(Previous page) Saxophone modeled with myriad techniques. (Courtesy of Toru Kosaka, STUDIO EggMan.)

3.1.1 This surreal scene illustrates the concept of *gluttony* with an unusual assortment of objects and creatures created with multiple modeling techniques. (© Jim Ludtke.)

CHAPTER 3

Basic Modeling Concepts

Summary

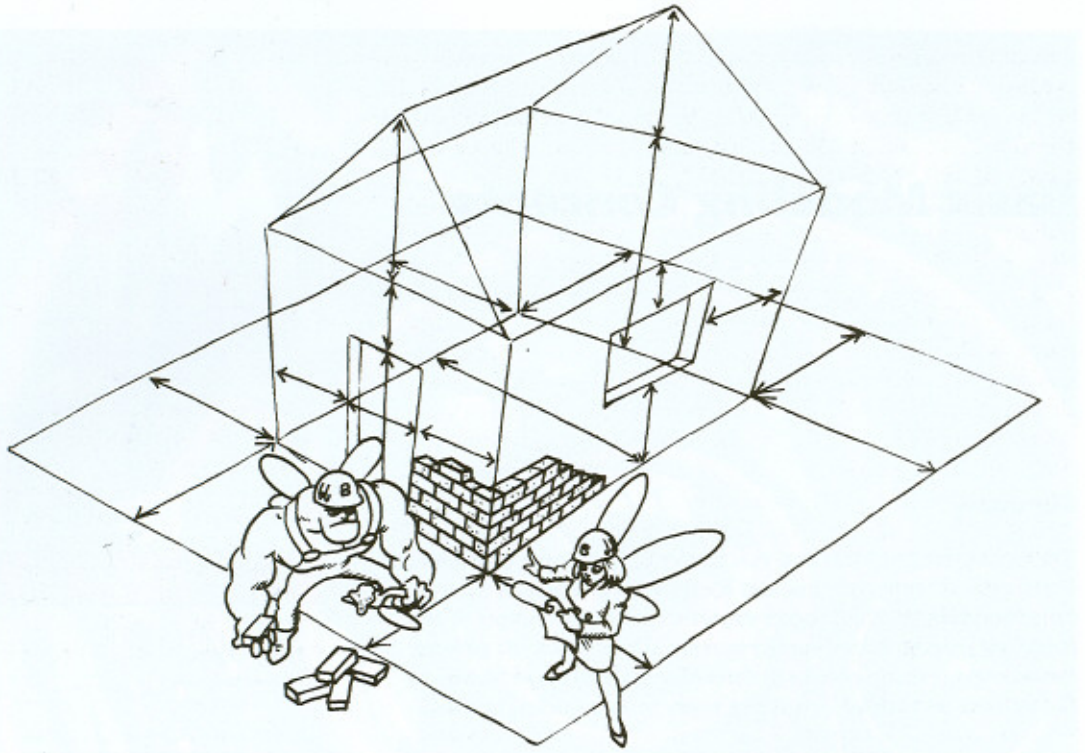
THE SPATIAL DESCRIPTION AND PLACEMENT of imaginary three-dimensional objects, environments, and scenes with a computer system is called **modeling**. This chapter explores the basic concepts of the modeling process, including the numerical description of objects, moving and resizing objects in three-dimensional space, common file formats, and advice on getting ready for a modeling session.

QUICK INDEX

3.1 Space, Objects, and Structures

We live in a three-dimensional world. We move among other people, climb mountains, run on the beach, and admire the landscape around us. We go in and out of buildings, walk up and down the stairs, drive through bridges, and grab utensils for writing, cooking, and combing ourselves. Our daily life happens in three-dimensional environments and is full of three-dimensional objects and characters (Fig. 3.1.1). We see and feel three-dimensionality all the time. But unless we are involved in an activity or profession that is related to building things, like silverware or furniture, buildings or bridges, we rarely concern ourselves with how our three-dimensional reality was put together and what techniques were used for building it.

When it comes to the modeling of our reality we usually take a lot of things for granted. But if we want to model three-dimensional scenes with a computer program, we have to familiarize ourselves with the large selections of computer software tools that can be used for modeling objects and environments. In three-dimensional computer modeling it is quite common to use a combination of tools for building just one object. For example, think of the difference between a chair that was built using just two tools—a manual saw and a hammer—and a chair that was built with six tools—a thick saw, a thin saw, a lathe, a curved chisel, a hammer, and a sanding tool. It is obvious that while the first chair could have an interesting design the variety of shapes would be limited. The second chair would have richer



3.1.2 Building a simple rectangular room requires taking measurements that include the orientation of the walls, the location of the doorway, the windows, and the distance between the walls.

and more refined modeling. The simple computer-based modeling tools are described in this chapter, and most of the advanced modeling tools are covered in Chapter 4. Now let's step back and talk about some general issues involved in modeling in three dimensions.

Many of the basic conventions used in three-dimensional modeling software describing three-dimensional scenes are based on traditional conventions used in various disciplines. For example, to convey space in a clear and concise way, architects use conventions related to measuring, composition, and sequence. Even the design of a simple rectangular room requires measuring many times so that all the components of the room end up where they were planned to be (Fig. 3.1.2). Furthermore, to interpret accurately an architect's drawing and build it, masons need to take measurements. Over the ages masons and architects have developed conventions so that they can be precise and clear about measuring **spaces**, building **objects**, and arranging them in **structures**.

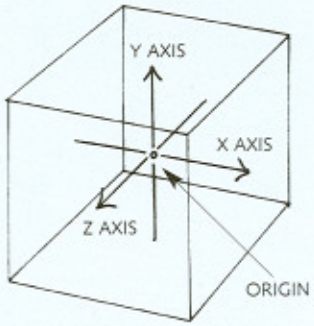
We use similar conventions to describe the dimension, placement, and sequence of objects and environments in a three-dimensional space simulated with a computer program. A beginning builder will soon find out that there are many different methods available for measuring space, and at first this variety can be confusing. But experienced builders usually find this richness of methods very empowering and have the option of using one or another depending on the requirements of the project. Let's start our defini-

tion of three-dimensional space with the boundaries that define our **workspace** or **scene**. The simplest way to do this is to imagine that we are working inside a large cube. We can think of this cube as our world or environment. Objects that exist within the cube are visible, those that fall outside are invisible (Fig. 7.2.1).

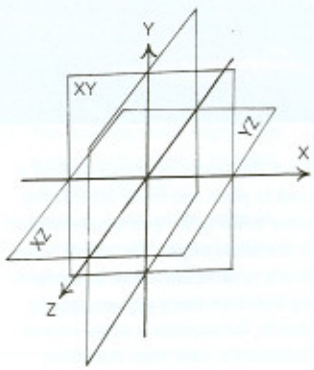
The main point of reference in this world is called the **world origin**. The origin is usually located in the center of the space, but it can also be placed or repositioned elsewhere depending on the modeling needs and strategies (Fig. 3.1.3). For example, if we were building a model of the solar system, it would make sense to have the world origin where the sun is, in the center, because all the other objects in the system are placed around the sun, and can be easily described in terms of the sun. If we were building an underwater scene that included both fish under the water and boats above the water, we might want to position the origin at the point where air and water meet. In the case of a three-dimensional model of an airport, the world origin could be placed at ground level, matching the position of the control tower. All three-dimensional spaces have three basic **dimensions**: width, height, and depth. A common method for representing these dimensions in a three-dimensional space is by using arrows or **axes** (Fig. 3.1.3). It is common to label the **axis** representing the width of a **three-dimensional space** with the letter X, the height axis with the letter Y, and the depth axis with the letter Z. The point in space where these three axes intersect, or cross each other, is the world origin.

The **rectangular coordinate system** can be used to define specific locations and accurately position the points of objects in three-dimensional space. René Descartes, an eighteenth-century French philosopher and mathematician, formalized the idea of using three axes labeled X, Y, and Z to represent the dimensions in three-dimensional space. The coordinate system he devised is commonly referred to as the **Cartesian** (or rectangular) **coordinate system**. Each axis in the system can be divided into many units of measurement. In principle these units are abstract values that can represent different units of measurement and scales of dimension. On each axis the values to one side of the origin are positive, and the values on the other side are negative. As shown in Fig. 3.1.3, the positive direction of each axis in a right-handed coordinate system is represented with an arrowhead.

There are many ways of representing the direction of an axis and, consequently, the directions in which values on that axis are positive or negative. Usually though, in what is called a **right-handed coordinate system**, the values on the X axis become larger to the right of the origin, the values on the Y axis increase as they move above the origin, and the values on the Z axis grow as they get close to us. In a **left-handed coordinate system** the values on the Z axis decrease as they get closer to us. There are several variations of the directionality of the rectangular coordinate system, but most three-dimensional modeling programs use the right-handed coordi-



3.1.3 The origin is a point of reference usually located in the center of the three-dimensional space. It can also be located elsewhere in three-dimensional space. A three-dimensional space has width, height, and depth dimensions each represented by the three axes in the Cartesian coordinate system. The numerical values in the figure correspond to those in a right-handed coordinate system.



3.1.4 The three planes, or views, that can be defined with the XY, XZ, and YZ pairs of axes are useful for building models from different points of view.

3.1.5 The world coordinate system is used to place and move objects, the entire building for example, in relation to the world origin. The objects' coordinate systems are useful for performing transformations only on specific objects, for example a single column. (Sainsbury's, view from Blackfriars Road South. © Hayes Davidson.)

nate system to describe the virtual world.

The three axes in the rectangular coordinate system can be paired with each other in three different ways so that each pair of axes defines a plane or a view. The XY axes define the **front plane**, the XZ axes define the **top plane**, and the YZ axes define the **side plane** (Fig. 3.1.4). There are other coordinate systems in addition to the popular rectangular coordinate system. The **spherical** or **azimuthal coordinate system** is also widely used because it provides a simple way for placing objects in a three-dimensional world in terms of their distance to the object, their angle around the point of interest, and their altitude angle above the point of interest. The spherical system is especially useful for placing and moving cameras and light sources in a three-dimensional scene (Fig. 3.4.8).

Any **world** or **global coordinate system** is useful for placing or moving objects in the world or in relation to each other. World coordinates are absolute values that are relative to the origin of the world. These coordinates do not depend on any specific object in the world and are applied to all objects in the world indistinctly. (Global transformations, as we shall read later in this chapter, are easily expressed in terms of the world coordinate system.) In addition to the world coordinate system, however, each object in the world can have its own **object** or **local coordinate system** (Figures 3.1.4 and 3.1.5). Object coordinate systems are values relative to the origin of the object, which is sometimes placed in the center of the object. For all practical purposes object coordinate systems are used only to specify positions, orientations, or transformations of the object in question. (Local transformations are almost always expressed in the coordinate system of the object.)

3.2 Building with Numbers

Throughout time we have developed a sophisticated vocabulary for describing with the spoken or written word the shape of three-dimensional objects and their relative positions to one another. We can use that vocabulary for communicating with others about three-dimensional objects and their positions in space. But, even though

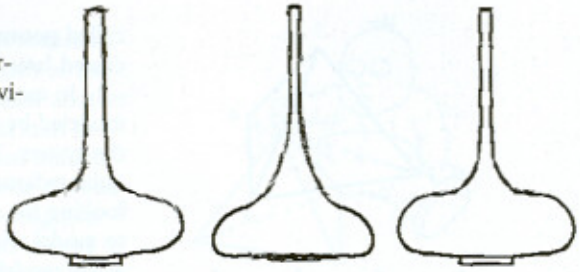
verbal descriptions of objects can be very concise, they lack precision. Verbal descriptions of objects can be interpreted in a variety of different ways, not only for the obvious issues of dimension but also for the subtle issues of proportion and shape. As you read the following description of a flower vase try visualizing it in your mind or by drawing it with pencil and paper.

If you have some experience with modelmaking or pottery you were probably able to follow the description of the shapes in the vase and the relation between them, and your flower vase might look similar to the results shown in Fig. 3.2.1. But if you have little experience with three-dimensional models, your sketch may be quite different, or you may have been unable to finish reading all of the description. Maybe you lost interest because you found it difficult to visualize all the shapes and the ways in which they were attached to one another.

Most individuals and today's computer systems are incapable of recreating in detail verbal descriptions of complex three-dimensional shapes. Computer modeling in three dimensions requires very precise and unequivocal descriptions of shape. The method of choice for precise and unequivocal descriptions of shape and their location in space consists of using numbers. With numerical description we can specify the position of an object in space and the details of its shape: height, width, depth, diameter, curvature, and number of sides. Figure 3.2.2 illustrates most of the numbers required for describing a fairly simple three-dimensional shape.

Much of the success in modeling three-dimensional objects and environments with a computer system lies in understanding how a particular computer system describes a shape with numbers. The exact numbers can mean different things to different software programs. For example, some systems give great attention to the decimal, or floating point, numbers (i.e., 5.379) that describe the subtle shape of a small curve, while other computer systems may ignore those numbers completely and recreate the curve based on a whole numerical value (i.e., 5). We must also keep in mind that some of the numbers that describe a shape might be of little value to us if we were building the object with traditional materials, such as wood, but the same numbers often provide the computer system with crucial information for building the object with computer modeling techniques. For example, the order in which we number points in a shape can yield very different results.

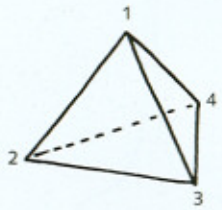
The essence of all software-based three-dimensional modeling techniques consists of creating a **data file**, or a list of numbers, that defines models in a way that can be understood by the computer program. Whether we create a simple cube or a collection of computer shapes representing a human hand, the numbers describing the object are kept in a file so that the program can load them into memory, display them, modify them, display them again, save them, and so on. The files that contain the data describing the object are

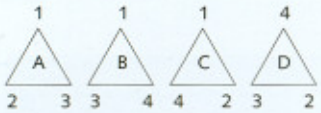


Verbal Description of a Vase

- The vase has a very long neck and a short, round base.
- The neck is about five times the height of the base, the width of the base is about twice its own height.
- The cylindrical neck grows out of the base slowly.
- At the point where the neck touches the base it has a width that equals the height of the base.
- As it moves upward the neck gets narrow, and as it passes the first fifth of its height the neck reaches a thin, delicate width that remains constant until the end of the neck.
- A small section of the oval shape that constitutes the base of the vase is sliced off so that the bottom of the base is flat.
- The resulting sharp edge at the bottom of the base is rounded off just a little bit.
- Halfway between the edge of the base and its center, a thin slice of a short cylinder is attached to the base.

