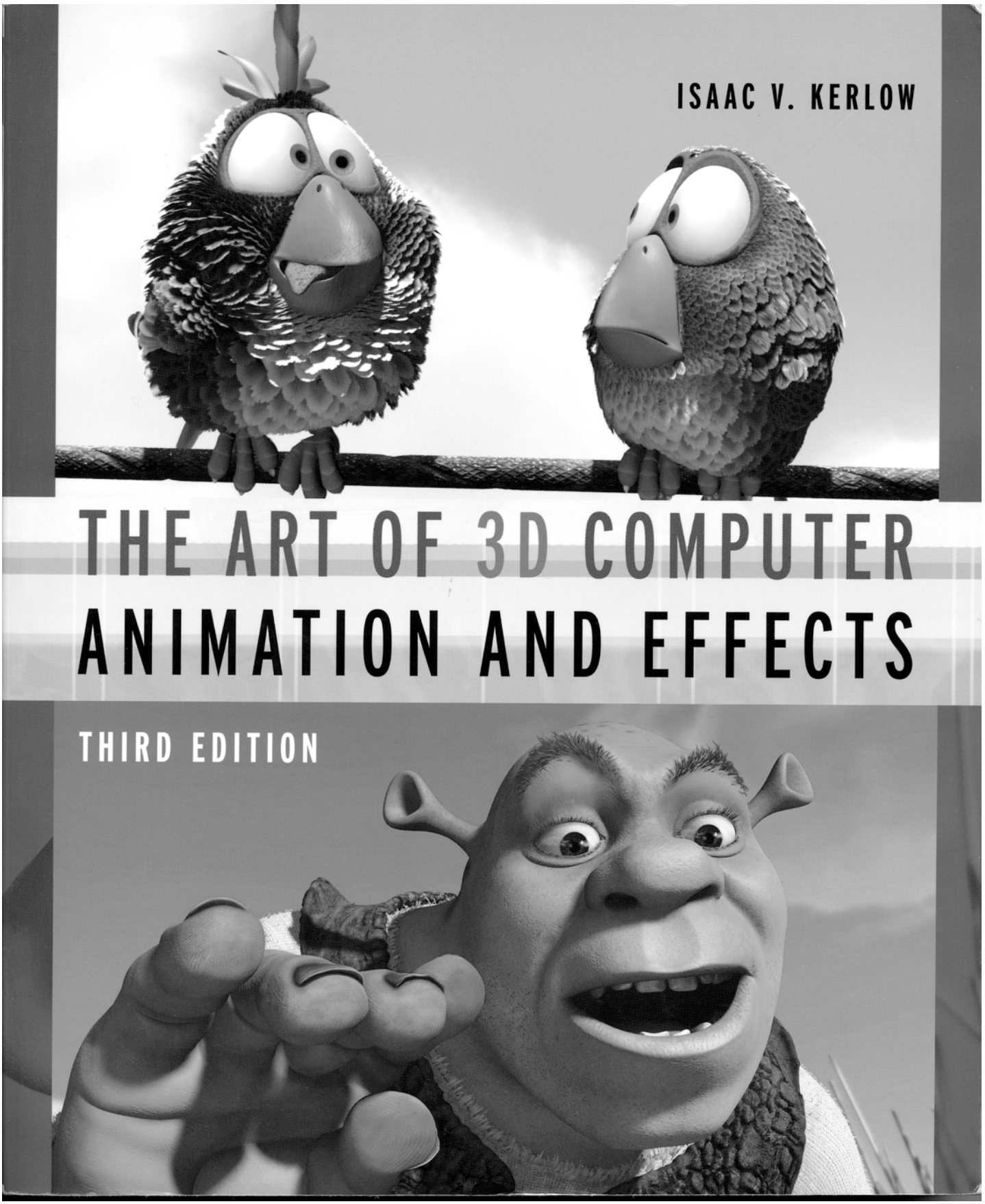
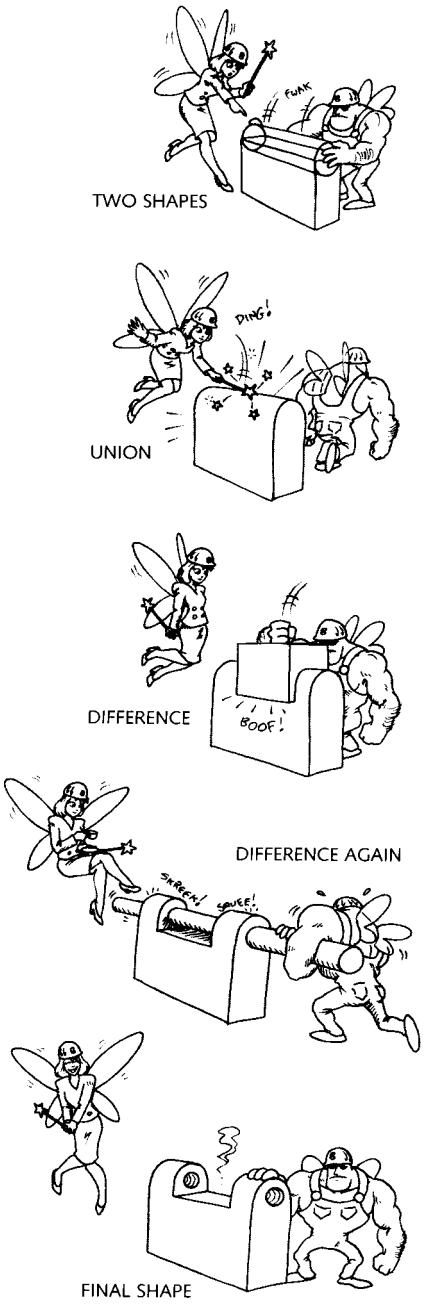


ISAAC V. KERLOW

THE ART OF 3D COMPUTER ANIMATION AND EFFECTS

THIRD EDITION



5.3.2 This sequence illustrates the logical operators of union and difference being used to model a complex three-dimensional object from four components.

5.5 Procedural Description and Physical Simulations

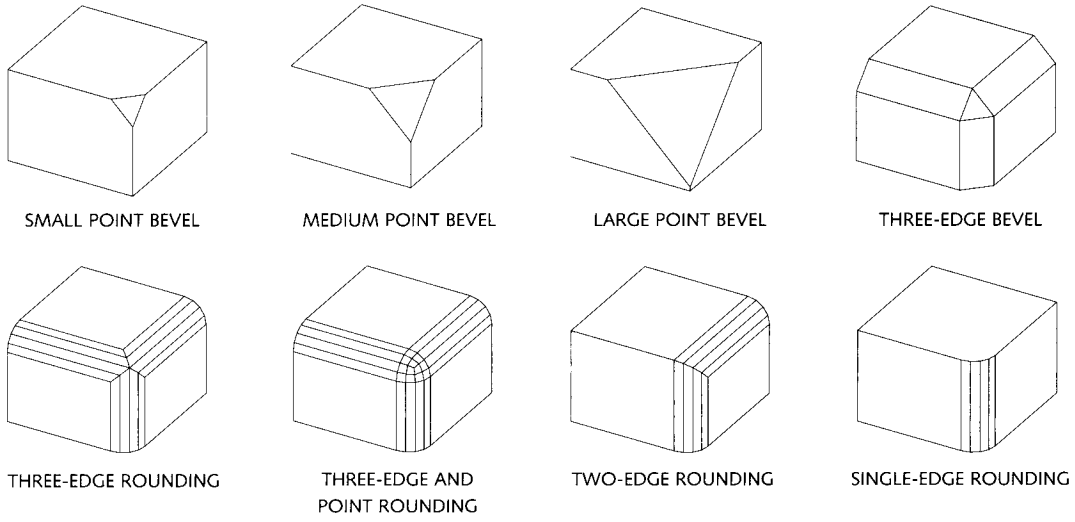
Procedural descriptions of three-dimensional models—especially those found in nature—are effective alternatives to the regular and sometimes rigid shapes obtained with geometric modeling systems. With procedural description objects are not modeled by sculpting their exterior shell. The modeling techniques of **procedural description** get their name from the fact that, with them, objects are modeled by simulating, for example, their natural growth process that is described in the form of procedures (Fig. 5.5.1). Fractal geometry and particle systems, for example, are two procedural description methods that create a modeling complexity that is difficult to obtain with geometric modeling. Both of these methods are well suited for the generation of natural-looking forms because they allow for randomness, recursion, and accidents of shape like those typical of natural shapes. A wide variety of plants can also be described with a combination of procedural description techniques. Finally, the shape of many natural phenomena that we do not think of as having a three-dimensional shape—waves, clouds and smoke, for example—can be built by simulating their physical behavior.

