Foreword

Before the advent of dedicated PC graphics hardware, the industry's first 3D games used CPU-based software rendering. I wrote the first Unreal Engine in that era, inspired by John Carmack's pioneering programming work on *Doom* and *Quake*. Despite slow CPUs and low resolutions, the mid-1990s became a watershed time for graphics and gaming. New visual effects appeared almost monthly, marked by milestones like *Quake*'s light mapping and shadowing and Unreal's colored lighting and volumetric fog. That era faded away as fixed-function 3D accelerators appeared. Deprived of the programmability that drove innovation and differentiation, 3D games grew indistinct.

Today, a new Renaissance in 3D graphics is under way, driven by fully programmable GPUs—*graphics processing units*—that deliver thousands of times the graphics power available just ten years ago. Combining incredible parallel computing power with modern, high-level programming languages, today's GPUs have unleashed a Cambrian Explosion of innovation and creativity. Real-time soft shadowing, accurate lighting models, and realistic material interactions are readily achievable. But the most important gain of programmability is that you can do *anything* with a GPU so long as you can find an algorithm to express your idea. *GPU Gems 2* demonstrates many such ideas-turned-algorithms.

Let us take a moment to review the set of resources available to today's graphics programmer. First, you have access to a GPU that can perform tens of billions of floating-point calculations per second in programmable shading algorithms. It's your workhorse; if you can move your problem into the realm of pixels and vertices, then you can harness the GPU's immense power. Second, you have a CPU, the system's general-purpose computing engine. The CPU sends commands to the graphics processing unit, manages resources, and interacts with the outside world. Finally, you have access to artistic content—texture maps, meshes, and other multimedia data that the GPU can combine, filter, and procedurally modify during rendering.

The Gems in this book employ these resources in novel ways to render realistic scenes, process images, and produce special effects. In doing so, many of the previous era's

graphics rules may be broken. GPUs are fast and flexible enough that you may render a given object many times, decomposing a scene into its components—lighting, shadowing, reflections, post-processing effects, and so on. You can employ the GPU for decidedly non-graphics tasks like collision detection, physics, and numerical computation; and within texture maps you can encode arbitrary data, such as vectors, positions, or lookup tables used by shader programs. And while visual realism is now achievable on GPUs, it is not your only option: nonphotorealistic rendering techniques are available, such as cel shading, exaggerated motion blur and light blooms, and other effects seen frequently in Hollywood productions.

Seven years after I wrote Unreal's original software renderer, my company began developing a new game engine, Unreal Engine 3, designed for the capabilities of today's modern GPUs. It has been an incredible experience! Where we once built 300-polygon scenes with static lighting and texture maps, we now combine dynamic per-pixel lighting and shadowing with realistic material effects in million-polygon scenes. We've seen an explosive growth in the power and flexibility available to programmers and artists alike. But while much has changed in graphics development, several truths have remained: that graphics requires a unique combination of engineering, artistry, and invention unmatched in other fields; that innovation moves at an incredible pace as hardware performance increases exponentially; and that graphics programming is *a heck of a lot of fun*!

Here in *GPU Gems 2*, you'll find a wealth of knowledge and insight, plus many just plain neat ideas, which can be readily applied on today's graphics hardware. But the techniques here are only a starting point on your adventure—the real fun and opportunity lie in finding new ways to customize and combine these Gems and to invent new ones.

Tim Sweeney
Founder and Technical Director, Epic Games

Screenshots from Unreal Engine 3 Technology Demo, http://www.unrealtechnology.com