3.2.1 Different interpretations of the verbal description of a flower vase.





Geometry Format #1

Point	X	Y	Z
1	0	0	0
2	-1	-2	1
3	1	-2	1
4	0	-2	-1

Face	Point	Point	Point
A	1	2	3
B	1	3	4
C	1	4	2
D	3	4	2

Geometry Format #2

Face	XYZ	XYZ	XYZ
1	0 0 0	-1 -2 1	1 -2 1
2	0 0 0	1 -2 1	0 -2 -1
3	0 0 0	0 -2 -1	-1 -2 1
4	1 -2 1	0 -2 -1	-1 -2 1

3.2.2 The difference between two geometry formats for the same object is quite evident here. These two listings were generated with two different types of modeling software.

called **geometry files**. Examples of all geometry file formats are discussed later in this chapter.

In most modeling systems, three-dimensional objects can be modeled by typing the numbers that describe the object directly into the system. This method of **direct numerical description** can be quite tedious and time-consuming. We rarely use it unless we are looking for an extremely specific shape or a detail that can be hard to model with regular modeling tools. Even when we use the interactive modeling tools provided by the software it is still possible to peek at the numerical information that the software uses to describe and manipulate three-dimensional shapes. Most systems allow us to get this numerical information in varying degrees of detail (Fig. 3.2.2). Both the shape of an object and its position in three-dimensional space are expressed in terms of numerical values (Fig. 4.6.1). From the computer's point of view, numerical values are easy to manipulate and easy to repeat. This facilitates the building of three-dimensional objects with modeling software, as well as the duplication of objects using the **cut-copy-paste** techniques used by most of today's programs.

3.3 Points, Lines, and Surfaces

Now that you have learned how to locate points in three-dimensional space and how to create and edit lists of numbers that represent XYZ spaces you are ready to start thinking about building a simple model. The three-dimensional object illustrated in Fig. 3.3.1 is defined by four points, six lines, six edges, and four planes.

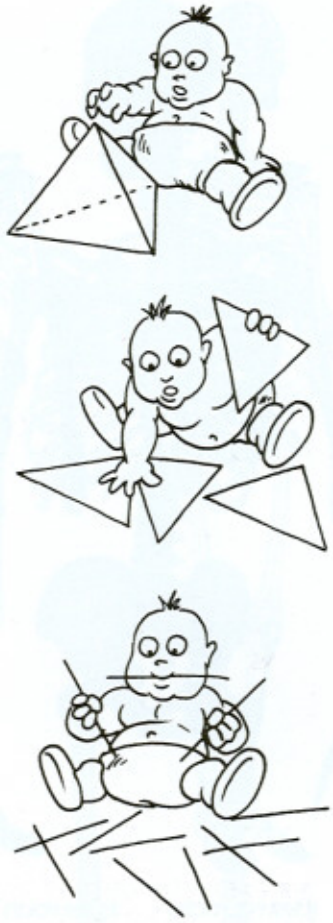
Points, lines, and surfaces are among the basic elements that can be used to build three-dimensional objects. A **point** can be easily defined by its XYZ location. A **line** can be defined by the XYZ location of its two endpoints. An **edge** is defined by two adjacent surfaces. A planar **surface** can be defined by the position of its bounding lines. An object is usually composed of several points, lines, and surfaces. A three-dimensional object can be described to software as a list of numbers. This list is usually generated automatically by the computer program, but it can also be generated directly by the user. As stated earlier, it is not necessary in most cases to input all of these numbers by hand. In fact, we do not need to be aware that all this number-shuffling is taking place. But once in a while you will encounter modeling situations when it will be paramount that you understand the meaning and proper structure of these numbers. It is for those occasions when the information in this section will come in handy.

Simple objects like the one pictured in Figures 3.2.2 and 3.3.1, and even much more complex objects, can be easily described or edited in most modeling software providing their point XYZ positions and the connectivity lists to the three-dimensional program. Whether this is done by typing their numerical values directly on the keyboard or by transferring the XYZ data collected by a three-dimensional scanner, a simple methodology can be followed. First, label all the

points and all planes in your sketch or printout—if one is available (Fig. 3.2.2, top). Then write down the XYZ position of each of the points in list form (Fig. 3.2.2, Format 1 points). Finally, make another list that includes each of the planes and all the points that must be connected to define them. It is important that all the points in each plane are connected in the same direction—either clockwise or counterclockwise (Fig. 3.2.2, Format 1 faces). Some computer programs require that you connect the points clockwise, while others require counter-clockwise, but usually all programs require that the order be consistent throughout the entire object.

The planar surfaces that define most three-dimensional objects are also called **facets**—as those in a cut diamond—or polygons. The word polygon has its roots in the Greek word *polygónon*, which means "with many angles." Polygons are closed planes bounded by straight lines. Polygons can be regular or irregular. Many of the three-dimensional shapes created with three-dimensional computer software are made of polygons. Simple geometric shapes may be defined with dozens of polygons. Objects like a teacup that require a fair amount of detail may also require hundreds of polygons to model that detail. Complex objects, such as the detailed model of a human, may require thousands of polygons (Fig. 3.3.2). The modeling of natural phenomena, such as a forest or simulation of the explosion of a supernova star, may require millions of polygons.

Sometimes we can define objects with curves instead of straight lines, and curved surfaces instead of flat polygonal surfaces. At first building objects with curved surfaces can be more demanding than using polygonal surfaces because curved surfaces are more complex. Read Chapters 4 and 5 for additional information on curves and curved surfaces.



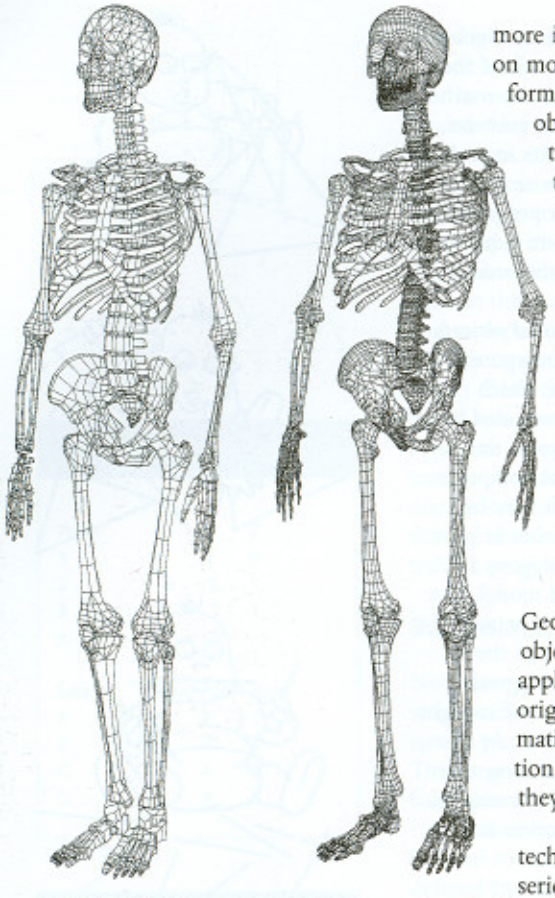
3.3.1 Three-dimensional objects are defined by points, lines, and planes. This simple pyramid has a total of four vertices and four sides.

3.4 Moving Things Around

Once we have built some objects we can move them around in three-dimensional space and create a composition or a scene. Sometimes it becomes necessary to move some of an object's components—a group of points, for example—before the modeling is completed.

The functions used for modifying the shape of objects, their size and proportions as well as their position in space, are called **geometric transformations**. The name of these simple but powerful tools obviously comes from the fact that they can be used to transform—to change, to move, to modify—the geometry of objects. In effect, these **mathematical operations** can modify the numerical information that describes the objects that we build in the environment and even the environment itself. The most widely used geometric transformations are translation, rotation, scaling, and perspective projection.

Geometric transformations can also be applied to the camera that "looks" at the scenes we model and arrange, and also to the lights that reveal our creations to the camera. See Chapter 7 for



8,979 POLYGONS 35,305 POLYS

3.3.2 The first skeleton model is built with 8,979 polygons, the second one is built with 35,305, and the third one (opposite page) with 141,788 polygons. Notice the higher density of polygons in areas of the surface that have more modeling detail.
(© Viewpoint Datalabs, used with permission.)

more information on camera motion, and Chapter 8 for more on moving lights around. In general, when specifying transformations to be applied to a single object or a group of objects it is important to specify the type of transformation, the axis or axes in which the transformation is to take place, the point around which the rotation or series of rotations will occur (whether the transformation is local or global), and the order in which transformations are to take place in a sequence of several of them. Geometric transformations are usually calculated by most programs with the aid of a **transformation matrix**. This 4×4 matrix is used to calculate the new XYZ values after a transformation is applied to all the points of a three-dimensional element. A few programs allow users to manipulate XYZ values directly in the matrix in addition to, or instead of, using more user-friendly tools.

Global or Local Transformations

Geometric transformations can be performed on single objects or on entire environments. Transformations that are applied to the objects using the environment's axes and/or origin are called **global transformations**. When transformations are applied to a single object—or a limited selection of objects—using the object's own axes and origin, they are called **local transformations**.

In general, software programs offer two basic techniques for specifying whether a transformation—or a series of transformations—is global or local. It can be done by selecting the objects directly with the mouse or by typing the names of the objects on the keyboard.

We can start to define a local transformation by selecting or activating one or several objects—but not all of them. In such instances, the transformation will be applied to the active objects only. In general, when one object is selected as the recipient of a local transformation, the object's center and axis are usually used as the centers of rotation and scaling, and the axis of translation, rotation, and scaling. (The centers of rotation or scaling are usually located in the center of the object unless specified otherwise.) Some programs, however, offer the option to apply local transformations to an object based on the environment's center and/or axis instead of the object's center and/or axis. The results can be quite different (Fig. 3.4.1). For example, an object scaled along its axis after being rotated retains its shape, while an object scaled along the world's axes will not retain its shape. Check the manuals of the software you use to find out how local and global transformations are implemented. Having a clear understanding of this is crucial to the correct operation of your software.

When performing global transformations, or local transformations that occur along or around the global axis, the order in which transformations are applied to an object or a series of objects can affect the final result. For this reason rotation and scaling sequences should be planned carefully, although translation sequences can be applied in any order. **Concatenated transformations** is the name sometimes given to a series of global transformations applied in sequence. Figure 3.4.2 illustrates the different results obtained by applying the same global transformations to a trio of objects but in a different sequence each time.

In general, if all objects in a scene are active, the transformation is global and applied to all objects. Most software programs apply transformations to all the active objects. When performing a global rotation or global scaling, the center of the environment usually doubles as the center of rotation and scaling for all the objects unless specified otherwise.

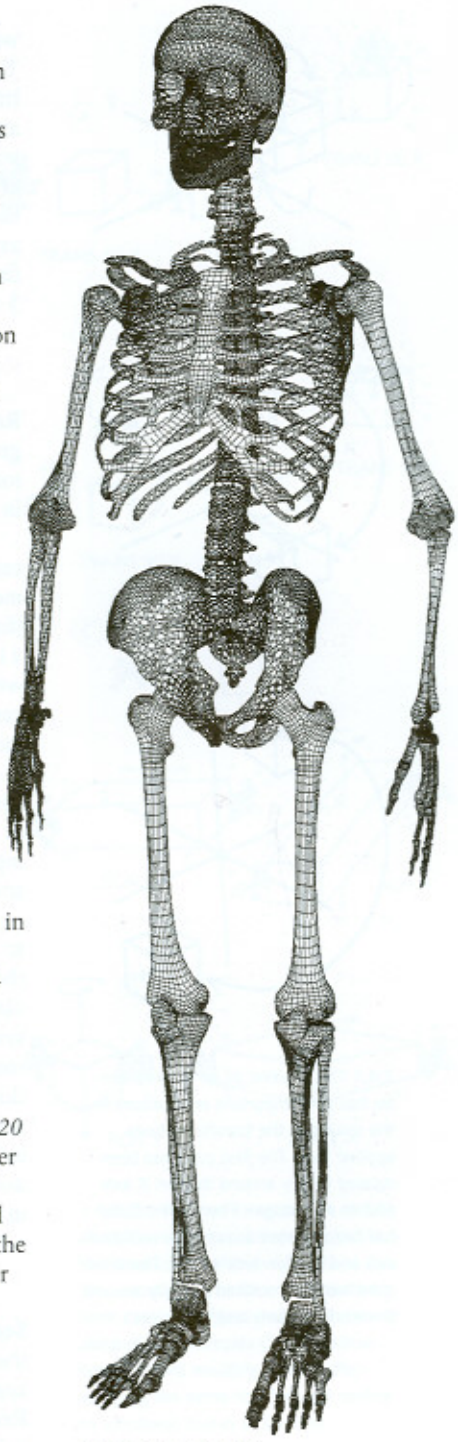
Absolute or Relative Values

When working with most interactive modeling programs, applying transformations to one or several objects is as easy as selecting them and dragging them to a new location in three-dimensional space. It is quite common to use the mouse and the mouse button to control the position, orientation, and size of the models in the environment. However, it is sometimes necessary to type specific values on the keyboard for controlling the exact position, orientation, and size of models. When typing values becomes a necessity one must keep in mind that all transformations can be specified as **absolute values** or as **relative values**.

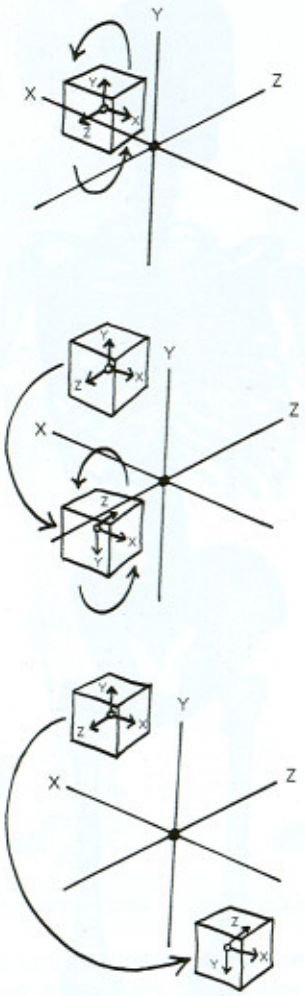
Absolute values, or numbers, always refer to an exact position in space where the object must be relocated regardless of where the object was located in space before the transformation. Relative values, as their name indicates, are numerical values that express the number of units that must be added or subtracted to the current position of the object. Relative values are relative to an existing absolute position. For example, if we have a sphere with a center located at XYZ coordinates 30 30 30, the command *trans sphere 0 20 0* (if the numbers were relative) would reposition the sphere's center at XYZ coordinates 30 50 30 because 20 units would have been *added* to the sphere's position. However, if the numbers being used were absolute numbers, the sphere's center would be relocated to the 0 20 0 XYZ position, regardless of the fact that the sphere's center was previously located at 30 30 30.

