Advanced Modeling Techniques for Computer Graphics

DAVID S. EBERT

Computer Science and Electrical Engineering Department, University of Maryland ⟨ebert@cs.umbc.edu⟩

In the past thirty years, modeling techniques in computer graphics have evolved significantly as the field has matured and attempted to portray the complexities of nature. Polygonal models, patches, points, and lines are insufficient to represent the complexities of natural objects and intricate man-made objects in a manageable and controllable fashion. Higher-level modeling techniques have been developed to provide an *abstraction* of the model, encode classes of objects, and allow high-level control and specification of the model. The goal of these advanced modeling techniques is to provide a concise, efficient, flexible, and controllable mechanism for specifying and animating models of complex objects and natural phenomena. Most of these advanced modeling techniques can be considered *procedural* modeling techniques: code segments or algorithms are used to abstract and encode the details of the model instead of explicitly storing vast numbers of low-level primitives. The use of algorithms unburdens the modeler/animator of low-level control, provides great flexibility, and allows amplification of his efforts through parametric control: a few parameters to the model yield large amounts of geometric details (Smith [1984] called this "database amplification"). This survey examines several types of procedural techniques, including fractals, grammar-based models, volumetric procedural models, implicit surfaces, and particle systems.

FRACTALS

Fractals [Peitgen et al. 1992] have a precise mathematical definition, but in computer graphics their definition has been extended to refer generally to models with a large degree of *self-similarity*: subpieces of the object appear to be scaled down, possibly translated and rotated versions of the original object. Along these lines, Musgrave [Ebert et al. 1994] define a fractal as "a geometrically complex object, the complexity of which arises through the repetition of form over some range of scale." Many natural objects exhibit this characteristic, including mountains, coastlines, trees, plants (e.g., cauliflower), water, and clouds. Fractals can generally be classified as deterministic or non-deterministic (also called random fractals), depending on whether they contain randomness.

Random fractals have been used extensively in computer graphics to model natural objects, most notably terrain. Most fractal terrain-generation algorithms work through recursive subdivision and pseudorandom perturbation. An original surface is defined and divided equally into subparts. New vertices are added and pseudorandomly displaced from the original surface, with a displacement magnitude that decreases at each iteration as the frequency increases. Therefore, the first iteration gives the large peaks on the surface, and later subdivisions add small-scale detail. Only the parameters for controlling the random-number generator, the level of subdivision, and the "roughness" of the surface are needed to define an extremely complex terrain. Recent work in fractals has included the simulation of diffusion-limited aggregation

(DLA) models and the use of *multi-fractals* [Ebert et al. 1994], which allows different fractal dimensions (degrees of "roughness") in the models to simulate natural terrain better.

GRAMMAR-BASED MODELS

Grammar-based models, primarily L-systems [Prusinkiewicz and Lindenmayer 1990], also allow natural complexity to be specified with a few parameters. Grammar-based models have been used by many authors, including Lindenmayer, Prusinkiewicz, and Fowler, to produce remarkably realistic models and images of trees, plants, and seashells. These models use formal languages, parallel graph grammars called L-systems, to describe natural structures algorithmically and are closely related to deterministic fractals in their self-similarity, but fail to meet the precise mathematical definition of a fractal.[1] An L-system is a formal language where all the rules are applied in parallel to provide a final "sentence" describing the object. In the L-system, each terminal symbol represents a part of the object or a directional command to be interpreted by a three-dimensional drawing mechanism (turtle graphics). A "sentence" for a tree would contain words describing each branch, its length, size, and branching angle, when it develops, and its connection in the tree. More complex L-systems, *IL-systems*, include context-sensitivity, word age information, and probabilistic rule evaluation, which allows each plant to be unique. Recent work in L-systems allows better developmental models, more advanced biologically based growth models, incorporation of more growth parameters, and environmental effects.

VOLUMETRIC PROCEDURAL MODELS

Another procedural modeling technique, volumetric procedural modeling (also called hypertextures, volume density functions, and fuzzy blobbies), uses algorithms to define and animate three-dimensional volumetric objects and natural phenomena [Ebert et al. 1994]. These techniques have been used to model natural phenomena such as fire (Stam and Inakage), gases such as smoke, clouds, and fog (Ebert, Perlin, Sakas, Stam), and water (Ebert, Perlin). The volumetric procedures take as input a point location in space, a time parameter, and parameters that describe the object being modeled, and return the density and color of the object for that location in space. Complex volumetric phenomena can, therefore, be described with a few parameters. Perlin has successfully used this technique to create realistic rock arches, woven fabric, smoke, and fur [Ebert et al. 1994], basing his procedures on a statistical simulation of turbulence and random noise to give natural-looking complexity to the objects. Ebert et al. [1994] have used similar functions to model and animate steam, fog, smoke, clouds, and solid marble. These procedural techniques allow the use of simple simulations of natural complexity (noise, turbulence) to speed computation, but also allow the incorporation of physically based parameters, where appropriate and feasible. This flexibility is one of the many advantages of procedural techniques.

