FOREWORD

What is a realistic image? This is an age-old question in art, and a contemporary question in computer graphics. This book provides a modern answer involving the computer and a new definition of realism.

The classic definition of realism has been veridical realism. Does the picture pass the comparison test? That is, would an observer judge the picture to be real? This is traditionally described by Pliny's story (in Book 35 of his *Natural History*) of the ancient painter Zeuxis who painted a picture of a boy carrying some grapes, and when the birds flew up to the picture, he approached the work and, in irritation, said, "I have painted the grapes better than the boy, for if I had rendered him perfectly, the birds would have been afraid."

Nowadays the ultimate in fooling the eye is special effects in the movies. Almost every movie involves hundreds of special effects that are seamlessly combined with live action. It is impossible to tell what is real and what is synthesized. Equally amazing are full-length, computer-generated pictures such as *Shrek*. Although few would be fooled into believing these worlds are real, it is more the artistic choice of the storyteller than a technological limitation. A major achievement in the last two decades is that computers allowed us to achieve veridical realism of imagined scenes.

Besides direct comparison, there are other definitions of real. Masters such as Vermeer used optical devices to aid them in painting realistic pictures, and modern photorealists such as Richard Estes paint over a projected image of a photograph. Thus, another definition of real is to be traced or copied from an image. In this sense the montage of composite layers in a movie is photoreal, since different elements come from different film sequences. There are many other definitions of realism. For example, real can mean a choice of subject matter, such as everyday life versus a myth or an idealized form.

The definition of realism that I like the most is the one I first heard from my colleague, then at Pixar, Alvy Ray Smith: he claimed photorealism was roughly equivalent to visual complexity. Two factors underlie visual complexity, diversity in the types of primitives and their sheer numbers. This definition resonates with computer scientists, since computers are very good at both supporting a wide range of computational primitives and processing enormous amounts of data. This book is about using the computer to generate visual complexity, an approach called procedural modeling.

The causes of visual complexity in the computer-generated image are the ingredients of perception: color, texture, edges, depth, and motion. The equivalents in object-space, or in the scene, are color, pattern, reflection, illumination, shape, and motion. All these factors come together in composite materials such as wood, stone, and cloth and in natural phenomena such as clouds, steam, smoke, fire, water, landscapes, and planetoids. Procedural models for these myriad objects are the subjects of this book.

Why are computers so good at generating visual complexity? The reason is profound as well as practical.

First, computers expand the types of models that may be used. For example, a surface may be defined as the zeros of an implicit function of x, y, and z. The simplest implicit functions are quadratic functions of the coordinates and define the famous quadric surfaces: spheres, cones, cylinders, and so on. Using a modern programming language with all its built-in functions and arithmetic operators, much more complicated expressions are just as easy to form and to evaluate. Perlin's hypertextured surfaces arise from this flexibility and generality.

Second, computers can generate many from few. A few parameters (or a small amount of geometry) magically expand into a large, detailed model. "Data amplification" gives the user tremendous power, leveraging their efforts and offloading tedious specification of every single detail. A related concept is Kolmogorev complexity, or the smallest program capable of generating a given function. A very few lines of code can produce beautiful pictures. The hacker "demoscene" dramatically illustrates this idea. Here programmers are given the constraint that the size of the file containing both code and data (models, textures, sounds) must fit in less than 64KB. From this file emerges a richly detailed animation.

Third, computational models are by necessity discrete and finite. Although at first this may seem like a limitation, since computational procedures must approximate continuous mathematics and physics, it may in fact open up many new possibilities. For example, an approximation of a smooth curve may be generated from an n-sided polygon by a simple corner-cutting algorithm. Each step consists of cutting off all the corners of the polygon, replacing a vertex with an edge and two new vertices. After an infinite number of iterations of the cutting procedure, the input polygon will converge to a smooth curve. However, on a computer, we can never perform an infinite number of steps, so perfectly smooth curves can never be constructed. Once we give up on idealized mathematical smoothness, we can generalize corner-cutting polygons to subdividing 3D polyhedral meshes; although these new algorithms do not form smooth objects, a whole new universe of different types of curves and surfaces can now be generated on the computer.

For these reasons procedural modeling is a very powerful new tool that is enabled by the computer. This approach is what is fundamentally different about computer graphics and traditional forms of image making.

An important issue that remains, the Achilles heel of this approach, is controllability. Whether it is a physical simulation with its initial or boundary conditions, or a procedural model with its parameters, the end result must serve the needs of the user. The benefit of filling in detail automatically comes at a cost: the user loses control over the details. The need for controllability drives the development of interactive, what-you-see-is-what-you-get systems. This tension between precise control and programmed complexity remains an interesting research issue. In practice, virtual characters are usually modeled manually, and their motion is generated using key-frame animation. However, buildings, landscapes, and crowds are increasingly being generated using procedural techniques.

This new edition is particularly timely. Although the interest in procedural modeling subsided for a while, there has suddenly been an explosion of new research and development. Processing power continues to increase faster than human modeling power, and as a result models produced procedurally have a level of detail that cannot be produced by hand. New approaches have also emerged: machine learning has been coupled with procedural modeling so that it is now possible to analyze and then synthesize textures, shapes, motions, and styles from examples.

Another major new development is programmable graphics hardware. Graphics processing units, or GPUs, have always been increasing in performance much faster than CPUs. In the last few years, GPUs switched from a fixed-functionality to a flexible, programmable graphics pipeline. Now it is possible to download procedural models into these processors. Currently, GPUs are mostly limited to evaluating procedural texture and reflection models, but in the not too distant future they will be able to produce geometry and motion procedurally as well. Procedural models thus have technology on their side, since they use less bandwidth and communication resources than traditional approaches to graphics systems.

This book describes the complete toolbox of procedural techniques from theory to practice. The authors are the key inventors of the technology and some of the most creative individuals I know. This book has always been my favorite computer graphics book, and I hope you will enjoy it as much as I have.

Pat Hanrahan