Translation

Translation is the simplest of all geometric transformations. This operation is used to move an object or group of objects in a linear



141,788 POLYGONS



3.4.1 The position of an object's center has very important implications in the results of the transformations applied to it. The first cube has been rotated locally around its own X axis and its own origin. The second cube has been rotated around the world's X axis and its own local origin. The third cube has been rotated globally around the world's X axis and origin.

way to a new location in three-dimensional space (Fig. 3.4.3). Translation is the simplest and easiest to control of all geometric transformations. Translation can occur along one axis or along several axes at the same time. The order in which several global and local translations are applied to one object does not affect the final position of the object. For example, an object that is translated 5 units along the X axis, then 10 units on the Y axis, and finally -7 units on the Z axis would end up in the same positions as an object that is translated first 10 units along the Y axis, then -7 units on the Z axis, and finally 5 units on the X axis.

Rotation

Rotation is the geometric transformation used to move an element or group of elements around a specific center and axis. The amount of rotation is usually specified in terms of an angle of rotation (measured in degrees) and a direction of rotation (Fig. 3.4.4).

Depending on whether the rotation is global or local, objects can be rotated around their own center, the center of the environment, or even the center of their "parent" in a **hierarchy** of objects (for more on transformation of model hierarchies see Chapters 5 and 11). When rotating an object around its own center it is possible with many programs to reposition that center. Consequently, the center of rotation of an object may not always be placed in the geometric center of the object.

Rotations can be used to present different sides of an object to the camera. Rotations are very effective for arranging subtle details in a scene, such as to expose sides of a model with the most interesting shapes or detail, to simulate motion, or to emphasize the perspective of the objects in the scene.

Because rotations always happen around an axis, it is important to know which way rotations are supposed to occur. Depending on the value (positive or negative) that defines a rotation, the rotation can be clockwise or counterclockwise. In a right-handed coordinate system, positive rotations are always counterclockwise and negative rotations are clockwise. A simple method for remembering the direction of rotations consists of representing the axes on which the rotation is taking place with our extended right hand thumb as shown in Figure 3.4.5. If the thumb points to the positive direction of the axes, the direction of a positive rotation is defined by the direction in which we close the hand and make a fist.

Scaling

Scaling is a geometric transformation used to change the size and/or the proportion of an element or a group of elements. Scaling can be applied to an object in a proportional or a nonproportional mode. **Proportional scaling** consists of resizing an object along each axis in equal amounts. The result of proportional scaling is always a larger

or smaller object with the same proportions as the original object. With **nonproportional scaling** the object may be resized by different factors along each axis. Nonproportional scaling can be used to change the proportions of a three-dimensional object so that it becomes taller or shorter, wider or narrower, or deeper or shallower (Figs. 3.4.6 and 3.4.7). Because of its ability to easily modify the shape of objects, nonproportional scaling is widely used in computer animation to simulate the "squash and stretch" distortions typical of three-dimensional objects in motion.

When a scaling operation is performed, not on a single object but on all the objects in the environment, we get an effect that is similar to a camera zoom.

Perspective Projection

Perspective projection is a transformation of critical importance because it makes possible the representation of three-dimensional environments on the flat surface of the computer's monitor or a sheet of paper. A perspective view of a three-dimensional scene is created by projecting each point of an object from the viewpoint onto the picture plane. The points in the three-dimensional object coordinate system are then transformed to the two-dimensional image coordinate system.

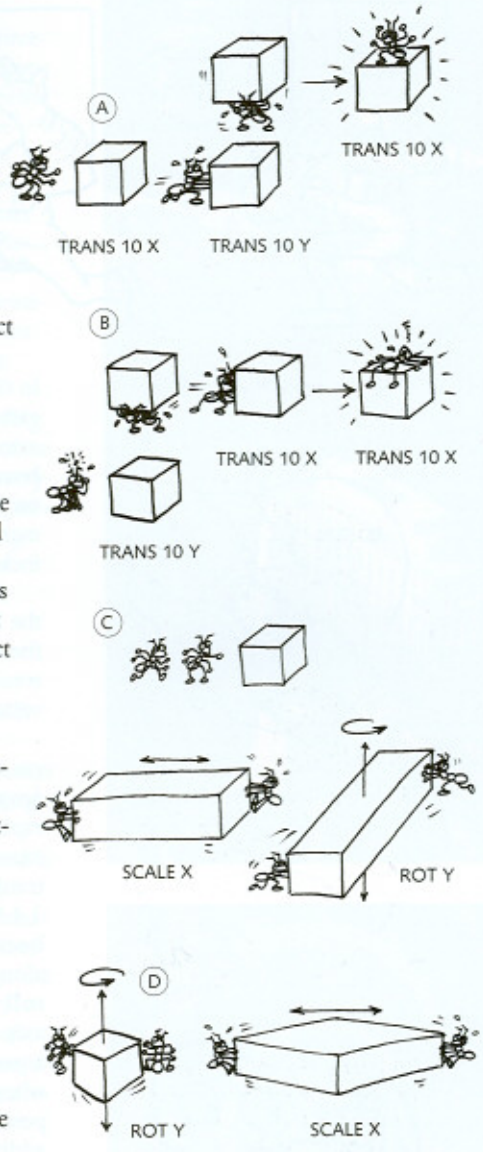
Perspective projection is a transformation that happens automatically in virtually all three-dimensional software. It is not necessary that we ask for perspective projection each time we do something to our scene. The three-dimensional environment is constantly being transformed into a two-dimensional view using perspective projection techniques. The final two-dimensional images obtained on the screen can be modified by moving the objects in the three-dimensional environment or by altering the camera. See Chapter 7 for more information on perspective projection.

Navigation

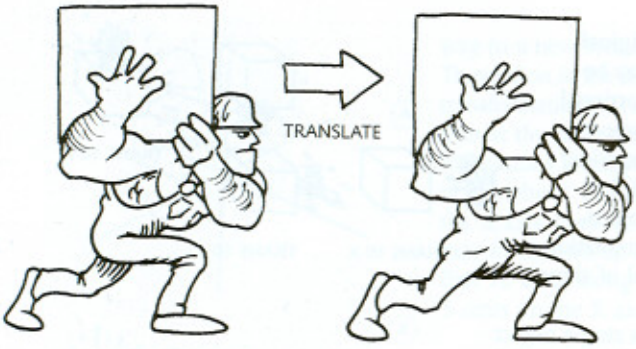
Navigation usually refers to the motions that place the camera in different parts of the scene. Navigation can be used during the modeling process for looking at the models from points of view that show the model in more detail. Navigation can also take place before the rendering process to focus on areas of interest, or before the animation process to place the camera where it helps tell a story more effectively.

The spherical or azimuthal coordinate system is often used to navigate through the world by specifying camera positions in terms of the camera's angle around the horizon, its angle above the horizon, and its distance from the object (Fig. 3.4.8).

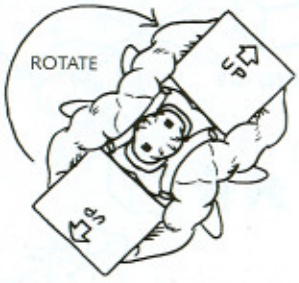
Navigating through three-dimensional space by moving the camera can take place on any of the four view windows provided by



3.4.2 The first two examples (A and B) show the same translations applied to the same object in two different sequences. The results are identical; in both cases the object ends up in the same place. Examples C and D show two different resulting shapes after applying the same rotation and scaling to an object, but in a different sequence.



3.4.3 An object being translated.



3.4.4 A bird's-eye view of an object being rotated.

3.4.5 In a right-handed coordinate system the direction in which your hand closes to make a fist is the direction of a positive rotation around any axis represented by the extended right-hand thumb.

almost all three-dimensional software. These windows include the perspective view and the three orthographic views: top/bottom, front/back, and right/left. All the camera motions described here can take place in the perspective window, but in some programs some camera motions—such as yaw/pitch or azimuth/elevation—cannot be viewed in the orthographic views because these motions can be calculated only in three-dimensional space, and not on a flat surface.

The basic characteristics of a virtual camera in three-dimensional space (what the camera sees) are defined by the camera position, its point of interest, and the camera lens. These characteristics can be quickly set by typing numerical values on the keyboard. These characteristics can also be set interactively by clicking on the control buttons provided by some programs, or by directly manipulating the camera with a variety of input peripherals that include the mouse, graphics tablet, trackball, joystick, or dial box.

Even though all camera positions and moves can be input from the keyboard, it is a lot more practical and fun to position and move the camera interactively. In any case, each possible camera move will result in the modification of at least one of the camera's three basic values: position, orientation, and focal length.

The motions of computer animation virtual cameras are based on the camera moves defined in traditional cinematography. Most programs use the same camera names used in traditional cinematography, but some use a different nomenclature. All of the camera moves, even the most complex ones, can be expressed in terms of translations or rotations around one or several camera axes (Fig. 3.4.9). A **dolly** is a translation of the camera along its X axis, a **boom** is a translation along its Y axis, and a **truck** is a translation along its Z axis. A **tilt** is a rotation of the camera around its X axis, a **roll** is a rotation around its Z axis, a **pan** is a rotation around its Y axis. Sometimes a tilt is called a **pitch** (as in airplanes pitching), and a pan is called a **yaw**. A **zoom** is a special type of camera move where only the camera's simulated focal length is modified but its position and orientation remain untouched. (Read Chapter 7 for additional information on cameras and camera moves.)

3.5 File Formats for Modeling

There are many formats for saving the information contained in three-dimensional geometry files. Many of the existing **file formats** containing descriptions of object geometry are exclusive to specific computer programs and are not portable. This means that the information contained in these files is formatted according to conventions that are particular to the software in question, and the files are not compatible with other programs. A few geometry file formats are **portable**, which

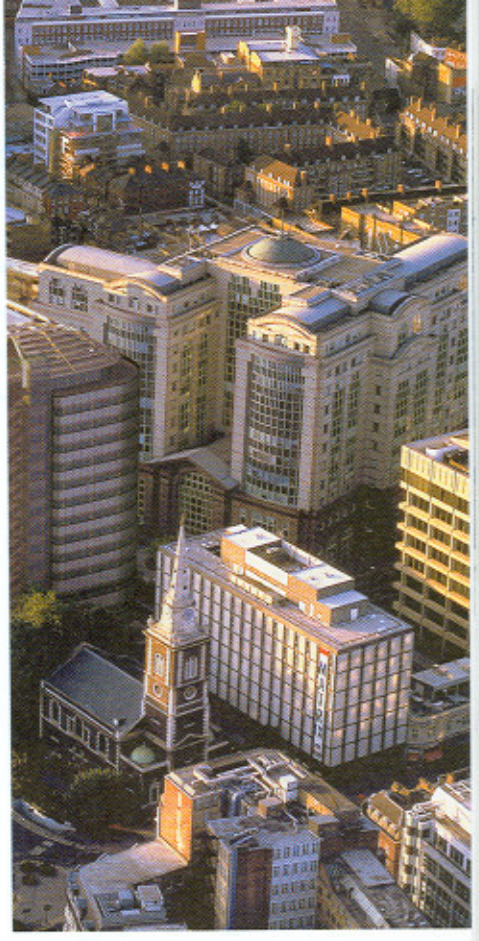
means that they can be exchanged among several programs. All three-dimensional models created within a modeling program can be save and retrieved in the application's own **native file format**. A specific three-dimensional software, for example, can save all of the three-dimensional models created with it in a file format that has been optimized for its own requirements.

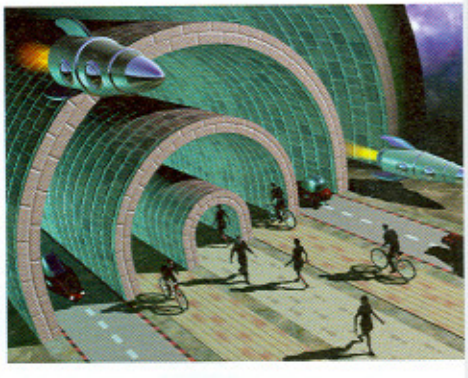
The obvious advantage of using native, or custom, file formats is that it is easy and fast for any particular program to read files in its own native format. Files saved in native formats usually load faster and require less space for storage. There are a number of **conversion utilities** that can translate geometry files in native formats between applications in varying degrees of accuracy. Models that have been built with standard techniques or that have a simple topology can usually be converted in this way very successfully (Fig. 3.5.1). But trying to convert complex modeling files from one native file format to another might modify some subtle details—or destroy them altogether—and might require a fair amount of manual adjusting. Solutions to the format incompatibility problem include using "universal" file formats for saving information about three-dimensional models or converting one native file format directly to the native file format of another program.

The file formats used for transporting geometry information between modeling programs are often called **universal file formats**, and two of the most popular ones include OBJ and DXF. The **OBJ** or **.obj** format, short for object, was popularized by Alias|Wavefront software products for high-end computer animation and visual effects production. The **DXF** format, short for **Drawing Interchange Format**, was developed by Autodesk, Inc. for handling both two- and three-dimensional geometry information, and is widely used in computer-aided design (CAD) applications. Even when using universal file formats to save three-dimensional information, there can be minor differences in the ways different programs interpret the information. This latitude in interpretation is due to the fact that many of the universal file formats describe three-dimensional information in a very general way. The DXF files, for example, contain some two-dimensional information that is often discarded when imported by three-dimensional software. It is also common for three-dimensional programs to interpret in different ways the precision and/or the curvature of a line that defines a surface.