Fractal Geometry

Fractal geometry is especially effective for creating random and irregular shapes that resemble shapes found in nature (Fig. 5.5.2). This modeling technique was developed by Benoit Mandelbrot in the 1970s. It can be applied to existing three-dimensional meshes, or it can be used to generate entirely new models or parts of models. When applied to an existing model, **fractal procedures** work by dividing the polygons in the object recursively and randomly into many irregular shapes that resemble those found in nature. The amount of subdivision is usually expressed in the form of a factor or **level of recursion** (Fig. 5.5.3). Fractal geometry can also be used to create objects from scratch by using random seed values or iterations of algebraic formulas (Fig. 5.5.4).

Particle Systems

Modeling with **particle systems** is based on employing simple shapes, usually small spheres or points in three-dimensional space. These shapes, or particles, have **growth attributes**. When these attributes are simulated, the particles have specific behaviors that result in specific particle **trajectories**. As the particles grow over time their trajectory defines a certain shape that results in a three-dimensional model. The growth process of many of the attributes of the particles, including their height, width, branching angle, bending factor, number of branches, and color, can be controlled or randomized. Figure 12.4.2 shows a scene that was modeled by simulating feathers of birds or petals of flowers by creating particles that grow



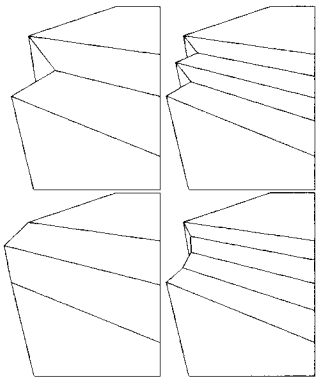
5.4.1 The first three shapes display a point bevel of different magnitudes. The second group of shapes includes a three-edge bevel, a three-edge rounding, and a three-edge and point rounding. The last two shapes are examples of a two-edge rounding and a single-edge rounding.

densely on the surfaces of three-dimensional objects. In this project, colored maps were applied to the feathers, the bird is animated with a skeleton deformation technique, and all the feathers grow at their positions as the body moves. (Read Chapter 12 for more examples and information about animating with particle systems.)

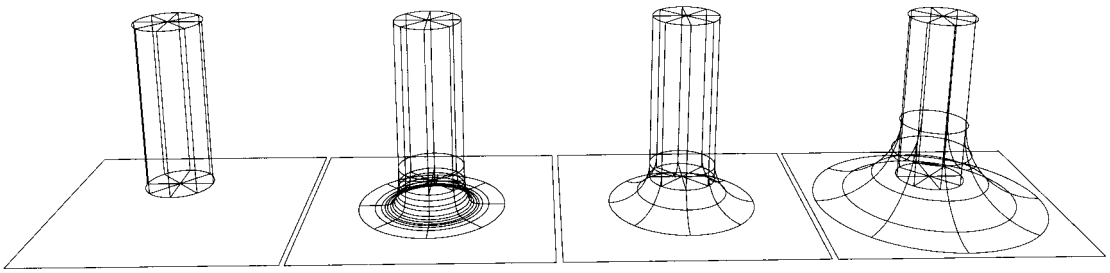
Procedural description, and particle systems in particular, are used to create a variety of natural materials and phenomena whose shape constantly changes throughout time. This includes, for example, snowstorms, clouds, flowing water, moving soil, and fire. When particles are used to recreate the motion of water, for example, they each represent a drop of water with attributes like density, cohesion, transparency, and refraction. Particles have a life span during which they behave a certain way, and then fade away or merge with other particles (Fig. 5.5.6). Of all the procedural modeling techniques, particle systems is the one that best recreates the dynamic shapes of natural elements. This technique produces large numbers of particles that do not have a specific shape and are usually spheres or dots. But the particles are grouped in shapes that change through time according to rules that define the behavior of elements such as water and fire. Particle systems are a very popular and powerful technique for the animation of effects both in animated and live action movies (Figs. 12.3.4, 12.4.1, and 13.7.1–2).

Modeling Plants

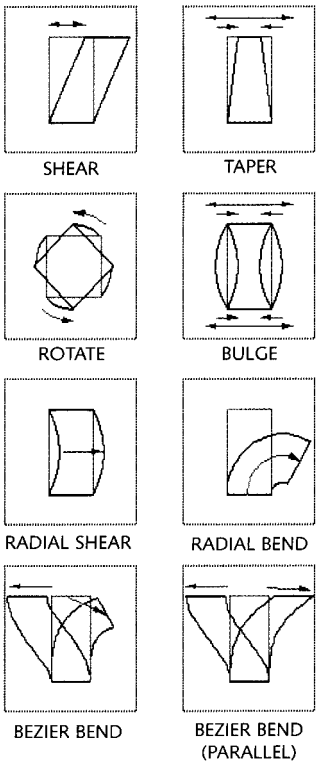
Three-dimensional models of plants and trees created with procedural techniques offer increased modeling control and more efficient modeling than most other techniques. This is due to the large number of elements and surface detail contained in a plant, as well as the complexity of shapes. Polygonal modeling techniques are often less than adequate for modeling realistic plants. Curved surfaces are



5.4.2 This figure shows the results of four different fillet styles applied to the corner and edge of a cube.



5.4.3 Blending a cylinder and a plane with different degrees of transition between the two surfaces.



5.4.4 Sequence of icons illustrating different techniques for controlling patch deformation. (form•Z icons, © 1991–1995 auto•des•sys, Inc.)

capable of modeling the shapes found in plants, but when done this way modeling plants quickly becomes time consuming.

Models of plants can be built by encoding their characteristics in a series of rules or procedures that are used as the basis for a growth simulation. This method can also be used to animate the growth process of plant models in a scene. There are several ways to generate models of plants with procedural techniques. These general techniques can be classified as either space-oriented or structure-oriented.

Space-oriented procedural techniques for modeling plants and animating their growth are based on the effect of the environment on them. **Structure-oriented procedural techniques** techniques, on the other hand, are based on the conditions that are internal to the plant, more specifically the growing process of the plant and the resulting structure that is characteristic of a plant species. Many of the procedural modeling techniques centered on the growth process of plants are often expressed in terms of the mathematical models formalized by biologist Aristide Lindenmayer during the late 1960s. These structure-oriented models describe the growth process of plants at the level of cellular interaction and are known as **L-systems**, short for Lindenmayer systems. L-systems are especially suited to represent structures that **branch in parallel**, just like a tree trunk splits into several branches at the same time (Fig. 5.5.7). Another characteristic of L-systems is that at each branching cycle, the branching procedures generate **successor modules** that replace the **predecessor modules**.

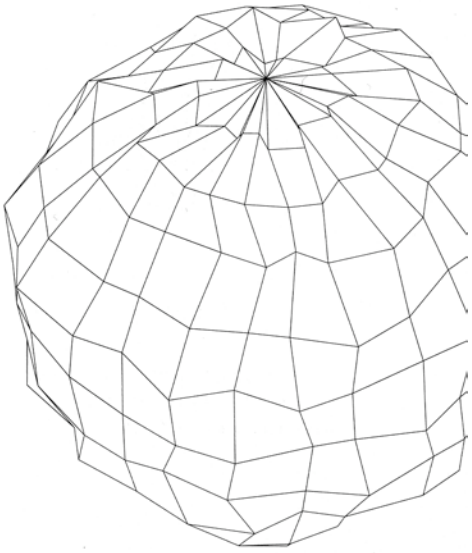
There are many variations of L-systems. Some are defined by parameters that represent exclusive conditions under which the L-system can bloom. Others are based on **stochastic** (or somewhat random) **values. Context-sensitive L-systems** are those whose performance is defined by the characteristic of the preceding module. The technique of **graftals** is an example of context-sensitive L-systems that allows the creation of complex images from small databases, a technique known as **database amplification**. Graftals are based on production rules and generation factors; they avoid random number generators, and can employ geometric or nongeometric objects—spheres, cylinders, or fuzzy objects with smooth edges.

Environmentally sensitive L-systems are defined by environmental characteristics, such as exposure to light and collision with

objects. Pruning is an example of three-dimensional models based on a simulation of a growing plant whose shape is determined by an environmental variable (Fig. 5.5.8).

Some of the main techniques for controlling the simulation of a structure-oriented plant system include lineage and the interaction between the components of the growth process. **Lineage** is the transfer of attributes from one level of the plant to another at the time of branching. The interaction of the components of the growth process includes, for example, taking into account the watering conditions or the availability of nutrients that define the way in which plants grow. The plants modeled with procedural description are more faithful to the real plant based on the number of components that are considered into the derivation of the plant shapes.

