Foreword

A long time ago, in terms of the innovation of computers (which is about fifteen years for those of us stuck in the real world), deploying a cross-platform application that rendered graphics was challenging, to put it mildly. At that time, one could argue that the problem was less about computer hardware, and much more about software and APIs. Vendors had their own libraries and methods for coercing hardware into drawing pictures for us, but they were all proprietary, and the applications that were deployed on multiple vendors' hardware had huge swaths of code living within the confines of #ifdef blocks. Okay, so that part hasn't been totally eliminated, but for graphics, the situation has clearly improved.

Around that time, a group of smart folks with lots of experience in creating computer hardware and the interfaces that drove them attempted to address the problem. While the library proposed was designed and implemented by a single company, Silicon Graphics Computer Systems, Inc., perhaps their most innovative idea was to share this interface with direct competitors, accepting and integrating their ideas and methods into the library. Hence, OpenGL was born into a large, loving family called the OpenGL Architecture Review Board (the "ARB"), which understood that computers and graphics were nothing without the applications that drive them.

If you consider the prospect of designing a programming interface that abstracts away all the differences that might exist between different implementations of graphics algorithms, this was no small task. Furthermore, for the interface to be accepted and used, it had to be simple enough to get tasks done, but also had to meet the requirements of interactive, 3D computer graphics, which is to be as efficient as possible. Rarely is it easy to be both simple to use and highly efficient. In a way, that's the essence of software engineering, and this interface was being designed while that discipline was still formulating its basic principles.

To make matters more complicated, the authors of OpenGL knew that the computer graphics field was barely out of its infancy at that point. The rapid evolution of graphics techniques and their hardware implementations was

already in full swing; if OpenGL were to be a contender, it would need to evolve with the field.

All heady stuff, to say the least.

Now, fast-forward to the present. The book you're holding is about OpenGL, the dominant programming interface for cross-platform, 3D computer graphics and image processing—the same interface that was designed those many years ago. It's remained true to its heritage, and is easy for novices to use to create interactive applications using 3D graphics, but powerful enough to drive the most performance-sensitive graphics applications. To that end, however, the interface (at version 2.0 at the time of this writing), has more than 800 entry points exposing current features of computer graphics in OpenGL parlance. Navigating through all those functions and methods is a daunting task, and an experienced guide would be very helpful.

Enter Paul Martz, who's been a long-time member of the graphics community, watching and contributing to OpenGL's evolution. He's offered to share his experience to help make your path to OpenGL mastery as painless as possible. Paul has considered the entirety of OpenGL and then distilled the essential elements of modern OpenGL programming and best practices into these pages to be your guide. I congratulate him on his accomplishment, and wish you the best of luck with OpenGL.

Happy rendering!

Dave Shreiner
Mountain View, CA
January 2006