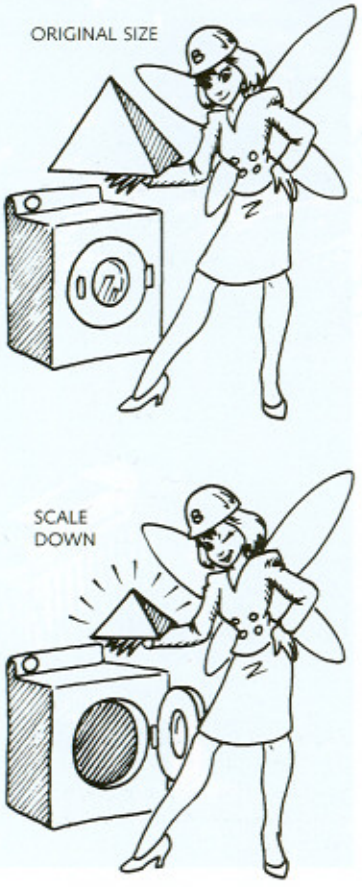
Virtually all three-dimensional modeling programs offer some degree of **file conversion.** That capability is found either inside the standard file management options (under a command or menu option name such as Import or Retrieve) or as a standalone utility conversion program that can be executed outside of the modeling program. Most programs can also **export** its three-dimensional modeling data into other native or universal file formats.

Many of today's three-dimensional modeling programs offer some degree of **foreign-to-native** file format conversions. The number of data formats that a particular three-dimensional program may

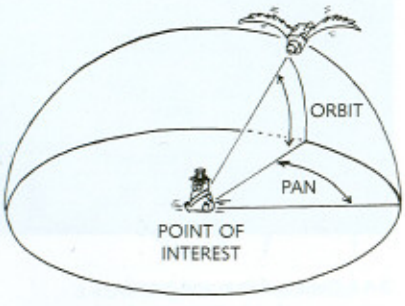




3.4.6 These buildings and tunnels were created by duplicating and scaling a variety of shapes and an extruded arc. (Top: aerial view of St. Botolph's House. © Hayes Davidson. Bottom: © Jim Ludtke.)



3.4.7 An object being scaled, in this case by using a shrinking machine.



3.4.8 The spherical or azimuthal coordinate system allows for orbiting or panning the camera around the subject.

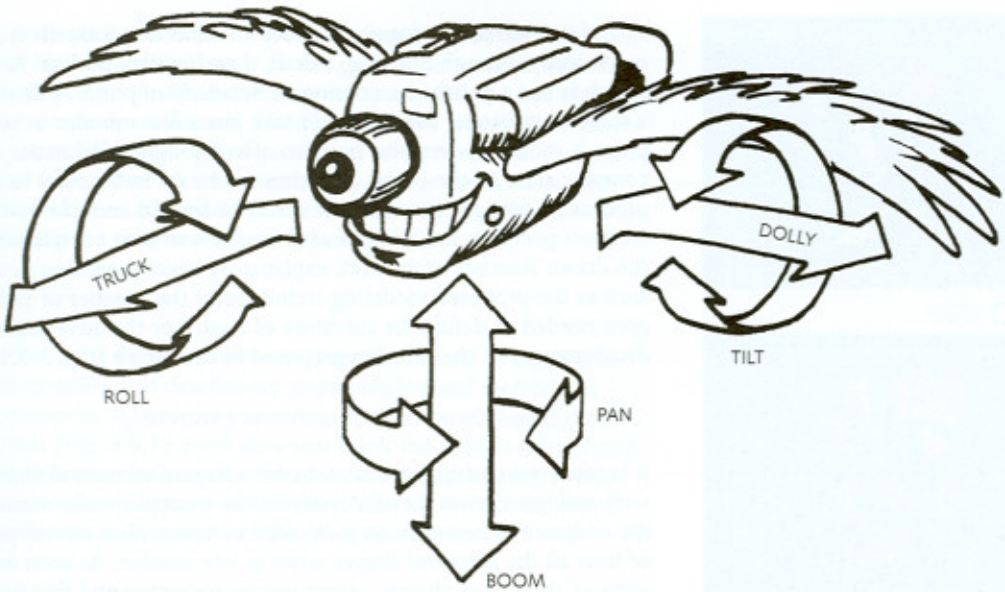
be able to convert may range from just a couple to several dozen formats. All file format conversions are controlled by **import filters,** which are tables that instruct the conversion utility program how to translate each of the elements encountered in the original—or foreign—file. Figure 3.5.1 shows how a particular file conversion program exports a data file into three different formats. Even when the most reliable file formats and conversion filters are used there are always small variations between the results obtained with different programs. Figure 3.5.2 illustrates the wide range of options available in software when importing or exporting data in the DXF format. One of the reasons for offering so many options in a conversion to a "standard" format is because not all the aspects—or options—of the DXF format are supported by all programs that are capable of reading DXF files. Figure 3.5.3 illustrates how a specific program deals with one aspect of the native-to-DXF file format conversion.

The results obtained with different file conversion utilities vary widely. Some file conversions are almost flawless (with only minor details requiring adjustment), while others rarely produce desirable results. There is no easy way to know if a file conversion program will work perfectly or not: each has to be tried and evaluated.

During the early 1990s the **Virtual Reality Modeling Language**, widely known as **VRML**, gained popularity as a convenient way to describe in a portable format three-dimensional environments for real-time online display. VRML is barely used today, but it brought forth several innovations that are now being implemented in other fledgling streaming three-dimensional standards for use in the Web and other online applications. One of VRML's innovations was to allow for the creation of virtual reality environments where multiple participants can interact with each other in three-dimensional spaces. **X3D**, for Extensible 3D, is a newer open-source standard for implementing interactive three-dimensional environments on the Web and in embedded devices. X3D is a newer **scene description language** that addresses object geometry, as well as rendering, navigation, interaction, and networking of virtual environments. Other file formats used for real-time display are covered at the end of Chapter 5.

3.6 Getting Ready

Modeling can be a time-consuming activity because of the great attention to detail that is required. This can only be accentuated if one encounters a lot of unexpected hurdles along the way. In spite of the flexibility offered by many computer modeling systems, trying to fix complications caused by poor planning sometimes can be more time-consuming and headache-inducing than starting over again. For this and other reasons—also related to practical issues such as time and budgets—it is very important to consider some of the preproduction guidelines listed below. All of these strategies are to take place before one actually starts building the three-dimensional models.



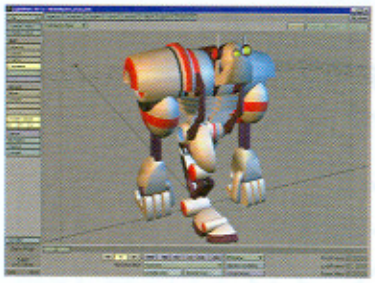
3.4.9 Navigating through three-dimensional space can be done by using the traditional cinematography camera moves: dolly, boom, truck, tilt, pan, and roll.

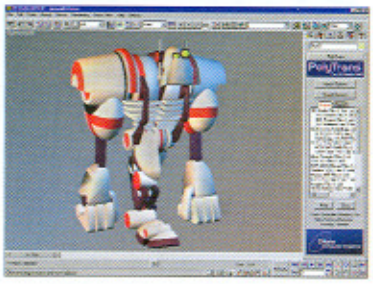
Sketch Your Ideas First

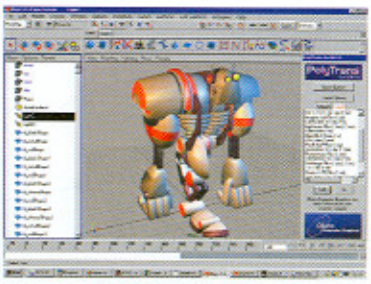
Sketching your modeling ideas on paper or modeling them in clay can sometimes be faster and more economical than starting to model directly on the computer. While there is nothing intrinsically negative about creating three-dimensional models without having a sketch at hand, in most cases starting that way increases the chances of running into small problems that might have been easy to avoid. Small details such as the ways in which two complex shapes will blend into one another, for example, can be visualized faster and cheaper on paper or modeling clay. In most cases, it is not an issue of whether something can be sketched and visualized directly on the computer, but one of economics: It can be much more expensive to sketch with the computer system than with a number two pencil, plain white paper, and an eraser.

Another advantage of developing three-dimensional ideas on paper or with modeling clay is that both media are absolutely portable and do not present any type of compatibility problem. This is especially important when you are required to present your work to others before actual production starts. It is very easy to show someone a sketch on paper; that can be done anytime and anywhere. There is no need, for example, for your client to travel to your location because they do not have computers or for you to have to reserve one of the workstations in your company so that your client can see your ideas. There are few techniques for sharing your visual ideas with others as direct, portable, participatory, and friendly as a sketch on a piece of paper or a clay scale model.

3.5.1 (Next page. Credits: ApeBot model © 1999 Matt McDonald, Vision Scape Imaging, and Newtek, Inc. Respective screen shots © 1999 Newtek, Inc., Kinetix, Inc., Avid Corp., and MultiGen-Paradigm Corp. Images courtesy of Okino Computer Graphics.)







3.5.1 Four images show how a geometry file originally created with Lightwave™ is converted with PolyTrans v. 2 software into three other native file formats: 3D Max, Softimage, and Maya.

If you are still unconvinced, keep in mind that a sketch is just a quick study, a rough drawing, a draft, a preliminary outline. A sketch is not a polished rendering or detailed sculpture. A sketch for a three-dimensional model should take just a few minutes to complete; it should be tentative but also offer enough detail in the more complex areas of the object. Sketches are meant to be aides in the production process, not works of art to be framed and admired by museum-goers. In some instances, it can be useful to complement the drawn sketches with short, explanatory notes regarding details such as the proposed modeling technique or the number of polygons needed to define the curvature of a part, or the advantages and disadvantages of the solution proposed in the sketch (Fig. 3.6.1).

Use Multiple Camera Views While Modeling

It is often quite advantageous to model a three-dimensional object with multiple camera views. A sculptor, for example, walks around the sculpture as he works on it in order to have a clear mental picture of how all the parts and shapes relate to one another. As soon as some of the shapes change, others require reshaping and fine-tuning. In much the same way, it is convenient for an individual using a computer three-dimensional modeling system to look at the object from multiple points of view as it evolves. This can be easily accomplished by constantly rotating the object around its own center. But it can be more convenient to have several views—often called windows—active while modeling. That way one can get immediate feedback from different points of view while concentrating on the modeling.

Most three-dimensional modeling systems allow having four active views open at once (Fig. 3.6.2). It is common to use a front view, a side view, a top view, and a camera view. The camera view usually allows for total control of the point of view: the simulated camera can be placed close to the object being modeled—for examining details—or far away—for evaluating the overall shape. Some viewing positions such as 60-0-30, 45-0-45, 20-0-120, 45-0-220, and 30-0-60 are popular for positioning cameras during the modeling process because these positions resemble the angles of some of the standard three-dimensional projections commonly used in drafting.

In certain situations, the more views required from the computer program, the longer it will take to process the information and to update the screen. This may force you to work with just the camera view and one other view that can be switched between front, top, and side views as needed. (Read Chapter 7 for more information on setting the camera.)

Write Your Numbers Down

Writing your numbers down can often help both before you start modeling and throughout the modeling process. Initially it is important to write down (on a simple piece of paper, in the project spec

sheet, or in your production journal) the numbers that describe general but important things such as: the general dimensions of your object, its position in three-dimensional space, and the boundaries of the active area or workspace. Writing down this type of information can be especially useful when you return to a project that was put on hold for a long time or when somebody who is not familiar with the project has to take over because you have decided (or somebody has decided for you) to work on something else.

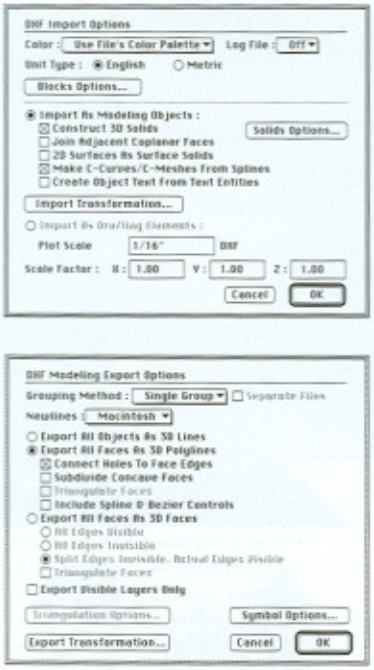
Do Not Lose the Blueprints

Blueprints are a necessity in cases where the objects to be modeled are too complex and detailed for improvisation and memory. In those instances, it becomes paramount to hold on to the original blueprints (Fig. 3.6.3). Even after you think the models are finished, you or someone else on your team (or on the opposite team) may decide that the models are not finished after all. In such an event, you or someone else may need the blueprints again.

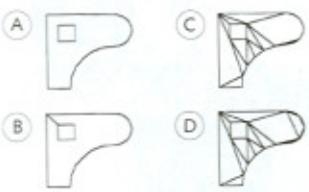
Polygons or Curves?

Most of today's three-dimensional modeling programs are exclusively polygon-based, but many are curved-based, and some offer both capabilities. Choosing between polygons or curves for modeling a three-dimensional object has obvious implications as far as the model's shape is concerned (for more on these implications see "Geometric Primitives" in Chapter 4). However, the rendering implications of modeling with polygons or curves are less obvious but critical in some cases (Fig. 3.6.4).

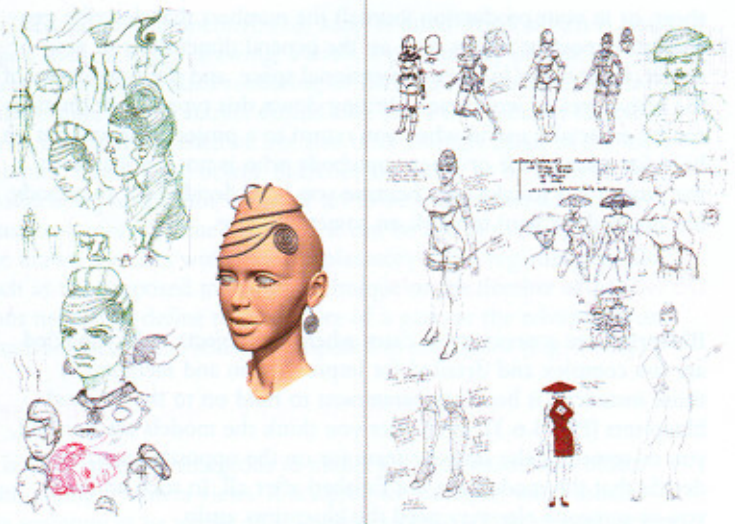
As we will learn later in Section III: Rendering, many rendering programs require polygonal structures from the modeling programs. This means that whenever curves are used—but before the three-dimensional model can be processed by most rendering programs—the curves have to be converted to polygonal structures. In most cases, this conversion is not a problem: Many programs perform this conversion automatically. However, in some cases, whether to start modeling with polygons or curves can become an issue that requires planning. For example, some curve-based modeling programs do not accept polygonal models, and likewise, some polygon-based modeling programs will have a difficult time reading files of models that have been specified with curves. Furthermore, in many three-dimensional modeling programs—even those that offer both polygonal and spline-based modeling techniques—some advanced functions such as bevelling or clipping will work only on polygonal meshes. Most of the sophisticated programs that offer two-way conversions between polygon meshes and spline surfaces do so at the expense of the shapes involved. When these conversions are performed there is always a significant amount of distortion that sometimes requires time-consuming, point-by-point rearranging.