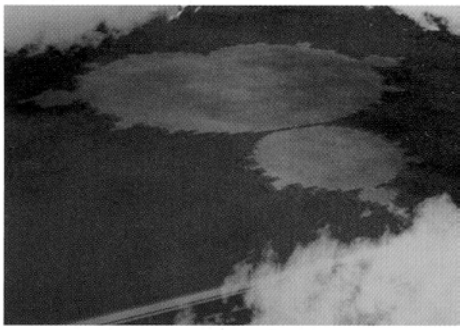
Techniques that seek to find efficient ways to process the large amount of data neccessary to model and render realistic plants are actively explored. The obvious challenges include dealing with billions of geometric primitives, distribute them throughout the terrain in a realistic way, and render all the subtle light and shadows effects that one observes in natural landscapes. Figure 5.5.9 illustrates the result of a technique that can populate the environment with a combination of ecosystem simulation and gardening techniques where the location of plants is done by hand. The figure also contains the statistics of instancing and geometric compression for the scene. This technique also makes extensive use of **model approximate instancing** where plants or groups of plants are replaced by instances of representative objects before the scene is rendered. In exploring complexity the researchers behind this technique found out that when one of the scenes approaches 100,000 plants each plant is visible on average only in ten pixels of a 1000×1000 pixel image.



5.4.5 Random distortion of a sphere.

Modeling with Physical Simulations

The easiest way to model natural materials whose shape is in constant change is not to sculpt them but to simulte them. Physical simulations are used extensively to model natural phenomena like rain, fire, smoke, and even wind; some of those techniques are covered in Chapter 12. Figure 5.5.10 shows a dialog box that allows users to animate a grid of waves with the basic variables of average wave height and slope, direction, frequency, and speed. Other global settings determine the complexity of the waves, the wave patterns, and underwater shots. Secondary motion in the simulation can be controlled at different levels of detail in the form of ripples, swells, and boat wakes (Fig. 11.2.8). The shading parameters in a physical simulation of water may include variables such as glitter, mapping of two-dimensional image and bump maps on the surface, reflection blurring, and index of refraction (Fig. 5.5.10). Depth and visibility fading is often used for underwater scenes. Figure 5.5.5 shows a simulation of pouring water in a glass, and Fig. 5.5.10 shows the result of a daytime ocean simulation.



5.5.1 This ray-traced scene, entitled *Crop Circle*, contains more than 100 million primitives. It was specified using a technique called procedural geometric instancing, which represents the field of wheat by hierarchically instancing a single wheat stalk. At instantiation, each stalk of wheat examines its position to determine if it should bend over or remain upright. (© 1994 John C. Hart, Washington State University.)

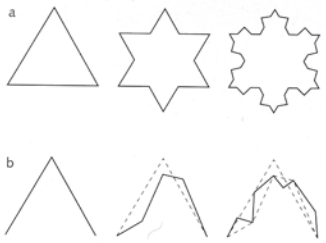
5.5.2 Details from *Seasons of Life* illustrate how fractal procedures can express the roughness and fragmentation of natural phenomena. (Courtesy of Midori Kitagawa De Leon.)

The shape of solid but flexible objects can also be controlled with physical simulation software tools. Figure 5.5.11 shows two pieces of fabric whose shape has been defined by a simulation of gravity and other forces that affect them. Natural-looking hair is usually simulated with multiple strands of particles that are guided by a few control hairs (Figs. 5.5.12 and 5.5.13).

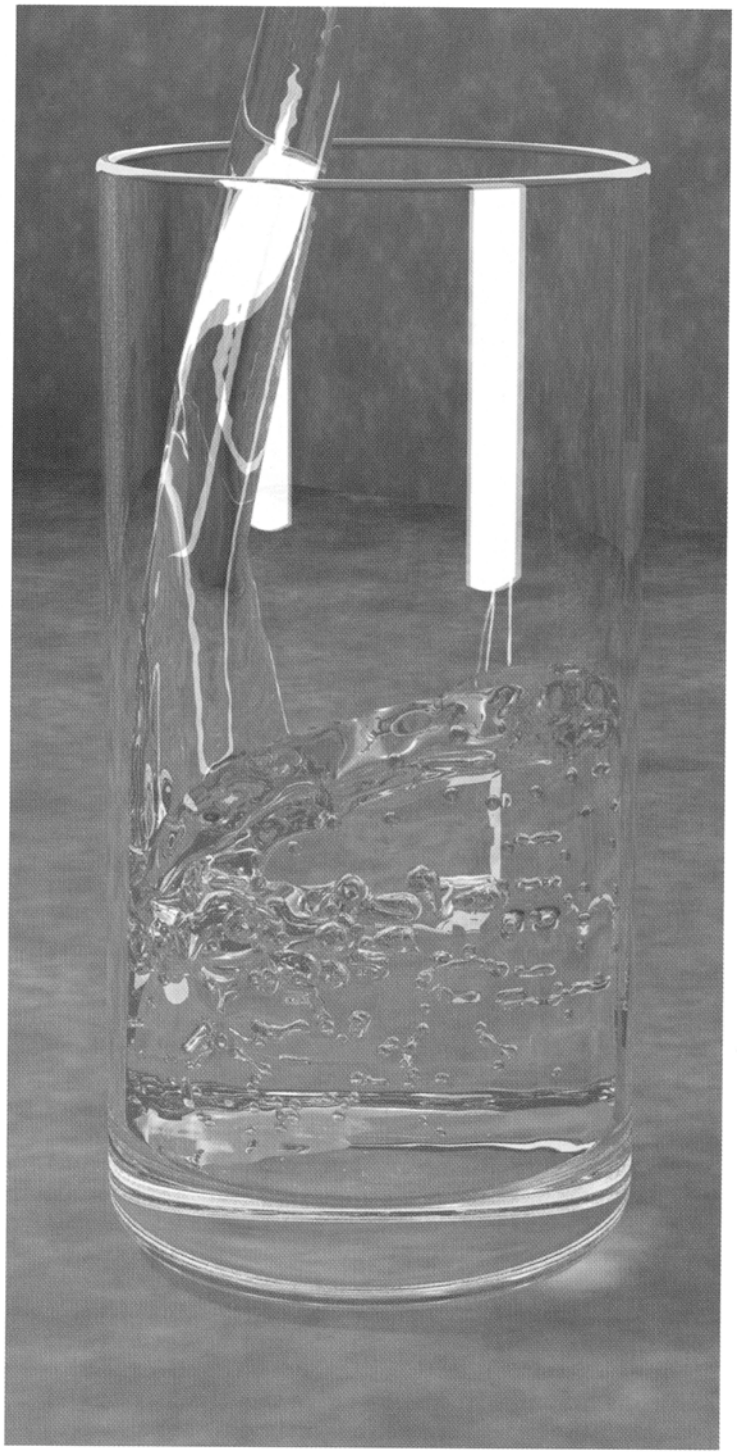
5.6 Photogrammetry and Image-Based Modeling

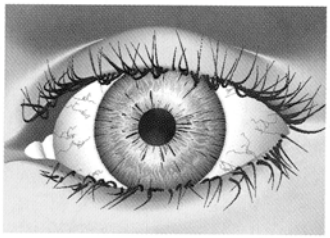
Photogrammetry techniques allow us to reconstruct three-dimensional enviromnents by extracting three-dimensional information from a series of photographs of an environment from known locations. The reconstruction process in many **photogrammetry** programs starts by indicating in the scanned photographs the corners or main edges of the major shapes. Photogrametry software reconstructs a simple version of the geometry by analyzing and comparing perspective lines and shading information in several related photographs of the same environment. Figures 6.8.1 and 6.8.2 provide a stunning example of image-based modeling.

Another technique that can be very effective in modeling large-scale three-dimensional environments is **laser scanning**. This technique often uses a linear laser scanner to collect three-dimensional point data which is then converted into a polygonal mesh. An interesting use of laser scanning is coupled with traditional film or video cameras to capture high-resolution image maps that can be applied to the geometry (Fig. 5.6.1–5.6.2).



5.5.3 A fractal recursive subdivision is applied to a regular polygon (top), and to the detail of a three-dimensional mesh (bottom).

