book sold out before ever making it to the warehouse.

That was a long time ago, and in what seems like a galaxy far, far away…

Only three years later 3D accelerated graphics were a staple for even the most stripped-down PCs. The "API Wars," a political battle between Microsoft and SGI, had come and gone; OpenGL was firmly established in the PC world; and 3D hardware acceleration was as common as CD-ROMs and sound cards. I had even managed to turn my career more toward an OpenGL orientation and had the privilege of contributing in some small ways to the OpenGL specification for version 1.2 while working at Lockheed-Martin/Real3D. The second edition of this book, released at the end of 1999, was significantly expanded

and corrected. We even made some modest initial attempts to ensure that all the sample programs were more friendly in non-Windows platforms by using the GLUT framework.

Now, nearly five years later (eight since the first edition!), we bring you yet again another edition, the third, of this book. OpenGL is now without question the premier cross-platform real-time 3D graphics API. Excellent OpenGL stability and performance are available on even the most stripped-down bargain PC today. OpenGL is also the standard for UNIX and Linux operating systems, and Apple has made OpenGL a core fundamental technology for the new Mac OS X operating system. OpenGL is even making inroads via a new specification, OpenGL ES, into embedded and mobile spaces. Who would have thought five years ago that we would see Quake running on a cellphone?

It is exciting that, today, even laptops have 3D acceleration, and OpenGL is truly everywhere and on every mainstream computing platform. Even more exciting, however, is the continuing evolution of computer graphics hardware. Today, most graphics hardware is programmable, and OpenGL even has its own shading language, which can produce stunningly realistic graphics that were undreamed of on commodity hardware back in the last century (I just had to squeeze that in someplace!).

With this third edition, I am pleased that we have added Benjamin Lipchak as a co-author. Benj is primarily responsible for the chapters that deal with OpenGL shader programs; and coming from the ARB groups responsible for this aspect of OpenGL, he is one of the most qualified authors on this topic in the world.

We have also fully left behind the “Microsoft Specific” characteristics of the first edition and have embraced a more multiplatform approach. All the programming examples in this book have been tested on Windows, Mac OS X, and at least one version of Linux. There is even one chapter apiece on these operating systems, with information about using OpenGL with native applications.

—Richard S. Wright Jr.