3.5.2 Dialog boxes to control the import and export of files in the DXF format. (Dialog boxes from form•Z. © 1991–1995 auto•des•sys, Inc.)



3.5.3 The decomposition process of a concave shape with a hole (A) when exported to the DXF file format. First all holes are connected to the edges of the shape (B), then the concave shape is decomposed into several convex parts (C), next all parts are further decomposed into four-sided parts (D), and finally, the shape is triangulated so that all the component parts become elements in a triangular polygonal mesh. (Courtesy of auto•des•sys, Inc.)



3.6.1 Sketching your ideas and trying different poses, gestures, and variations can help to refine your ideas before the beginning of modeling production. (Sketches and rendered image from *Thomas in Love*, © 2002 Aentre Chien et Loup/JBA/RTBF.)

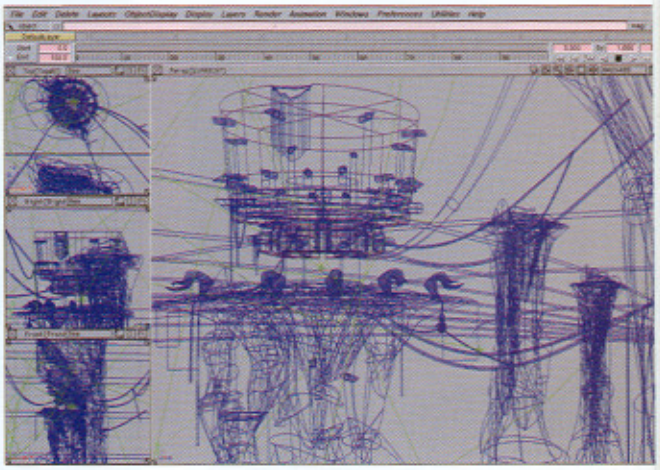
Will the Model Be Used for CADAM?

Objects that are modeled just to be rendered or animated are built very differently from objects that will serve as models for **computer-aided design and manufacturing** (CADAM). It is extremely important to know whether one's models will be used for CADAM. If so, a specific modeling methodology has to be chosen and followed consistently throughout the project. Figure 3.6.5 shows a three-dimensional model that was used to fabricate a sculpture with stereo lithography techniques (Fig. 15.8.3).

The two significant differences between modeling for CADAM projects or modeling for animation projects lie in the modeling technology used, and in the fabrication and structural implications of the objects modeled. Very few computer systems offer both modeling techniques. The majority of software is either just boundary-based or solid-based. In those situations this issue is automatically solved by the limitations of the software. It is in cases when the software has both capabilities that we have to choose between object shells or solid objects. When we build three-dimensional objects for rendering or animation purposes, we are almost always interested in the surfaces of the objects and very rarely in their inside volume. For that reason, when we model objects for rendering and animation, we usually use **boundary** and **geometry modeling techniques**. Boundary geometry focuses on the surface or **shell** of objects, and ignores the **volume** and inner structure.

This is similar, for example, to making a photographic portrait of a person. We are mostly interested in capturing expression, gesture, skin texture, posture, eye color, and other details. In general, we are quite uninterested—as far as completing a successful portrait—in whatever is under the skin of this person: muscles, bones, and organs.

On the other hand, when we build three-dimensional computer models for the ultimate goal of fabricating them—with a computer-controlled milling machine or stereo lithography system—we are fundamentally interested in the inside of the object, its structural soundness, and whether the shapes we have included in our model can actually be fabricated efficiently (Figs. 15.1.1 and 15.8.4–15.8.5.) For all these reasons when we model objects for CADAM projects we often use **constructive solid geometry** (CGS) techniques. These techniques are not concerned with how fast a three-dimensional model would render, how realistic it would look, or how efficiently it would animate. Instead, CGS techniques focus on whether our three-dimensional model meets structural criteria, whether it has the exact required dimensions, and whether it contains the specified amount of material.



3.6.2 Most three-dimensional programs can display multiple projections of the camera view. Shown here are the perspective projection, and the top, front, and side orthographic projections. See the sketches for this environment in Figures 2.4.2 and 2.4.3. (© 1999 Oddworld Inhabitants, Inc. All rights reserved.)

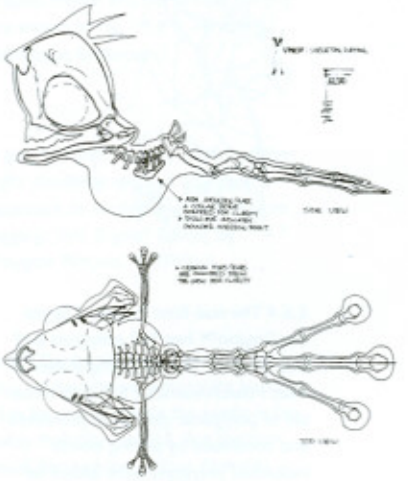
Modeling Is Related to Rendering and Animation

The life of a three-dimensional computer-generated model rarely ends with the modeling process itself. Most three-dimensional models go on to be rendered, and many continue through the production process to animation.

As you will read later in this book many creative and technical decisions made during the modeling process can simplify, complicate, or even paralyze the rendering or animation processes or both. It would be premature to explain which modeling solutions are more likely to complicate a certain rendering technique or an animation sequence. (It is certainly hoped that you will gain this information by reading the entire book.) For now, keep in mind that before you embark on future modeling projects, you should get as much information as possible about the plans regarding the rendering and animation of the objects, if any.

For example, a certain rendering technique, production deadline, or camera position may require that you cut in half the number of polygons used to define a section of the object, or the animation script might require that you group the objects a certain way (Fig. 3.6.6). If you know about either of these conditions in advance, you will avoid the difficulty of taking apart a model that is finished in order to try to reduce the number of polygons or having to undo a complex five-level hierarchical structure with hundreds of objects in order to reestablish some basic links in a different way. Figure 3.6.7 shows the same geometry at two different **levels of detail** (**LOD**) created with a **polygon reduction** software.

Keep the rendering and animation requirements of your project in mind during the modeling process. This will help to keep wasted time to a minimum.



3.6.3 Blueprint detailing shape, proportions, and skeleton of a character. (© 2003 Oddworld Inhabitants, Inc. All rights reserved.)

3.6.4 The real-time model of *Spyro the Dragon*™ has 352 polygons and 230 vertices. The modeling started in Alias Power Animator 8.5 with a couple of polygonal geometric primitives and continued by pulling points. Individual polygons were added for features like the eyes, feet, and interstices between joints. (Spyro the Dragon™. Images Courtesy of Universal Interactive Studios, Inc. and Insomniac Games, Inc.)

Check the Preferences File

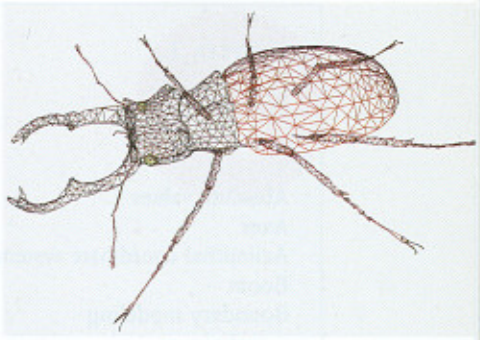
Remember that both the three-dimensional modeling program that you are using as well as your computer's operating system keep their preferred, or default, settings in a **Preferences file**.

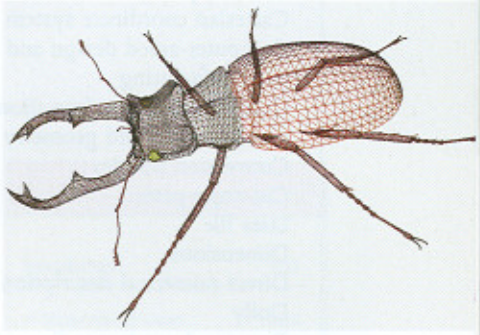
The contents of the Preferences file are important because they control directly and indirectly the result of many operations, functions, and tools. Some of the settings contained in a Preferences file may include, for example, the units used to specify the dimension of objects being modeled, or whether a tool for creating cubes will define by dragging from the center of the cube outwards or from one corner of the cube to the opposite corner. As you can see, some of these settings may affect the result of your three-dimensional modeling, rendering, and animation.

In general, the last person who opened a file or who used the program or the computer system is capable of altering the files by changing the Preferences file. In some systems, Preferences files are attached to the three-dimensional computer program and in some cases to the model files themselves. Check your system for details.

Check Your Memory Requirements

Most of today's three-dimensional modeling software will allocate enough of the system's memory (RAM and/or virtual). This means





that in most cases you do not have to be concerned about whether there will be enough space in the computer's memory for you to build your model. But sometimes, especially when complex three-dimensional models are being created in small computer systems, the issue of not having enough memory can become a problem. Most professional three-dimensional software today is comfortable with about 512 MB of RAM, but happier with one GB or more. Sometimes when the system does not automatically make sure that there is enough memory for you to keep building, the system will unexpectedly run out of memory and freeze. Also keep in mind that many modeling programs can recover from errors very gracefully (and will allow you to restore all your data), but others cannot.

Save Your Work Often

Save your work often, every 15 minutes or so, and make periodic backups of your important data files. Some applications automatically save the file(s) that you are working on at regular intervals specified in advance. Take advantage of such features.

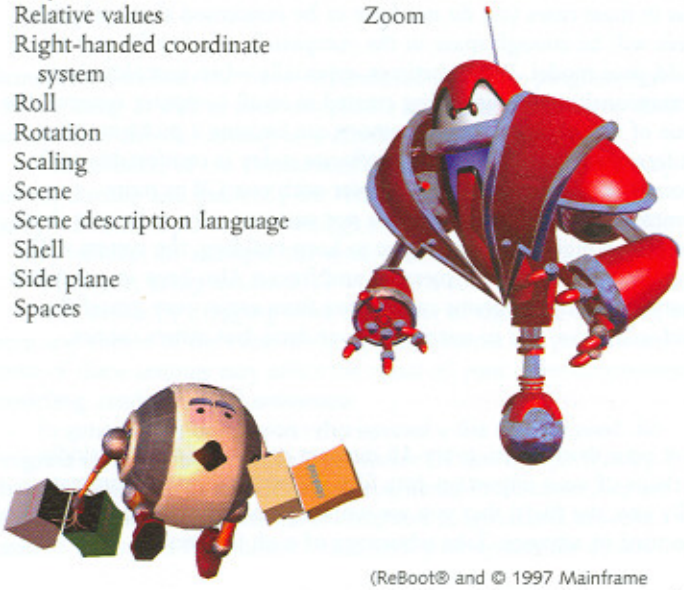
3.6.5 (opposite page) Computer visualization of a model built with Rhinoceros software. The stereolithographic model and the bronze casting are shown in Fig. 15.8.3. (© 1999 Bathsheba Grossman.)

3.6.6 (top left) Projects created within tight deadlines requires an efficient approach to modeling and animation rigging. (*Mr. Digital Tokoro* by Polygon Pictures. © TPVN.)

3.6.7 The beetle low-resolution geometry model (above, top) might be suitable for scenes where the beetle is far away from the camera, but the high resolution version (above, bottom) might be better for close-up shots or for scenes where the beetle is the main character in the scene. (Polygon reduction and rendering by VSimplify software. © 1999 Virtue Ltd.)

Key Terms

Absolute values
Axes
Azimuthal coordinate system
Boom
Boundary modeling techniques
Cartesian coordinate system
Computer-aided design and manufacturing
Concatenated transformations
Constructive solid geometry
Conversion utilities
Cut-copy-paste
Data file
Dimensions
Direct numerical description
Dolly
Drawing Interchange Format
DXF
Edge
Export
Facets
File conversion
File formats
Foreign-to-native
Front plane
Geometric transformations
Geometry files
Geometry modeling techniques
Global coordinate system
Global transformations
Hierarchy
Import filters
Left-handed coordinate system
Levels of detail, LOD
Line
Local coordinate system
Local transformations
Mathematical operations
Modeling
Native file format
Navigation
Nonproportional scaling
Object coordinate system
OBJ, .obj
Objects
Open Inventor
Pan
Perspective projection
Pitch
Point
Polygon reduction
Portable
Preferences file
Proportional scaling
Rectangular coordinate system
Relative values
Right-handed coordinate system
Roll
Rotation
Scaling
Scene
Scene description language
Shell
Side plane
Spaces
Spherical coordinate system
Structures
Surface
Three-dimensional space
Tilt
Top plane
Transformation matrix
Translation
Truck
Universal file formats
Volume
Virtual Reality Modeling Language
VRML
Workspace
World coordinate system
World origin
X3D
Yaw
Zoom



CHAPTER 4

Basic Modeling Techniques

Summary

THE BASIC TECHNIQUES FOR MODELING three-dimensional objects with a computer system are covered in this chapter. The chapter starts with a short but important note about lines, their use in the creation of surfaces, and the general differences between polygonal meshes and curved surfaces. Following that is a discussion of the simplest geometric modeling tools available in most of today's systems. After that comes a survey of several derivative techniques including revolving, extrusion, and sweeping. Techniques for creating terrains and simple free-form objects are followed by a survey of utilities that are useful to modelers of all levels. An overview of modeling for real-time display concludes this chapter.

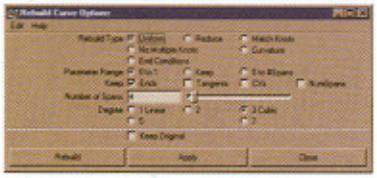
QUICK INDEX

4.1 Introduction

Just like traditional modeling techniques, the computer-based three-dimensional modeling process begins with an idea. Before the modeling process can start we try to visualize this idea of what we want to create, perhaps by creating some sketches or even detailed blueprints.