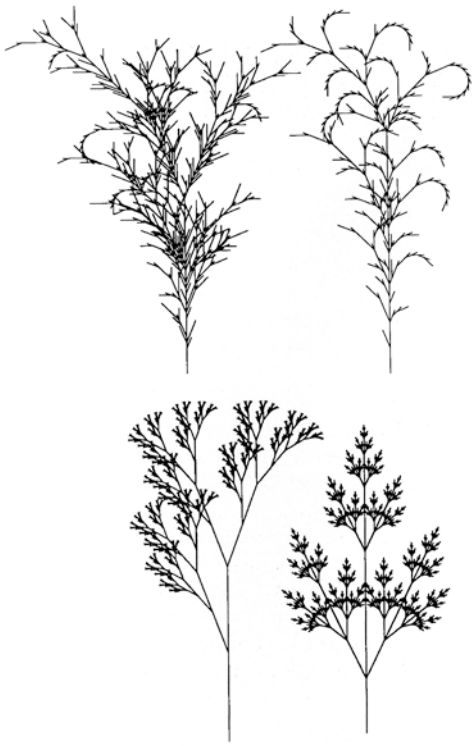




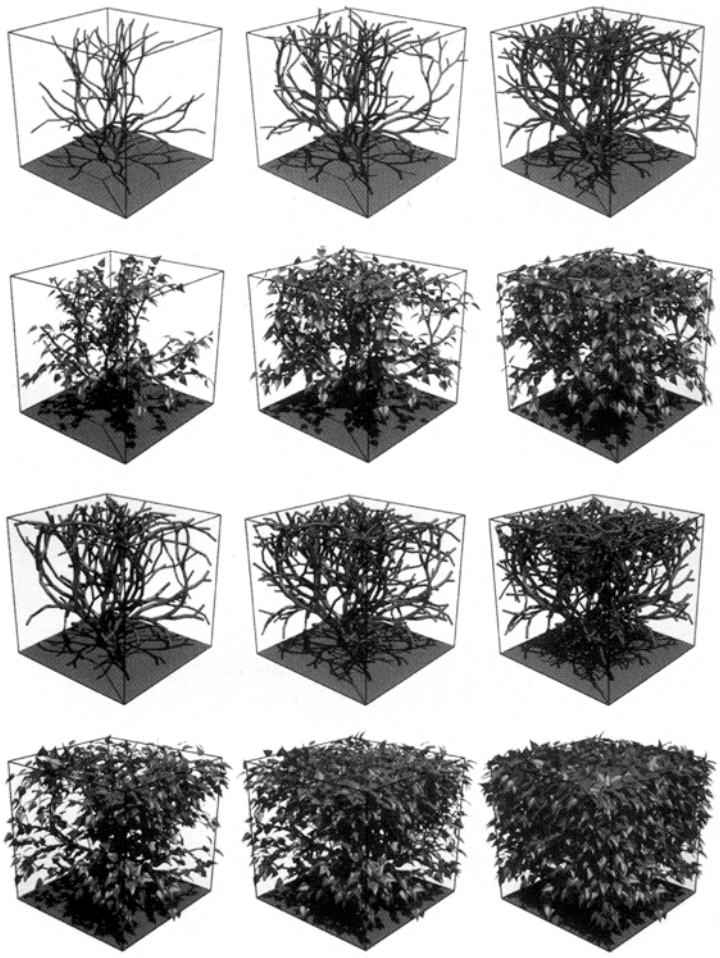
5.5.4 Rendering of an eye. (From "A Virtual Environment and Model of the Eye for Surgical Simulation," in the SIGGRAPH '94 Conference Proceedings. Courtesy of Mark Sagar, Department of Mechanical Engineering, University of Auckland, New Zealand.)

5.5.5 This dynamic simulation models the water surface and allows for interactions between objects and fluids. (© 2002 Next Limit SL.)

5.5.6 This still from the animation *Flow* shows a particle system combined with a water mesh simulation system. (Rendering and animation by Gavin Miller, modeling by Ned Greene. © Apple Computer, Inc.)



5.5.7 These plants clearly show the transfer of attributes, called lineage, from one level of the plant to another at the time of branching. (From "The Algorithmic Beauty of Plants," by P. Prusinkiewicz and A. Lindenmayer, Springer Verlag, New York, 1990. © 1994 Przemyslaw Prusinkiewicz.)



5.5.8 (top right) Simulated response of a tree to progressive pruning. The branches grow more densely near the edges of the cube used to clip the model. The leaves are defined as Bézier surfaces. Shown in red is the vigor of the reiterated branches on the appearance of the pruned tree. (© 1994 P. Prusinkiewicz, M. James, and R. Mech.)

5.7 Animation Rigging and Hierarchical Structures

Three-dimensional objects can be grouped together in a limitless number of ways in order to create structures that define the ways in which these models behave when animated, how they relate to each another, and how they are rendered. Groups of three-dimensional objects are usually called **hierarchical structures**, because within these structured groups some objects are always more dominant than others. (See Chapter 11 for additional information on levels of precedence, joints, and degrees of freedom.) Hierarchical structures are the most basic layer of an animation rig.

Objects within hierarchical structures have defined levels of importance and the objects within this hierarchy inherit attributes from the dominant objects. The object or objects at the top of the hierarchy are called **parents**, and the objects below are called **children** and grandchildren: Children inherit their parents' attributes.

Tree Statistics

Plant	*Objects*	*Instances*
Apple trees	1	4
Reeds	140	140
Dandelions	10	55
Grass tufts	15	2577
Stinging nettles	10	430
Yellow flowers	10	2751

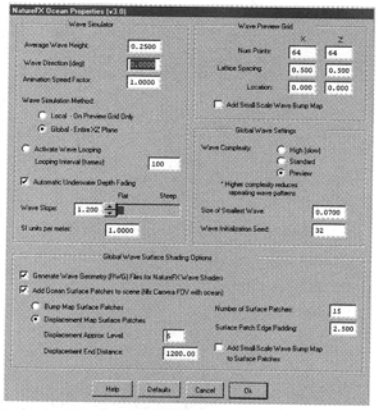
5.5.9 The images representing each species of plant in the top image were rendered separately and later composited. The table lists the number of prototype objects and their instances for each species. The compute time on a 195 MHz R-10,000 8-processor SGI Onyx with 768 MB of RAM was 75 minutes. The poplar tree in the top image is 16KB when saved as a procedural model but 6.7 MB in a standard text geometry file format. (Copyright Bernd Lintermann. Software by Oliver Deussen and Bernd Lintermann.)

Hierarchical structures can also be visualized as an inverted tree structure where the highest level of importance in the structure corresponds to the trunk of the tree. The branches that come out of the trunk are the next level of hierarchy, branches that come out of the main branches are the next level, and so on until we get to the leaves, which are in the last level of the hierarchical structure.

In most cases, there is just one set of hierarchy diagrams per scene, and these diagrams control all the modeling, rendering, and animation information about the objects. Some programs, however, provide several sets of hierarchy diagrams, and therefore hierarchy configurations, so that **modeling links** can be independent and different from the **rendering links** and the **animation links**. Modeling links, for example, make sure that the shape of all the children throughout the hierarchy changes when the shape of the child's parent is modified. Rendering links transmit the rendering attributes of the parent down the structure. Animation links automatically update the spatial position of the children when the parent is transformed.

The relations between objects in hierarchical structure are often complex and can be better visualized by a schematic representation

5.5.10a Surface of the sea modeled by simulating the motion of the water, and on the next page a NatureFX software dialog box used to control the the simulation of water surfaces. (Courtesy of Areté Entertainment, Inc.)



5.5.10b NatureFX software dialog box used to control the the simulation of water surfaces. (Courtesy of Areté Entertainment, Inc.)

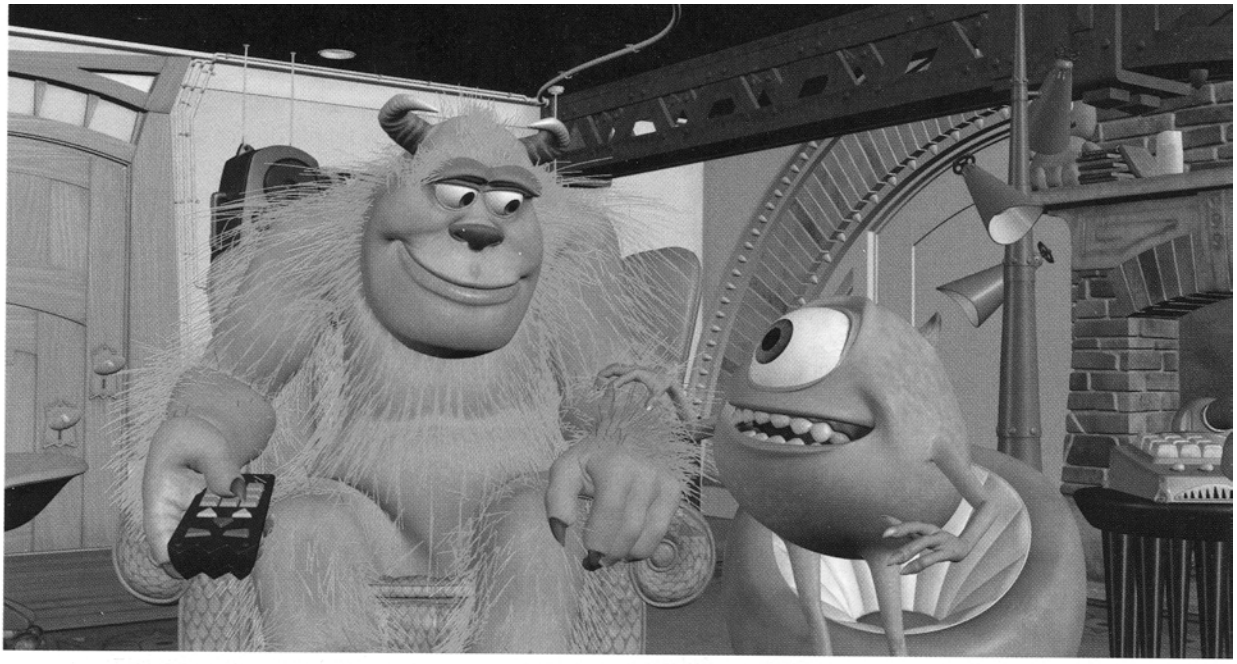
5.5.11 (right) Fabrix simulation software was used to shape the purple fabric and the gold backdrop. Clusters were positioned and scaled to gather a 20 × 40 Nurbs mesh into the hands of the high-resolution human modeled by Zygote. The simulation also tested for self-collisions. (*Fab-girl* by Shawna Olwen. © 1999 Reflection Fabrix Inc.)

