Foreword

To me, graphics shaders are about the coolest things to ever happen in computer graphics. I grew up in graphics in the 1970s, watching the most amazing people do the most amazing things with the mathematics of graphics. I remember Jim Blinn's bump-mapping technique, for instance, and what effects it was able to create. The method was deceptively simple, but the visual impact was momentous. True, it took a substantial amount of time for a computer to work through the pixel-by-pixel software process to make that resulting image, but we only cared about that a little bit. It was the effect that mattered.

My memory now fast-forwards to the 1980s. Speed became a major issue, with practitioners like Jim Clark working on placing graphics algorithms in silicon. This resulted in the blossoming of companies such as Evans & Sutherland and Silicon Graphics. They brought fast, interactive 3D graphics to the masses, but the compromise was that they forced us into doing our work using standard APIs that could easily be hardware supported. Deep-down procedural techniques such as bump-mapping could not follow where the hardware was leading.

But the amazing techniques survived in software. Rob Cook's classic paper on shade trees brought attention to the idea of using software "shaders" to perform the pixel-by-pixel computations that could deliver the great effects. This was embodied by the Photorealistic RenderMan rendering software. The book *RenderMan Companion* by Steve Upstill is still the first reference that I point my students to when they want to learn about the inner workings of shaders. The ability to achieve such fine-grained control over the graphics rendering process gave RenderMan users the ability to create the

dazzling, realistic effects seen in Pixar animation shorts and TV commercials. The process was still miles away from real time, but the seed of the idea of giving an *interactive* application developer that type of control was planted. And it was such a powerful idea that it was only a matter of time until it grew.

Now, fast-forward to the start of the new millennium. The major influence on graphics was no longer science and engineering applications. It had become games and other forms of entertainment. (Nowhere has this been more obvious than in the composition of the SIGGRAPH Exhibition.) Because games live and die by their ability to deliver realistic effects at interactive speeds, the shader seed planted a few years earlier was ready to flourish in this new domain. The capacity to place procedural graphics rendering algorithms into the graphics hardware was definitely an idea whose time had come. Interestingly, it brought the graphics community full circle. We searched old SIGGRAPH proceedings to see how pixel-by-pixel scene control was performed in software then, so we could "re-invent" it using interactive shader code.

So, here we are in the present, reading Randi Rost's *OpenGL® Shading Language*. This is the next book I point my shader-intrigued students to, after Upstill's. It is also the one that I, and they, use most often day to day. By now, my first edition is pretty worn.

But great news—I have an excuse to replace it! This second edition is a *major* enhancement over the first. This is more than just errata corrections. There is substantial new material in this book. New chapters on lighting, shadows, surface characteristics, and RealWorldz are essential for serious effects programmers. There are also 18 new shader examples. The ones I especially like are shadow mapping, vertex noise, image-based lighting, and environmental mapping with cube maps. But they are all really good, and you will find them all useful.

The OpenGL Shading Language is now part of standard OpenGL. It will be used everywhere. There is no reason not to. Anybody interested in effects graphics programming will want to read this book cover to cover. There are many nuggets to uncover. But GLSL is useful even beyond those borders. For example, we use it in our visualization research here at OSU (dome transformation, line integral convolution, image compression, terrain data mapping, etc.). I know that GLSL will find considerable applications in many other non-game areas as well.

I want to express my appreciation to Randi, who obviously started working on the first edition of this book even before the GLSL specification was fully

decided upon. This must have made the book extra difficult to write, but it let the rest of us jump on the information as soon as it was stable. Thanks, too, for this second edition. It will make a significant contribution to the shader-programming community, and we appreciate it.

—Mike Bailey, Ph.D.
Professor, Computer Science
Oregon State University