The conceptualization and design of the three-dimensional model usually constitutes the first stage in the simulation of a three-dimensional scene with a computer. From an artistic point of view this step is probably the most important one in the process because it is here where the basic characteristics of the scene are laid out: the shape, position, and size of the objects; the colors and textures; the lights; and the position of the camera. It is also at this stage where the basic ideas have to be analyzed and the best modeling methods chosen for each task.

I usually prepare the initial sketches that describe the three-dimensional objects and environments with traditional media, such as colored pencils or markers on paper. These sketches indicate the general characteristics of the objects such as size, relative position, color, and lighting effects. Once the sketches are finished, I analyze



4.2.1 The characters from this X-Box commercial are built with different types of lines and surfaces. The dialog box shows conversion options between different types of lines. (Top: Image courtesy of Blur Studio. Bottom: © Alias|Wavefront, a division of Silicon Graphics Limited.)

them and prepare a set of blueprints containing one or several detailed views of the object with dimensions.

There are many ways of translating the visual information contained in the sketches into numerical information suited for computer manipulation. Most three-dimensional modeling software allows users to build the model interactively. This means that the models that are being worked on can be displayed on the screen. Any and all model changes made by the user are displayed on the screen almost instantly. This interactive quality provides the visual feedback that is so important in developing the shapes of objects and the layout of three-dimensional spaces. Because of the lack of immediate tactile feedback when creating three-dimensional models with software, the real time visual feedback on the screen becomes almost indispensable.

When the modeling process is complete we usually end up with a file that contains a detailed description of the objects in our environment including information regarding their geometry, position, and hierarchy. Realistic images of the files can be obtained by rendering the file with some of the techniques presented in Chapters 6–9. There are many techniques for describing three-dimensional structures, and each produces different results and requires a different type of approach (Figs. 4.2.6–4.2.7).

Unless otherwise specified, all modeling techniques described in this chapter are based on boundary geometry and not on constructive solid geometry. As explained in Chapter 3, this means that the three-dimensional objects are built as hollow shells only and not simulated as true solid objects.

4.2 A Note About Lines

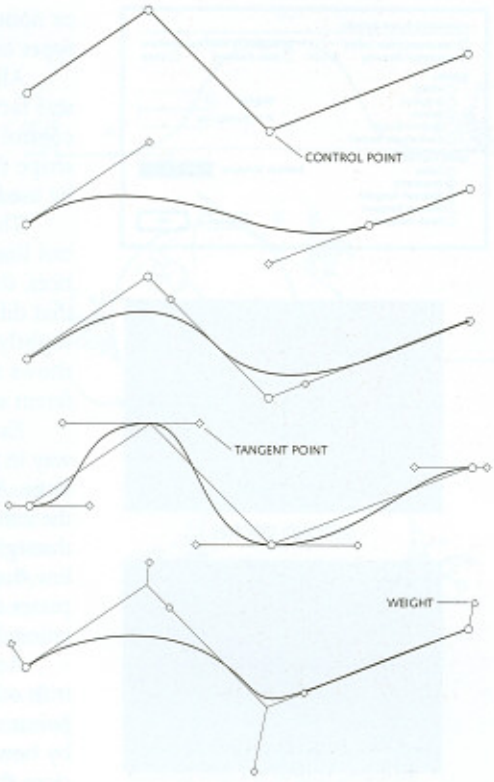
Lines are used to define the shape of the object and many of its surface characteristics. Lines are a fundamental component of all three-dimensional objects. For this reason it is important that we understand differences between types of lines, as well as their attributes and limitations. This section offers a brief characterization of some of the standard lines most commonly used in three-dimensional modeling. Keep in mind that the names used here are as general as possible, but your computer system may have a different name for a specific type of line or a line tool. The classification presented here is based on the practical characteristics of different types of lines, on their advantages and disadvantages, and also on their mathematical nomenclature. The following paragraphs explain some of the differences between types of lines. Please read them carefully. These concepts are crucial for understanding much of the material presented in the other chapters about modeling and rendering.

One obvious difference between lines is that some are straight and some are curved. Straight lines are concerned with defining the shortest distance between two points, but curves are concerned with subtlety of change and elegance of design. There are many differences between straight lines and curved lines: their mathematical

description, their behavior as they are used to model, the type of three-dimensional structures they yield, and in most cases, their visual appearance. Some three-dimensional modeling computer programs are capable of converting curved lines to straight lines and vice-versa, but in many cases the results of these conversions are sometimes surprising and might require considerable work before they can be used (Figs. 4.2.1 and 6.4.1).

Straight lines—as their name implies—do not have any curvature. Straight lines are defined by two endpoints only, and may have a slope but no change in angularity. In other words, the slope of curves is variable, but the slope of straight lines is not. In three-dimensional modeling programs, straight lines are sometimes called **polygonal lines** because they can be used to build polygons and polygonal meshes. The three-dimensional modeling computer programs that use exclusively straight lines are capable of building models only with polygonal meshes (and not with spline-based surfaces). The three-dimensional modeling computer programs that use curves are capable of building models with both curved surfaces and polygonal meshes.

Many programs offer two different line drawing tools. One draws straight lines and one draws curved lines. While it is difficult for a straight line to turn into a curve (because straight lines just do not have a variable for change of angularity), it is easy for a curve to turn into a straight line (just by setting the change in angularity to nothing). For this reason, many three-dimensional modeling programs offer just one single—and powerful—tool that draws just curves of all kinds, including curves that look like straight lines.

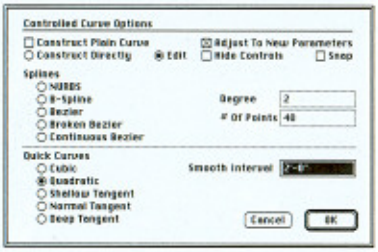


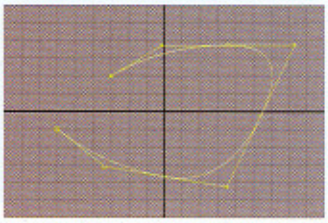
4.2.2 Five popular types of splines are illustrated here: linear splines, cardinal splines, b-splines, Bézier curves, and NURBS, or non-uniform rational b-splines.

Curved lines are usually defined by several points and deviate from a straight path without any sharp breaks in angularity. Curved lines are sometimes called curve segments and can be used to define curved surfaces and build meshes of curved surfaces.

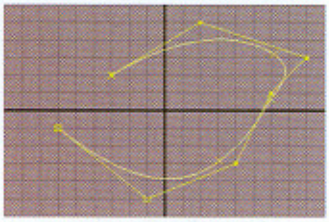
Curves are also often called **splines** because they resemble the physical spline—a long narrow strip of wood or metal—used by a draftsperson or construction worker to shape or fit curves between various points. The spline, traditionally used in the design and construction of ships' hulls, is shaped by lead weights. By varying the number and position of weights the spline can be turned into a smooth curve that passes through the required points. Even though not all curves fall into in the mathematical category of spline curves, some three-dimensional modeling programs use the term generically. That generalization is, of course, inaccurate.

There are many types of curves, and they can be catalogued based on their mathematical and geometric characteristics. In this text, however, we shall limit our summary to five of the more popular types of splines used in mainstream three-dimensional modeling: linear splines, cardinal splines, b-splines, Bézier curves, and NURBS

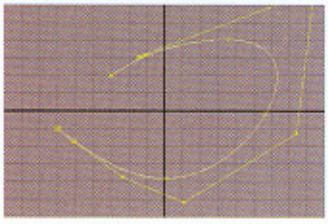




SECOND DEGREE



THIRD DEGREE



FIFTH DEGREE

4.2.3 The dialog box illustrates a few curve controls available in off-the-shelf software; and NURBS curves of second, third, and fifth degrees. (Top, dialog box from form•Z. © 1991-1995 auto•des•sys, Inc. Bottom, © Alias|Wavefront, a division of Silicon Graphics Limited.)

or nonuniform rational b-splines. Figure 4.2.2 illustrates these five types of curves.

All splines are generated from a defining polygon. Because of this fact splines are called **controlled curves**. The structures that control the splines are invisible—they are displayed only while we shape the spline—but they contain important information that can be used to reshape the spline.

The controls found in splines of different types include the control line or control polygon or hull, the control points or control vertices, the tangent points, the knots, and the weights. Keep in mind that different software programs use different nomenclatures and slightly different implementations of the controls. Figure 4.2.3 shows some of the controls for modifying splines provided by different software programs.

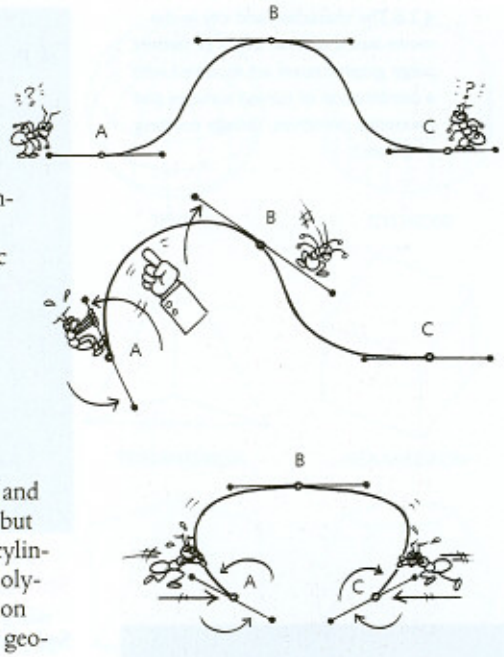
Each of the spline curves can be quickly characterized by the way in which it is controlled by the **control points** or **control vertices**. A **linear spline** looks like a series of straight lines connecting the control points. A **cardinal spline** looks like a curve that passes through all of its control points. The **b-spline** looks like a curved line that rarely passes through the control points. A **Bézier curve** passes through all of its control points. A **NURBS,** or nonuniform rational b-spline, does not pass through its control points.

Another easy way to characterize splines is to look at their controls other than the control line and the control points. Control points can control the **curvature** or **tension** of a curved line mainly by how close they are to one another and, in some cases, by how close they or their tangent points are to the curve (Fig. 4.2.4). The Bézier curve differs from the three splines mentioned here so far mainly because it has **tangent points** in addition to the control points. Tangent points are used to fine-tune the degree of curvature on a line without modifying the control points.

NURBS offer a high degree of local curve control by using weights and knots. These controls allow a portion of the spline to be modified without affecting other parts of the spline. One **weight** is attached to each control point, and they determine the distance between the control point and the apex of the curve. By default, all control vertices on a spline have the same weight factor, and that is called a **nonrational curve**. (B-splines, for example, are NURBS with equal weights.) When the values of the weights on the curve are modified then the curve is called a **rational curve**. Manipulating weights on a NURBS curve may improve the subtle shaping of a line, but it usually also slows down the rendering of the final model. Another disadvantage of working with different weight values is that many systems will ignore the data when model files are exchanged. In many cases, results similar to using different weight values can be obtained by placing two control points very close to each other.

The **knots** on a NURBS determine the distribution and local density of points on a curve. The minimum number of knots required to form a curve segment is equal to the degree of a curve plus one

plus the number of control points. The **degree of curve** refers to the high exponent in the mathematical formulas that generate curves. Each curve type (b-spline, Bézier, and NURBS) has a different mathematical formula, and each curve type may be created at different degrees (Fig. 4.2.5). The higher the degree of a curve the more computation is required to create it. Curves of the first degree correspond to linear segments, curves of the second degree correspond to quadratic curves, and curves of the third degree correspond to cubic curves. The higher the degree of a curve, the more control points and knots are required to form a curve segment.



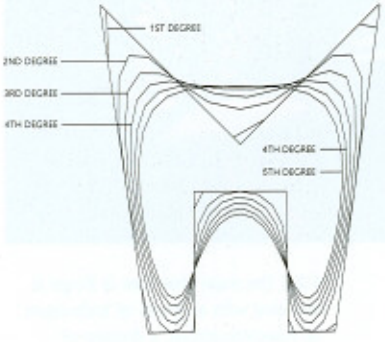
4.2.4 This sequence illustrates how control and tangent points affect the tension of a spline. In the middle drawing only the tangent points are moved. In the bottom drawing the control points are moved closer to one another and the tangent points change along to make the final curvature smoother.

4.3 Geometric Primitives

Virtually all three-dimensional modeling computer programs provide a collection of tools for creating simple shapes with a fixed structure known as **geometric primitives.** The number and type of geometric primitives varies from program to program, but the following list is a representative selection: cubes, spheres, cylinders, cones, toruses, regular polyhedra, and two-dimensional polygons. Figures 4.3.1 and 4.4.1 include some of the most common geometric primitives. Expressive characters can be built out of geometric primitives (Fig. 10.2.4).

In some programs, the different geometric primitive tools will appear all as a single menu selection while in others each or some of the tools may appear separately. In all cases, however, the feature that relates all geometric primitives to one another is the fact that they are standard shapes that the modeling program can create and manipulate effortlessly and usually from a simple predefined mathematical description. In principle, all geometric primitives may be created as polygonal structures or as curved patches.

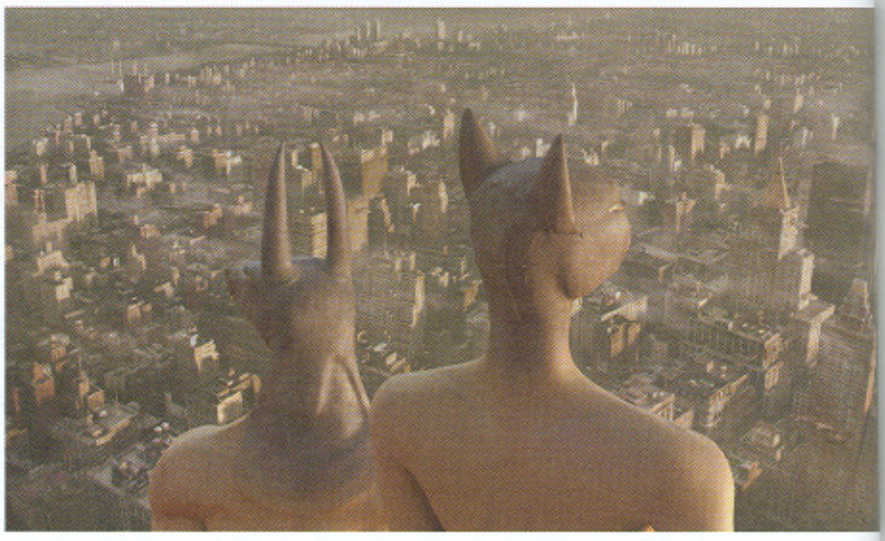
Geometric primitives can be used to represent simple shapes, or they can be used as the basis for more complex, composite three-dimensional shapes. In the former case, the shapes provided by the tool would require almost no modification except for changes to their position in space, size, and proportion in some cases. In the latter case, geometric primitives may be modified or used to build more complex objects with a variety of utility tools for trimming, attaching, and blending, among others. As most other tools, geometric primitives may be modified on-screen directly using the mouse, trackball, or electronic pen, or by typing the appropriate values in a dialog box.