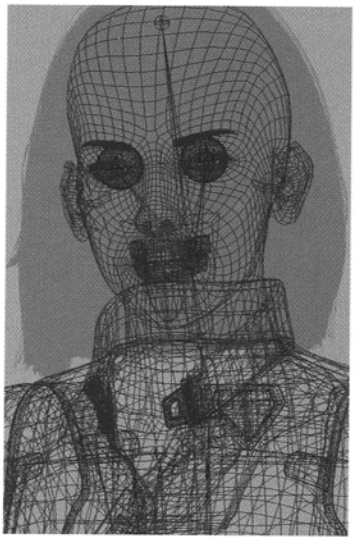
in the form of a line diagram. These diagrams are often built with boxes representing items in the structure and lines representing their place in their hierarchy and their relations with other items.

As shown in Fig. 5.7.1 there are many ways to group objects or nodes in a hierarchical structure. The best choice of hierarchy is usually one that takes into account the rendering and animation requirements of the scene. (See Chapters 11 and 12 for more details on how hierarchical structures affect the object's rendering and transformation attributes.) A node in the hierarchy that does not contain an object but holds several children together is called a **null parent**, or null for short. A null parent is used, for example, when two or more objects are grouped at the same level in the hierarchy. Nulls are often represented with italics or as empty boxes in the structural diagrams. Figures 5.7.1 and 5.7.2 illustrate the use of a null cell that can be used to keep the four legs and the wheels in one group so that they can be manipulated independently of the ring.

The specific steps required to establish hierarchical relationships between objects vary from software to software. But the basic concept has two variations. Clicking on the three-dimensional objects themselves in any one of the camera views, or clicking on the boxes representing the objects in the diagram representing the links between them. Some programs require users first to click on the object that is to be the parent and then on the children. With other programs the object that is to represent the children must be clicked before the parent. Hierarchical structures are more significant to the

5.5.12 (opposite page, top) Sully from *Monsters, Inc.* used 10,000 control hairs to define the placement, orientation, and animation of between 2.8 and 3.2 million blue hair strands. (© Disney Enterprises, Inc./Pixar Animation Studios.)

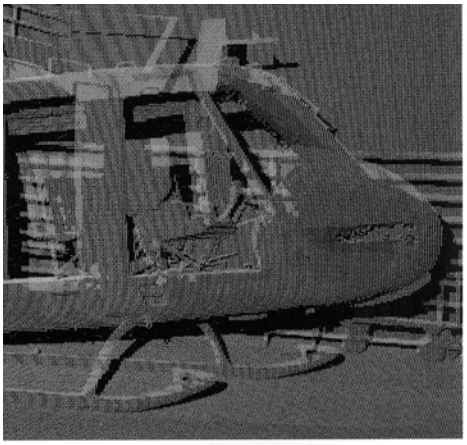




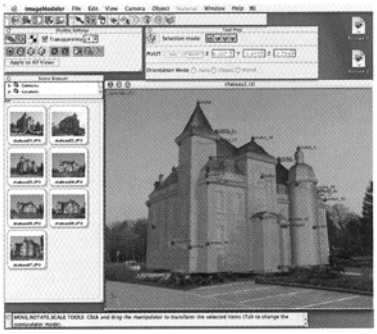
5.5.13 (bottom left and right) Between 30,000 and 60,000 strands were used to create Aki's hair, which accounted for 25% of the total rendering in *Final Fantasy: The Spirits Within*. The hair appears in 611 shots, almost half of the total. Aki's body and head model has up to 100,000 subdivided polygons. (© 2001 FFFP.)

stages of rendering and animation, they are usually created during the modeling process in the form of animation rigs.

An animation **rig** is a hierarchical control structure that is custom-designed to animate characters. The animation rig is the motion engine of the character; all the joints and all the motion logic of the character are contained in the rig. Very complex rigs require significant computation and when manipulated by animators they take longer to update, so it is best to include in only the essential features. In most large-scale projects control rigs are usually designed and assembled by specialized technical directors called riggers. For a small



5.6.1 Wireframe geometry and shaded version of a helicopter and the environment surrounding it as modeled with the *Panascan*™ laser scanning process. (Courtesy of Panavision.)



5.6.2 Dialog box from Image Modeler, a software that allows three-dimensional reconstructions from photographs. (© Realviz®.)

production with limited resources a simple rigging and skin binding structure can be more efficient and minimize rendering times.

Animation rigs can be visualized in different ways: as a hypergraph or hierarchy chart (Fig. 5.7.2b), as a rig skeleton showing bones and joints (Fig. 5.7.3), or as the set of controls that animators use to manipulate the rig system. These animation controls are represented with icons and color-coded (Fig. 12.5.6) so that animators recognize faster what controls do or what parts they control. During the animation process the rig skeleton is usually turned off as animators pose the character with a proxy, low resolution, version of the skin and with the animation handles (Figs. 5.7.3 and 5.75). Sophisticated rigs are capable of notifying the animator through visual cues when a limit is reached or when the animator is trying to do something that the character is not supposed to do, like trying to bend a knee backwards or pass a foot through the ground.

Animation controls in a rig can consist of a point, a curve or a complex surface. For ease of use animation controls often include multiple controls, a foot control, for example, might include foot pivoting, toe lifting, wiggling, and twisting. The legs and feet are crucial in animating a character since they are so important in walking. Most leg-feet rigs include joints for hip, knee, foot, ball of the foot and toe end (Fig. 12.1.8). Facial animation rigs also include multiple bones as shown in Figure 12.5.5.

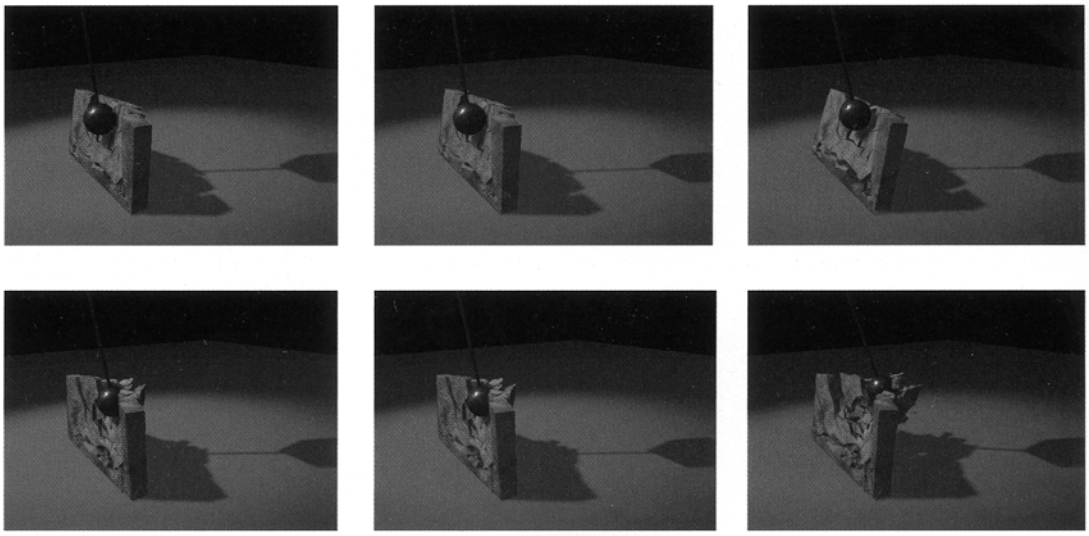
One of the basic issues in setting up a rig is defining the order in which rotations take place. The **order of rotations**, as explained in Chapter 3, is important because the sequencing of axis rotations creates different results (Figs. 3.4.1–3.4.2). Software programs usually require an order of rotations to be specified in a rig, with the Y axis commonly being first followed by X or Z in worlds where Y defines the vertical axis. This way characters can turn around in any direction around Y and move sideways or lean without reaching **gimbal lock**, which is a point at which the model runs out of rotations.

Skin Deformers

The animation rig controls and deforms the skin geometry of the character in a variety of ways: with simple joint attachments, using the actual bone geometry, or a complete muscle system. Binding skin points to one or several joints is the simplest technique for skin deformation. In order to increase the complexity of their translations these skin points can be weighted, or influenced by the rotations of multiple joints.