IMPLICIT SURFACES

While previously discussed techniques have been used primarily for modeling the complexities of nature, implicit surfaces [Wyvill et al. 1986; Wyvill and Gascuel 1995] (also called blobby molecules, metaballs, and soft objects) have mainly been used for modeling organic shapes, complex man-made shapes, and "soft" objects that are difficult to animate and describe using more traditional techniques. Implicit surfaces are a more concise representation than parametric surfaces and provide flexibility in modeling and animating soft objects. Im-

[1]Some authors consider L-systems to be deterministic fractals.

plicit surfaces are iso-valued surfaces created from blending primitives (skeletal elements) represented by implicit equations of the form $F(x, y, z) = 0$. Each primitive is a procedure that returns a functional value for the field defined by the implicit equation. A key feature of implicit surfaces is the procedural, smooth, often volume-preserving blending of primitives to form quite complex surfaces from simple primitives. Objects are defined as offsets (isosurfaces) from a series of blended skeletal elements (points, lines, polygons, spheres, ellipsoids, and so on). Modeling and animation of implicit surfaces is achieved by controlling the skeletal elements and blending functions, which provide complex models and animations from a few parameters (another example of data amplification). Recent work in implicit surfaces [Wyvill and Gascuel 1995] has extended their use to character modeling and animation, human figure modeling, and representing rigid objects through the addition of CSG (constructive solid geometry) operators.

PARTICLE SYSTEMS

Particle systems differ from the previous four techniques in that their abstraction is in control of the animation and specification of the object. Particle systems do use a large database of geometric primitives to represent natural objects ("fuzzy objects"), but the animation, location, birth, and death of the particles representing the object are controlled algorithmically. Particle systems are most commonly used to represent natural phenomena such as fire, water, clouds, snow, rain, grass, and trees [Reeves and Blau 1985]. A particle-system object is represented by a large collection (cloud) of very simple geometric particles that change stochastically over time. The procedural aspect and main power of particle systems allow the specification and control of this extremely large cloud of geometric particles with very few parameters. Besides the geometric particles, a particle system has controllable stochastic particle-animation procedures that govern the creation, movement, and death of the particles. These animation procedures often include physically based forces to simulate effects such as gravity, vorticity, conservation of momentum, and energy. Particle systems pose special rendering problems because of the large number of primitives, but specialized rendering techniques, including probabilistic rendering algorithms, have been developed to render particle systems [Reeves and Blau 1985].

FUTURE DIRECTIONS

Advanced modeling techniques will continue to play an important role in computer graphics. As computers become more powerful, the complexity that can be rendered will increase; however, the ability of humans to specify more geometric complexity (millions of primitives) will not. Therefore, procedural techniques, with their ability to amplify the user's specification and control, are the only viable alternative. The ability of these techniques to specify and control incredibly realistic and detailed models with a small number of user-specified parameters will evolve. More work will be done to allow high-level control and specification of models in user-understandable terms, while more complex algorithms and improved physically based simulations will be incorporated into these procedures. Finally, automatic generation of the procedural models through artificial evolution techniques, similar to those of Sims [1994], will greatly enhance the capabilities and uses of these advanced modeling techniques.

REFERENCES

EBERT, D., MUSGRAVE, F. K., PEACHEY, D., PERLIN, K., AND WORLEY, S. 1994. *Texturing and Modeling: A Procedural Approach*. AP Professional, Boston, MA.

PEITGEN, H.-O., JÜRGENS, H., AND SAUPE, D. 1992. *Chaos and Fractals: New Frontiers of Science*. Springer-Verlag, New York.

PRUSINKIEWICZ, P. AND LINDENMAYER, A. 1990.

The Algorithmic Beauty of Plants. Springer-Verlag, New York.

REEVES, W. T. AND BLAU, R. 1985. Approximate and probabilistic algorithms for shading and rendering structured particle systems. *Comput. Graph. 19* (July) 313–322. (Proc. SIGGRAPH '85)

SIMS, K. 1994. Evolving virtual creatures. *Comput. Graph.* (July) 15–22. (Proc. SIGGRAPH '94)

SMITH, A. R. 1984. Plants, fractals and formal languages. *Comput. Graph. 18* (July) 1–10. (Proc. SIGGRAPH '84)

WYVILL, B. AND GASCUEL, M.-P. 1995. *Implicit Surfaces '95, The First International Workshop on Implicit Surfaces*. INRIA, Eurographics (April).

WYVILL, G., MCPHEETERS, C., AND WYVILL, B. 1986. Data structure for soft objects. *Visual Comput. 2*, 4 (Feb.), 227–234.