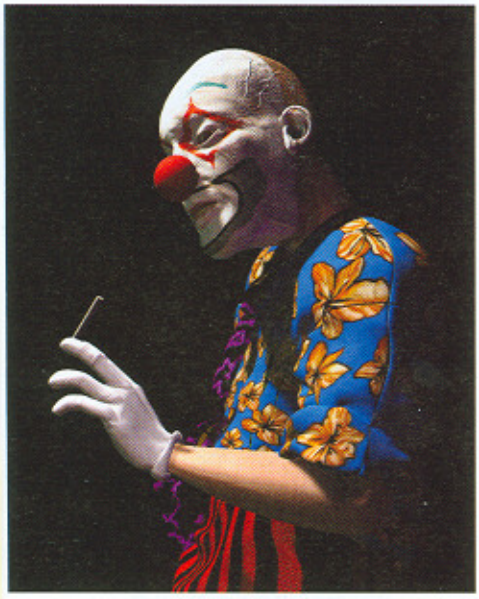
4.2.5 In general, the higher the degree of a spline, the further away it is from the controlling polygon (the outer shape with sharp angles).

Cubes

Cubes are usually modeled as six-sided, closed, three-dimensional objects. Since all sides of a cube have the same length, usually the only variable required for modeling cubes is the length of a side. Sometimes a number of subdivisions can be specified along each of the three axes. Cubes are almost always created as polygonal structures.



4.2.6 The characters and city in the movie based on Enki Bilal's *La femme piège* graphic novel are modeled with a combination of curved surfaces and geometric primitives. (Image courtesy of Duran.)



4.2.7 The main character in *Bingo* is modeled with a variety of techniques. (© AliasIWavefront, a division of Silicon Graphics Limited.)

Spheres

Spheres, like cubes, are also modeled as symmetric, closed, three-dimensional objects. In order to be defined, all spheres require a variable of **radius** or **diameter**, and they can be modeled as a polygonal structure or as a patch of curves. When modeled as polygonal structures, drawn with straight lines, a sphere's definition requires a number of divisions along the longitude (top to bottom) or latitude (around). These divisions resemble the parallels and meridians on a globe, and their number has a proportional effect on the geometric smoothness of the final shape. When modeled as curved patches, spheres require a type of spline to be specified in addition to the information listed above. Spheres are also very popular as the starting point for free-form modeling (Fig. 4.5.1).

Cylinders and Cones

Cylinders and **cones** are commonly defined as polygonal objects, and they may be shaped by the following variables: radius, height, number of longitudinal divisions, number of latitudinal divisions, and whether they are "capped" or not. The number of subdivisions used to build cylinders and cones defines the amount of modeling detail of the these objects. Objects with a small number of subdivisions can be rendered more quickly than objects with many subdivisions. When planning to render objects with image maps it is best to model them with a large number of subdivisions. Doing small rendering tests is essential to determine the optimum number of subdivisions that should be modeled into any geometric primitive. **Capping** determines whether the round sides of cones or cylinders are open or whether they are closed.

Toruses

A **torus** is a three-dimensional, closed shape that resembles a donut. A torus is like a cylinder that has been bent and stretched so that the two bases touch each other. The variables required to construct a torus are almost the same as those required for building a sphere, plus one additional variable, which is the interior radius. A full listing of modeling variables for a torus includes whether polygons or patches will be used, size of exterior radius, size of interior radius, number of latitudinal divisions, and number of longitudinal divisions. A torus is a geometric primitive that can also be built with radial sweeping techniques (Fig. 4.4.1).

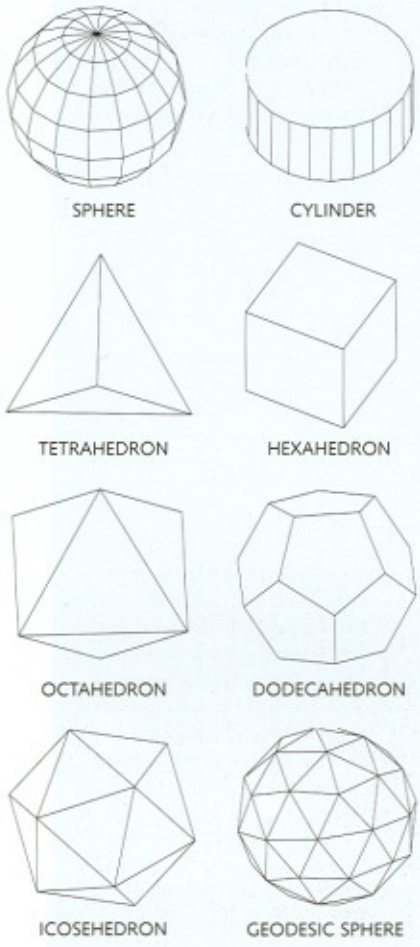
Regular Polyhedra

Many three-dimensional objects belong to the category of **regular polyhedra**, or objects with multiple facets. A polyhedron (singular of polyhedra) refers to a three-dimensional object that is composed of polygons. Some of the most common regular polyhedra include the 4-sided **tetrahedron**, the 8-sided **octahedron**, the 12-sided **dodecahedron**, and the 20-sided **icosahedron**. Regular polyhedra are usually modeled as polygon meshes and can be built by specifying a radius and a number of facets required.

Two-Dimensional Shapes

Two-dimensional shapes can be used to generate three-dimensional shapes by using derivative techniques such as extrusion or sweeping. Two-dimensional shapes usually include arcs, circles, spirals, triangles, squares, and other polygons.

Circles are two-dimensional, closed contours and require a radius or diameter, a number of control points, and a type of spline. **Arcs** are two-dimensional, open contours and require the same information that circles do plus the starting point and the ending point, both specified in degrees. **Spirals** are also two-dimensional, open contours and require a starting and an ending radius, a starting and an ending angle, a number of control points, and a height. **Polygons** (including triangles and squares) are two-dimensional, open contours, are almost always built with polygonal or linear splines, and can be defined by a number of sides and radius.

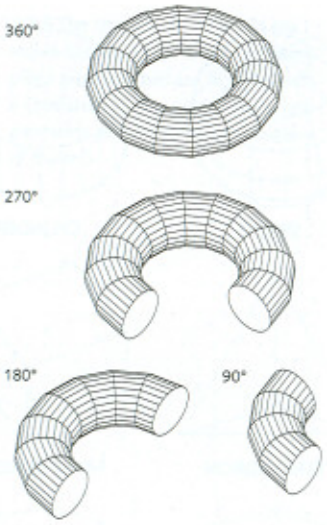


4.3.1 A sphere, cylinder, and regular polyhedra including a tetrahedron (4-sided), a hexahedron (6-sided), an octahedron (8-sided), a dodecahedron (12-sided), an icosehedron (20-sided), and a geodesic sphere.

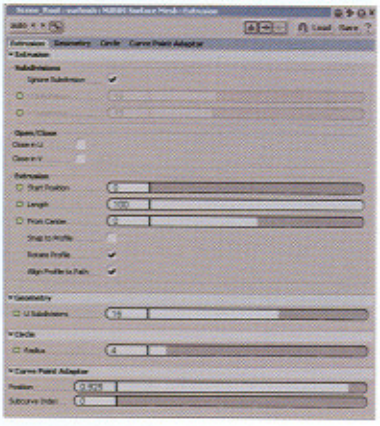
4.4 Sweeping

Sweeping is perhaps the most powerful derivative modeling technique, especially when you consider the complexity of the three-dimensional shapes created with it in relation to the simplicity of the input that is required for generating them.

The basic idea behind all sweeping modeling techniques consists of defining a two-dimensional outline that is swept along a prede-



4.4.1 Toruses and other geometric primitives can be created by sweeping a two-dimensional outline around an axis. Slices of geometric primitive shapes created this way are usually easier and faster than the equivalent shapes created with the geometric primitive tool and then sliced with a trimming tool.



4.4.2a The dialog box shows the type of controls used to create an extrusion along a straight path. (Image courtesy of Softimage Co. All rights reserved.)

fined path. As the outline is swept it defines a shape in three-dimensional space. The resulting three-dimensional model depends largely on the complexity of the **seed outline** and the complexity of the path (Fig. 4.4.2b). The three most popular sweeping techniques are extrusion, lathe or revolve, and free-form sweeps.

Simple Extrusion

In the conventional lingo of industrial design and manufacturing, **extrusion** stands for the process of shaping a material (such as plastic or metal) by forcing it with heat and pressure through a **die**. A die is a tool used for shaping or stamping different materials. The process of industrial extrusion is usually based on a stationary die just because of the limitations in handling both the hot materials and the heavy die. Meat grinders or pasta machines, for example, extrude the ground meat and the pasta through dies of different shapes.

Most three-dimensional modeling computer programs offer simple extrusion tools that—like their heavy industry counterparts—create three-dimensional shapes by starting with a two-dimensional outline and extruding or extending it along a straight path along one axis (Fig. 4.4.2a). Simple extrusion happens along any one axis. The two-dimensional outlines to be extruded can be created with geometric primitive tools or exported from other programs in highly portable file formats such as EPS (Encapsulated PostScript). Extrusion is sometimes called **lofting** because the two-dimensional outlines are duplicated and moved a level up.

Free-Form Sweeping

Some programs also offer the ability to extrude objects along paths of any shape and along any axis or combination of axes. An extrusion that takes place along several axes is sometimes called a sweep, sometimes called an extrusion on a path, or a **free-form extrusion** (Fig. 4.4.3). The results of free-form extrusion that is either scaled along the path or that is based on two paths are similar to those obtained with the skinning modeling technique decribed in Chapter 5.

Modeling by extrusion has been quite popular for centuries for creating meringue and ornaments on pastries, cookies, and cakes. The pastry extrusion tool, or die, moves with a sweeping motion along a decoration path. The motions of the pastry tool usually extend on a surface in a single continuous action, such as that of a broom or a brush, and in a wide curve or range.

Lathe

One very popular sweeping variation is commonly referred to as a **lathe** or a **revolve**. This form of sweeping is so popular that it is almost always presented as a standalone tool, separate from the general-purpose sweeping tool. The surfaces created with this technique

are usually called **surfaces of revolution**. The software-based lathe tool simulates a real lathe, which is a tool composed of a rotating base on which you place a cylinder of wood that is shaped by placing a steel blade on its surface as the base rotates around its vertical axis. A potter's wheel is used to perform an almost identical operation on a slab of clay. The clay or wood are cut uniformly around the cylinder as a blade or sharp tool moves in and out following a predefined path. The software lathe sweeps a two-dimensional outline around one axis; the two-dimensional outline may be open or closed. A new three-dimensional shape emerges as the two-dimensional outline is swept along a circular or radial path; it usually remains perpendicular to the sweeping path as they are swept. The resulting three-dimensional object is defined by the areas enclosed within the revolved two-dimensional outline. Surfaces of revolution require an angle of rotation and a number of steps or facets. The number of subdivisions is usually determined by the number of points on the outline used to generate the shape.

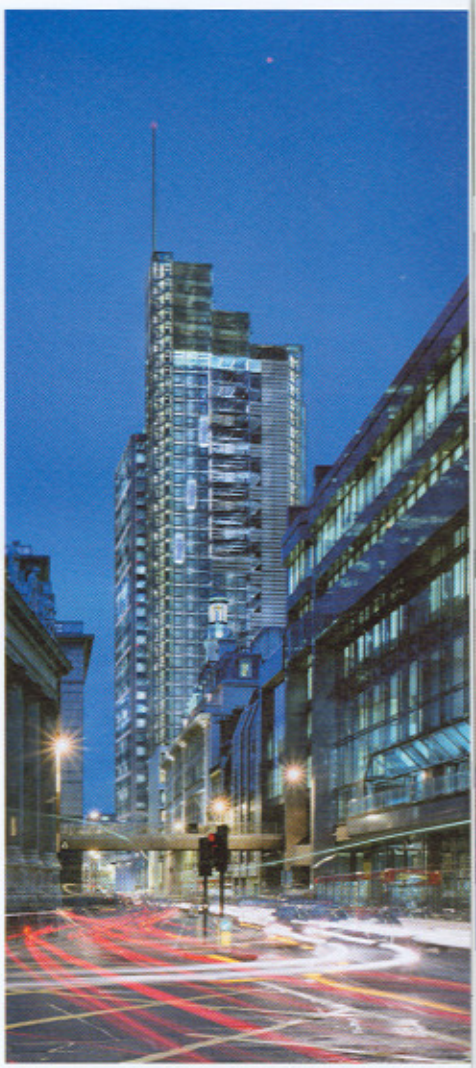
Surfaces of revolution that result from a 360-degree sweep are frequently closed, three-dimensional shapes. Sections—or slices—of three-dimensional shapes can also be created by sweeping less than 360 degrees. Two-dimensional outlines that do not touch the axis of sweeping will result in three-dimensional objects with holes (Fig. 4.4.4). In these cases, or when only a slice of a shape is created, the resulting shapes can be capped or uncapped.

The lathe modeling technique can also be used to recreate some of the geometric primitive shapes such as the cylinder and the cone (Fig. 4.3.1). Using the lathe for this purpose offers the advantage of increased control and economy of steps when trying to model a special version of a geometric primitive.