The skeleton geometry itself (not the abstract hierarchical skeleton) can also be used as a skin deformer. In this case the bone geometry is parented to the IK joints and used as skin weights. Using bone geometry as a skin deformer serves the dual purpose of being a useful visual aid for positioning skin, joints and muscles in the proper place. The bone geometry is only used to extract weight values and "baking" them into the skin, after which point the skeleton

NEW SECTION…….



12.3.7 The animation of these adobe walls struck by wrecking balls is simulated with linear elastic fracture mechanics. The simulation determines where the cracks should start and in which direction they should propagate after analyzing the stress tensors that are computed over a finite element model. As the animation develops, the software remeshes the geometry to create the dynamic fractures. The wrecking ball in the bottom row has 50 times the mass of the ball on the top row. (Images courtesy of James F. O'Brien, W. Wooten, and J. Hodgins. © 1999, Georgia Institute of Technology.)

But a simpler and more economical method for collision detection can be implemented by identifying the **obstacles** that the moving object is likely to encounter along the **collision path**. Figure 12.3.7 shows colliding objects that break into smaller pieces based on a finite element analysis simulation.

12.4 Procedural Animation

Procedural or **rule-based motion** techniques animate the elements in the scene based on a set of procedures and rules that control motion. Rule-based animation has a wide range of applications that include the animation of natural phenomena, flying birds, growing plants, fantastic life forms, and humans dancing or gesturing. (See Chapter 5 for more information on modeling plants with procedural techniques.)

Particle Systems

One of the most popular forms of procedural animation is exemplified by the animation of particle systems. Animation with **particle systems** recreates the motion of particles that follow some generally defined motion. In the majority of computer animation programs, the particles themselves do not have a specific shape, but they can be used to control other objects or attributes. When particles are used to recreate the light of fireworks, for example, they represent a point of light with a variety of attributes such as intensity, flickering, and tail-tracking values (Fig. 5.5.6 and 12.4.1–12.4.2).

Particle systems are used to represent dynamic objects that have irregular and complex shapes, each with its own behavior. Particles have a life span during which they are created, behave a certain way,

age, and die. Particles can also be used to control the motion of three-dimensional models—such as snow, water, or even a flock of birds—and to animate the growth process of plants by encoding their characteristics in a series of rules that can be used as the basis for a simulation (these methods are described in Chapter 5).

Flock Animation

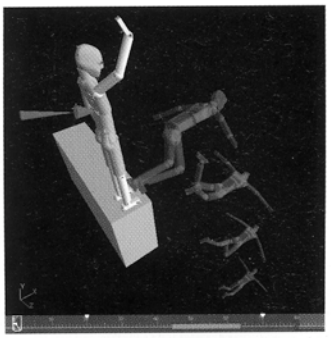
There are many different strategies to generate flock animation. In most cases, the behavior of the birds in the flock is contained in a series of rules that constitute the computer model that simulates the flock animation. These rules control all the variables involved in the behavior of the flock. These rules include, for example, whether the flock has multiple leaders or a single leader and, if so, in which patterns does the rest of the flock follow the leader. Some of the basic variables provided by most computer software to control the motion of flock animation include the way in which the members of the flock move towards a target, how they avoid obstacles, and how they relate to other members in the flock as the flight conditions change throughout time.

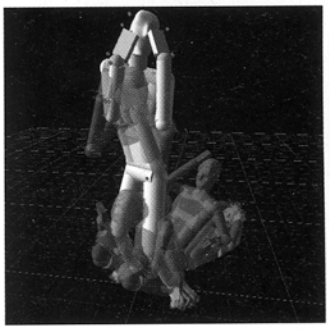
Animating a flock of birds with rule-based techniques is a more practical alternative than using keyframe animation. Flocks can be simulated with particle systems so that each particle in the system represents a bird. Each bird in the flock moves according to the laws behind the physical simulation, its own perception of the environment formalized by the rules of the system, and by a series of parameters defined by the animator. The overall motion of a flock can be represented as the result of the behavior of each individual bird and the interactions between them. A common strategy to recreate the flying behavior of each bird is based on rules that simulate some of its perception and the action of flying. Once the model is expressed in terms of rules then several birds can be simulated and allowed to interact with each other.

A significant difference between particle systems and flock animation based on particle systems is that in flock animation the particles are replaced by three-dimensional models; they have orientation, and they also have more complex types of behavior.

The behavior of flocks is determined by the internal conditions of each bird in the flock, and also by the external conditions that affect the flight of each individual bird and the flock as a whole. Birds in the flock present many forms of behavior and goals. Common behaviors and goals of flocks include, for example, avoiding collisions with other birds in the flock or objects in the environment, matching the speed of nearby birds, and staying together. Each of these behaviors requires a specific acceleration and direction.

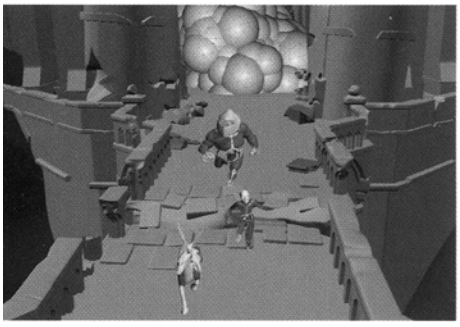
When the goals of dozens of birds are in play it is necessary to arbitrate all the individual requests. The flocking model used to create Figure 1.4.1 employs a variety of techniques to arbitrate indepen-

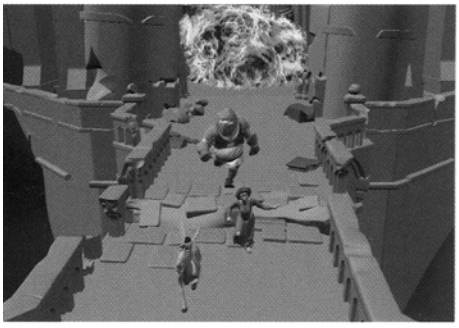




12.3.8 Complex human behaviors built in real time with endorphin physical simulation software. Animations can be imported into a variety of animation systems. Character stands waiting for his first interaction. A force represented by arrow is applied and secondary behaviors can also be specified, arms extended for example (top). After a quick punch to the head the character (bottom) recoils backwards, clutches at his back, impacts the ground as small secondary movements ripple through the body as the arm impacts on the ground, and finally relaxes into a "dead" behavior. (© 2003 Natural Motion.)

12.4.1 (next page) Simulation of a fireball for an effects escape sequence, also showing previsualization with proxy models and simplified rendering (Shrek™ and © 2001 DreamWorks L.L.C.)





dent behaviors. These techniques are based on a prioritization of all the component behaviors and their acceleration requests. Depending on the situation different requests get a higher priority. For example, maintaining all the birds in the flock together can receive a low priority if the flock is about to collide with a large obstacle.

Computer animations of flocks can also be created with a combination of particle animation and keyframe animation. This is especially useful in cases when the computer animation software does not provide a full-fledged rule-based system. The flock animation in Fig. 1.4.1 was controlled mostly with simple parameters that are adjusted at the keyframes, but the relation between flock members cannot be specified in the form of rules. In that case the process started by creating a set number of particles that will represent the members of the flock as they move from a **source mesh** for particles to a **destination mesh**. The precise path of the motion can be controlled by the magnitude and direction of a simulated force of gravity, and also by the way in which the particles select the points (or vertices) on the source mesh to leave from and the points on the destination mesh to arrive to. The distribution of the particles on both meshes can be easily controlled by the shape of the meshes and by whether the source and destination meshes have the same shape and number of vertices. The distribution of particles can also be controlled with numerical values that randomize the mesh or by concentrating most flock members on

a small group of vertices. The behavior of the flock as it moves from one point to another can be controlled in this example by numerical values that specify the amount of back-and-forth change, or jittering, in each of the geometric transformations, and the shape of the models being controlled by the particles. The shape of the three-dimensional model that is controlled by the moving particles can be specified in the form of one or several key shapes. Other approaches to flock animation are illustrated in Figures 12.4.3 and 12.6.1.

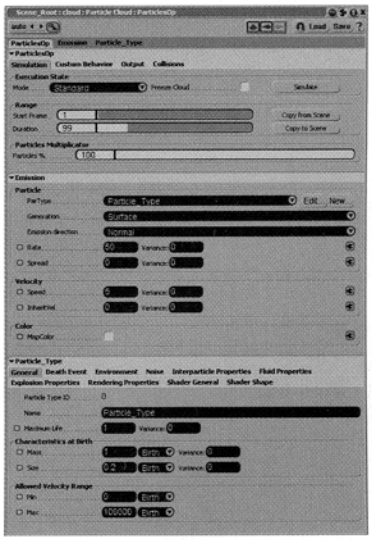
Goal-Oriented Animation

Some computer animation systems are capable of automatically choreographing the motion of an animated character based on a specific goal that has to be achieved. The animated characters in goal-oriented systems range from a simple robot arm or a fantastic creature to a more complex human-looking character. The goal for the character can be as simple as turning its head towards the light, or as complex as grabbing an object with the left hand, passing it to the right hand, and running out of the room while avoiding all the obstacles along the way. **Goal-oriented** computer animation is also often called **intention-based** or **automated** animation. Goal-oriented animation has its roots in the fields of robotics and expert systems, where computer systems are designed so that they can be as autonomous as possible, including the ability to plan different strategies to achieve a goal, evaluate the results, and continue to develop the strategies that were succesful while avoiding the strategies that lead to failure.

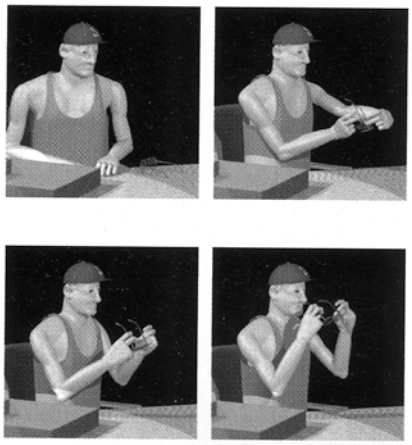
Goal-oriented animation techniques still belong by and large in research laboratories. Nevertheless their usefulness is slowly turning them into commercial products. The most important component of a goal-oriented animation system is the set of rules and procedures that allows a character to analyze and evaluate its environment and to determine the best way to achieve a goal, usually by reacting with motion, gestures, and manipulations of objects in the environment. Many goal-oriented animation systems also include an inverse kinematics module to deal with the position of jointed figures, and/or a motion dynamics module that deals with basic issues, including weights, forces, and collision detection.

Goal-based computer animation systems include the **codified procedures** that are necessary to analyze a goal, break it into tasks, evaluate the environment, predict and try to avoid potential obstacles, recover from mistakes, develop new strategies as a result of those experiences, and ultimately achieve the goal. Most existing goal-oriented computer animation systems specialize in a specific type of goal or a specific type of motion; otherwise their tasks would be too complex to implement. (When the complexity of goal-oriented animation approaches the complexity of human motion it is often more practical to record human motion itself directly on video or film.) Some goal-oriented systems, for example, specialize in simulating

12.4.2 This detail of a still frame from *The Holy Bird* is based on the traditional dances of Okinawa as interpreted by artist Ryoichiro Debuchi. (Assistant designer: Atsuko Katakura. Hardware: IBM RISC/6000. Software: DDS, Feather, and Wavefront. Production: Digital Studio, Inc. Courtesy of Ryoichiro Debuchi.)



12.4.3 Parameters to control a flock animation. (Dialog box courtesy of Softimage Co. All rights reserved.)



12.4.4 These still frames from a goal-oriented animation (top left to bottom right) illustrate how the software can generate the motion paths necessary for a character to manipulate a pair of glasses with both hands. The motion of the arms is achieved with both inverse kinematics and a sensorimotor model based on neurophysiological studies. (Images from "Planning Motions with Intentions" by Yoshihito Koga, et al., SIGGRAPH '94 Conference Proceedings. Courtesy of Yoshihito Koga, Stanford University.)

human gaits and other forms of multilegged locomotion. Others can animate hands grasping and manipulating objects, or even body gestures and facial expressions.

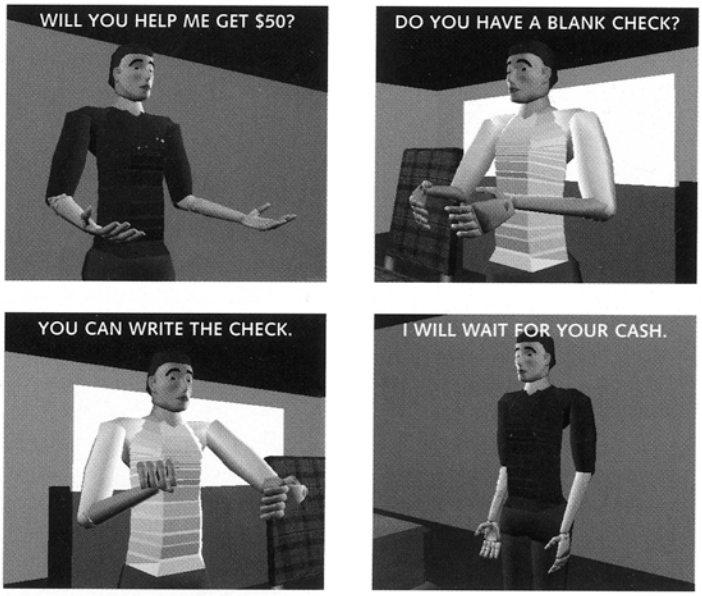
One of the main tasks of goal-oriented animation system consists of determining what sequences and paths of motions are necessary to achieve a certain goal. Finding an optimal motion path that will allow the goal to be completed involves testing for collision detection, angles of motion, and grasp ability of limbs. Establishing a **sequence of motions** involves determining how many steps are necessary to complete a motion and what is the optimal order of execution. Simple goals that involve motion usually translate into simple motion paths and simple motion sequences. But as the goal increases in complexity so do the paths and sequences of motions necessary to complete the goal.

One of the biggest challenges of goal-oriented computer animation is to deal effectively with complex sequences of motions, both in terms of being able to complete the tasks and achieve the goal, and also in terms of producing natural motion when human figures are animated. For this reason, most are based on some sort of **motion planner**. The systems that animate characters that grasp and manipulate objects use a **manipulation planner**. In addition, goal-oriented systems include kinematics or dynamics techniques for calculating motion. The manipulation planner used to create the sequence of images in Figure 12.4.4, for example, defines paths in terms of transit paths and transfer paths. In this case, the goal for the animated figure consisted of reaching for the glasses and wearing them. The task of the animator is limited to selecting the object that has to be moved and the location where the object has to be repositioned. The motion planner of this goal-based animation system determines that the character has to use both hands in order to complete the action. Not too many individuals can grab a pair of lenses and put them on with just one hand. The **transit paths** define the motions of the character without the objects being manipulated, for example, getting the arm to a position from which it can reach the object. The **transfer paths** define arm motions that also move the object. Transfer tasks are generated by analyzing and planning the motion of the object from its initial position through the completion of the goal. During the calculation of the path the manipulation planner identifies all the possible ways of grasping the object and the configurations of the object requiring a grasp or regrasp.

The majority of motion planners have a simplified set of rules that specify how motion takes place in general, and which motions are allowed in particular. The animation system illustrated in Figure 12.4.4, for example, allows only the arms of the animated character to touch the objects in the environment that are to be grasped. Objects in the environment that are obstacles can only be touched for the purpose of achieving static stability. In the interest of efficiency, most goal-oriented systems limit the number of possible motions and

grasps, or types of static and dynamic obstacles that are considered, or the number of possible solutions to situations when collisions occur.

A few goal-oriented animation systems attempt to automate the animation of human figures based on instructions given in plain English—or other human natural languages—as opposed to special-purpose animation languages. One example of such an approach is illustrated in Fiure 12.4.5. This animation system is able to automatically generate and animate conversations between human-looking figures. The conversations include motions such as facial expressions and arm gestures. The facial expressions are associated with motions of the head, eyes, and lips. The arm gestures include coordinated motions of the arms, wrists, and hands. The rules underlying this system are based on the relation between verbal and nonverbal communication. The combined meaning of speech, body language, and facial expressions can result in animated characters that are consistent, believable, and somewhat autonomous.

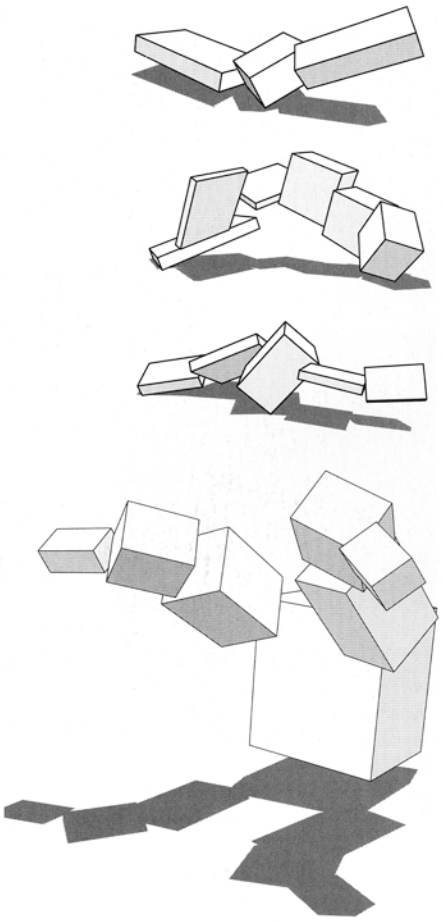


12.4.5 These four gestures were automatically generated by a goal-based animation system that creates a sequence of simple actions in response to a statement. First, a gesture requesting help accompanies the verbal request for money. Then an iconic gesture representing a rectangular check is generated from the words *blank check*. Later, a gesture representing writing on a piece of paper is generated from the mention of the action of writing a check. Finally, a motion of the hands up and down emphasizes the notion of waiting. (From "Animated Conversation: Rule-Based Generation of Facial Expression..." by J. Cassell et al., SIGGRAPH '94 Conference Proceedings. Courtesy of Dr. Justine Cassell.)

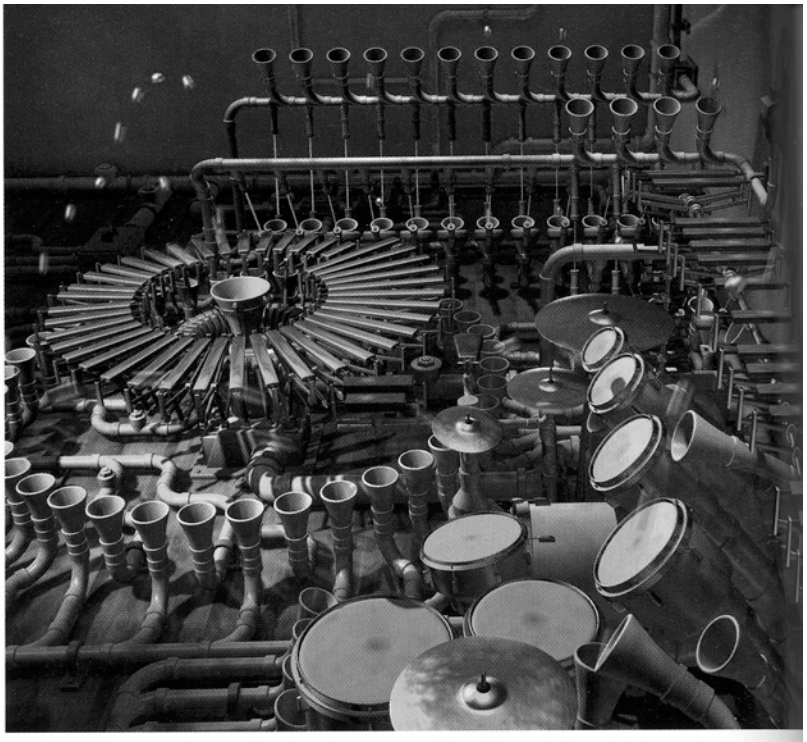
One of the most unique features of this rule-based animation system is that it is capable of accepting the text of the dialog within the context of a database that contains facts, goals, and beliefs of the animated characters about the world and about one another. The text of the dialog is preprocessed so that the linguistic and semantic aspects of a conversation can be interpreted by computer programs in charge of the speech synthesis, semantic analysis, and generation of gestures and facial expressions.

The rules of the program used to create the images in Figure 12.4.5 are based on a topology, or classification, of facial expressions and hand gestures that assign a meaning to a selection of gestures and expressions. It also establishes a semiotic relationship and a temporal synchronization between speech, gestures, and expressions. Much of the sequencing and coordination of actions is created with a computer model that activates transitions between actions based on conditions being met, or on rules of probability.

Within the computer animation program that was used to generate Figure 12.4.5 the **gesture motion** is specified with information about the location, type, timing, and handshape of individual gestures. The hand, wrist, and arm positions can be controlled independently. An interesting feature of this system allows users to control



12.4.6 These simulated creatures were optimized over 100 generations for locomotion on land. Their motion techniques include shuffling, lumping, crawling, rocking, and hopping. (Images from "Evolving Virtual Creatures" by Karl Sims. Courtesy of Karl Sims, Thinking Machines Corporation.)



12.4.7 (above right) The virtual musical instruments in *Pipe Dream* were animated with procedural data generated in the MIDI format. (© 2002 ANIMUSIC.)

the expressiveness of the gesturing of a character by modifying the size of its **gesture space**, which is usually dependent on the virtual age group and culture of the animated characters. The facial expressions are generated both automatically when based on the intonation and phoneme—or spoken sounds—and by hand when they add meaning to the spoken discourse. Gazing, one action involved in facial expression, is controlled automatically based on the purpose of looking, for example, whether to look away to gain concentration or look at the other character to reinforce a point in the conversation.

In addition to simulating human-looking characters, it is also possible to simulate and animate artificial types of life with goal-oriented computer animation techniques. The three-dimensional creatures illustrated in Figures 12.4.6 and on page 370 (Key Terms) were created with a computer program that uses genetic algorithms to define the shape of the creatures and the processes by which their motion is controlled. These creatures are the result of a genetic evolution simulation that seeks to optimize a specific goal: their locomotion on water or land. This is achieved by running survival tests throughout 100 generations of approximately 300 members each. The survival tests consisted of a physical simulation where the creatures are tested for fitness based mostly on their speed and ability to control their speed and direction of motion. The fittest creatures within a generation are selected by the program for survival and reproduction.

The creatures in this example are able to sense contact with their own selves and with the environment, and can avoid obstacles. These creatures have a morphology that evolves from a simple con-

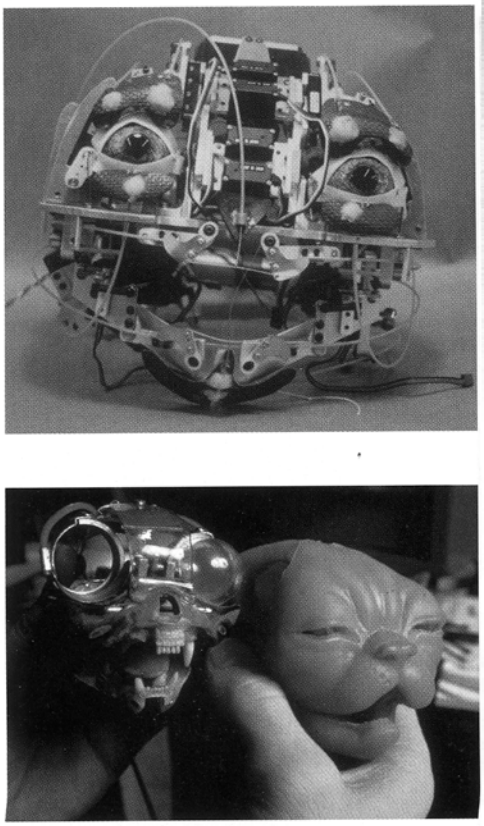
trol system that senses the environment, evaluates the situation, and reacts to it. The evolutionary process starts with a random generation of sets of nodes (body, limbs, and head) and the connections between them. A clear set of rules and procedures governs all the stages of this simulated world, including the initial assembly of the creatures, their behavior, their fitness evaluation, and their reproduction and optimization.

MIDI Output

Using music as the source for motion is another effective method for creating procedurally-driven animation. Unlike the digital recording of a specific sound or music, the **Musical Instrument Digital Interface** protocol, generally known as **MIDI**, contains information about how to play or interpret specific musical notes. MIDI contains information about the actual musical notes and keys, the length, the pressure, the instrument, and other parameters associated with musical interpretation. MIDI is commonly used to feed music synthetizers that execute the MIDI instructions in real time to create sound. Figure 12.4.7 shows an example of MIDI data used to procedurally drive the animation. In this case the MIDI data is processed with a proprietary software to generating sequences of motion for the given music. This software uses standard MIDI parameters like note, volume, pitch-bend, modulation, and sustain-pedal, to output position, rotation, scale, and light intensity information. The most interesting motions are obtained by using combinations of several algorithms to map the MIDI date to three-dimensional data. In the case of MIDImotion software, for example, the process is straightforward. Low notes can be mapped to large, slower-moving objects, while higher notes can be mapped to smaller, faster-moving objects. Drumstick motion can be derived by using a technique reminiscent of robotic motion planning. Additional motion such as ball trajectory and object impact can be calculated using basic physics. If the music is changed, the motion is automatically regenerated. Finally the generated parameter channels are used by commercial animation software via plug-in technology.

12.5 Facial Animation

The controls for facial animation have become very sophisticated because of an increased need for realistic animation. There are many techniques for animating **facial expressions**, including morphing between libraries of key poses, blend shapes, motion capture, motion dynamics simulations, and goal-oriented techniques. Facial animation is often generated with hybrid combinations of these techniques, and the relationship between the inner structure and the outer surface remains a key component in the success of the final result (Fig. 12.5.1). Facial animation is usually applied to the character after the body primary motion has been blocked out (Figs. 2.4.14 and 2.4.16).



12.5.1 These two images illustrate the importance of animating the facial skin through the use of simulated bones, joints, and ligaments. On top, the mechanism of an animatronic cat's head built for a feature film appears alongside the skin that will eventually cover it. Below is a complex animatronic underskull for a feature film fantasy character. (Courtesy of Jim Henson's Creature Shop.)