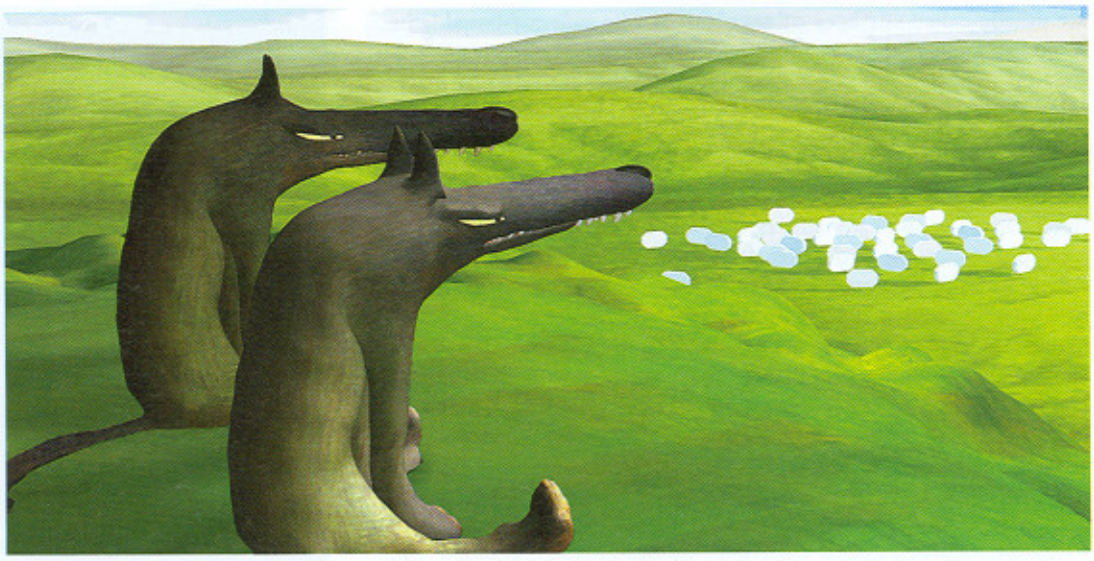
4.5 Free-Form Objects

Some projects require the creation of **free-form three-dimensional objects**. Creating these types of models can be time-consuming because they must be sculpted out of a mesh in a way that is very similar to sculpting or modeling a piece of soft clay. Simple free-form objects usually require a lot of point-picking, pulling-and-pushing, and overall "massaging" of the surface mesh (Fig. 4.5.1). The meshes may be planar, curved, or even based on subdivision sufaces. Planar or polygonal meshes, which are covered in this section, are particularly well suited for gaming projects where real-time rendering almost requires that polygons are used. Medium resolution polygonal meshes look smooth at a distance, especially when mapped with detailed image maps, but the illusion of smoothness breaks down as the model gets closer to the camera. High-resolution polygonal meshes like those obtained by scanning a three-dimensional maquette usually result in fairly large files, and they are sometimes used for animated feature films.

Free-form modeling—also called free-form deformation—is

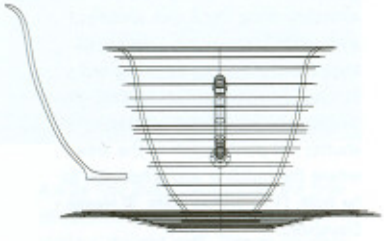


4.4.2b The outline of the different floors were defined on the XY (top) plane, and the resulting outline was extruded along the Z axis. Details were added later using a variety of techniques including trimming and Boolean operations. The finished rendering was composited with an open shutter still photograph of the area where the building was scheduled to be built. (110 Bishop Gate. © Hayes Davidson.)



4.4.3 Fluid shapes can be created with free-form sweeping, which is based on an extrusion along a complex path and a multitude of axes. These wolves have a primitive quality that makes them very different from both the cartoon and the realistic styles of modeling. (Pastilles Vichy. Agency: EURO RSCG BETC. Director: Pierre Coffin. Illustrator: Jean-Christophe Saurel. Production and Computer Graphics: Ex Machina. Images courtesy of Ex Machina.)

used when other modeling techniques are too rigid for building a specific scene or when using a combination of other tools would get the job done but would also require additional production time and a larger production budget. Free-form objects are also sometimes used because of the creative preferences of the individuals who design the look of the three-dimensional scene. Figure 4.5.2 illustrates a free-form technique that uses a simple polygonal mesh to generate and control a curved surface. In this case the mesh is created by extruding new polygons and welding them to one another. A flat polygon is extruded to create the basic torso, and the points are pulled to refine the shape (A). The breast shape is created by pulling points on a lathed spline (B and C). Extra edges on some torso polygons are created to weld to it the 8-sided breast object, which is duplicated and attached by welding vertices and deleting the interior faces (D and E). The bottom torso polygons are extruded to create the pelvis area, and the vertices are sculpted (F). The bottom facets are sliced, extruded, and sculpted to create the legs (G and H). Additional facets are created on the back by dividing the edges of some polygons and rounding the mesh manually, and the arms are created by forcing some edges on the sides of the torso into a 5-sided shape, extruding them, and moving points to sculpt detail (I, J, and K). The torso area is sliced to create a navel indentation, the faces on the bottom of the body are extruded downwards and adjusted to create the legs (L). The feet and hands are created by extruding down the bottom ankle, extruding some of the front faces, and creating additional polygons to define the toes and fingers (M and N). More detail is sculpted with additional polygons, and finally the spline surfaces are generated from the polygonal mesh (O). The finished model can be seen in Figure 10.5.11.



SWEEPING A 2D OUTLINE

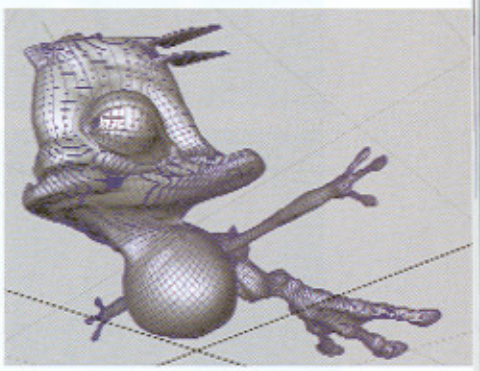
Free-form modeling techniques can also be used in conjunction with other techniques—especially those described in Chapter 5. The technique of free-form modeling has a couple of variations including direct point manipulation, deformation with lattices, and terrains.

4.4.4 This cup was modeled by sweeping a two-dimensional outline 360 degrees around the Y axis (opposite page), and adding a handle later. (Courtesy of Iris Benado.)

Virtual Sculpting with Polygonal Meshes

The most common and easiest way to create three-dimensional free-form shapes usually starts with an existing three-dimensional structure that is to be "sculpted" and transformed into the desired free-form object. This **virtual sculpting** process is, in essence, quite similar to the process of modeling fresh clay with one's hands. The initial shape of the clay is fairly unimportant in terms of the desired final three-dimensional shape. But as our hands massage, push, pull, and rub the shapeless clay, it is slowly transformed into a meaningful structure. A simple primitive like a sphere can be the starting point, or we can start from a three-dimensional scan of a real scale model. Some techniques for sculpting with free-form curved surfaces and for fitting curved surfaces to polygonal meshes are described in Chapter 5.

The **direct point manipulation** modeling process, as virtual sculpting is also known, starts by identifying the points—or control vertices—in the wireframe structure that can be displaced in three-dimensional space. Most three-dimensional modeling programs offer simple ways of picking a single point or a group of points. Direct point manipulation in the case of curve lines can be done directly to a point on the curve or to a control or a tangent point. Usually the selection is done by just clicking and dragging one or several points. Once the points have been selected they can be dragged in any direction. Some programs offer excellent tools for picking and manipulating single points, while similar capabilities in other programs leave a lot to be desired. Some of the most useful direct point manipulation options include the ability to select several points that are not contiguous, or the ability to lock the position of points in some parts of the object while other points are being manipulated. Figures 4.5.1 and 4.5.2 illustrate the result of free-form modeling.



4.5.1 This wireframe character was created with NURBS patches, including simple free-form modeling techniques that pull and push individual points in a mesh. Simple meshes like spheres are commonly used as virtual clay in free-form modeling both because they are easy to create and because they offer a good number of points to work with. (© 2003 Oddworld Inhabitants, Inc. All rights reserved.)

Deformation with Lattices

Direct point manipulation can be a very efficient technique when only a few points need to be manipulated or when the user is really skilled at free-form sculpting. There is another free-form modeling technique that can be more appropriate for the task than direct point manipulation—especially in cases where a uniform global deformation is desired or when the user does not have the skill or the time to manipulate a large number of points one at a time. This technique is called **deformation with lattices**. (Lattices are sometimes called

bounding boxes, not to be confused with the boundary boxes described later in this chapter.)

A lattice is a structure of points and lines that controls the points in the model. We can think of the lattice as a structure of grids that is connected to the points in the model with imaginary springs. Therefore, when the grids—or points on the grids—are moved, they drag the object's points with them (Fig. 4.5.3).

Every point on the lattice is connected to one or several of the model's points. The ability to control the deformation of the object by moving one or several grid points in the lattice depends directly on the number of lattice points. A small number of points on the lattice results in very rough or global distortion. Lattices with a large number of control points can be used to apply very subtle—or local—distortion on the object controlled by the lattice.

4.5.2 This Animé-inspired character is an example of straightforward modeling with the Mesh Smooth tools in 3D Studio MAX. Keeping a low polygonal count was as important as maintaining all polygons four-sided to create this human body. (© 1999 Michael B. Comet. All rights reserved.)

Simple Terrains and Functions

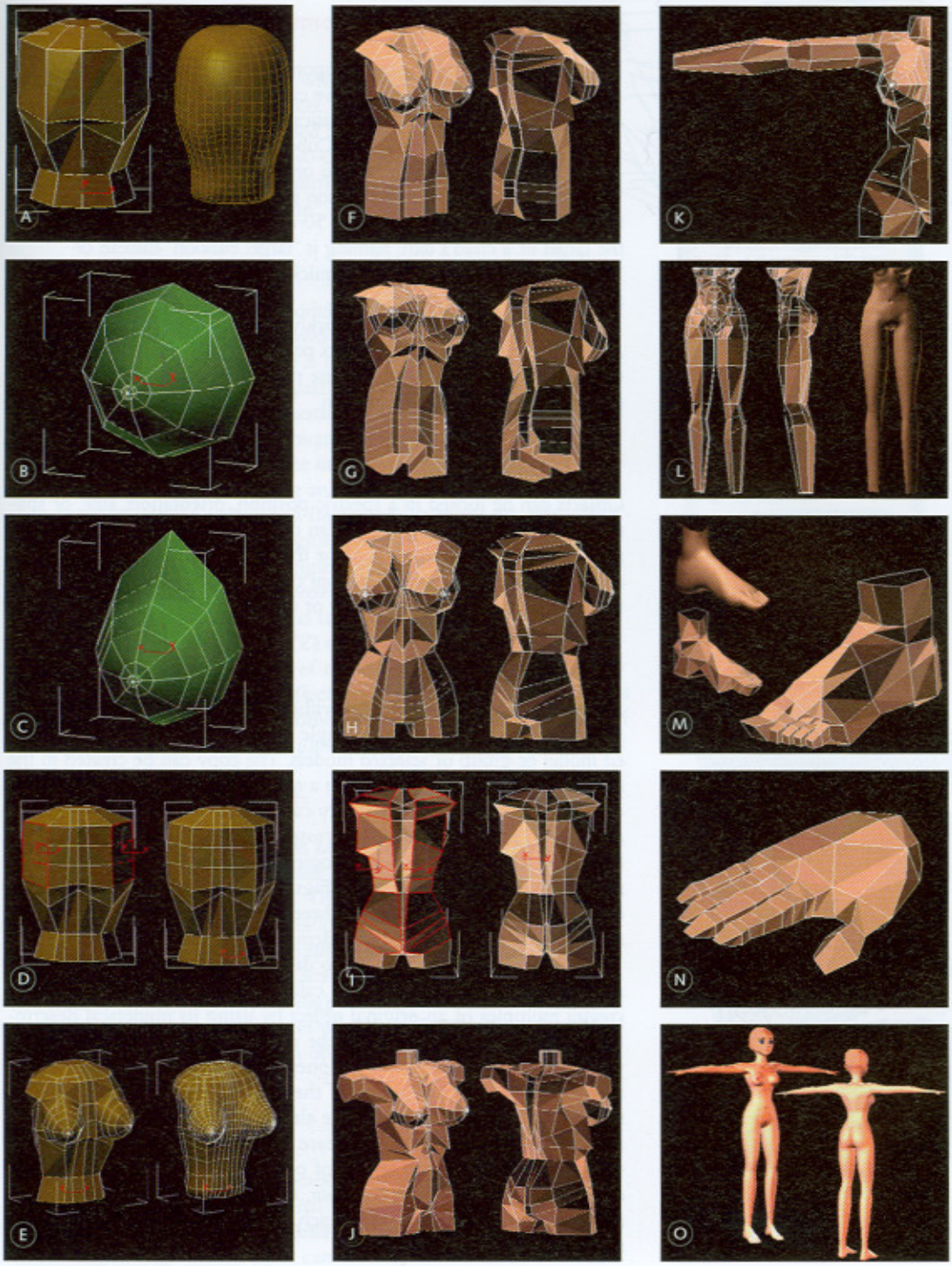
A great variety of techniques is available for creating **terrains** that simulate or recreate natural or imaginary landscape surfaces. A great variety of other techniques use **mathematical functions** for distorting those terrains.

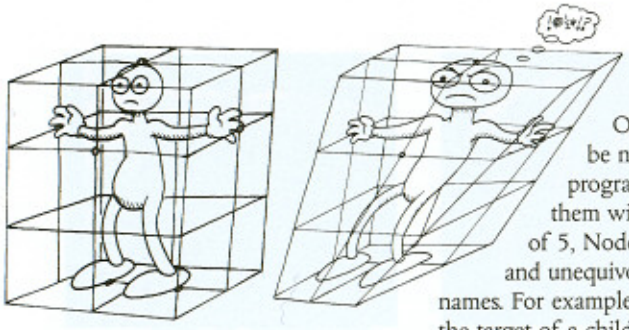
The simplest technique for creating a terrain consists of using a flat two-dimensional plane with XY subdivisions. Obviously, the more subdivisions on a plane, the more detail will appear on the final terrain model. As mentioned earlier, the position of points on the plane can be modified by direct point manipulation or by lattice deformation. Either of these techniques would be appropriate if the shape was supposed to resemble a natural terrain. But if you are trying to create a more fantastic terrain, the basic plane could be deformed with a mathematical function. Figure 4.5.4 shows terrains created by distorting a terrain with different functions.

Another technique for creating terrains consists of building a three-dimensional mesh based on two-dimensional contours that define an imaginary or real landscape. This technique is very data-intensive but also very efficient for creating accurate models of terrains. Because of their topological detail, terrains created with this technique are rarely distorted with mathematical functions (Fig. 5.1.4). A simpler technique for quickly building terrains consists of applying a black-and-white image as a displacement map to a three-dimensional flat plane (Fig. 9.6.2).

4.6 Basic Modeling Utilities

In addition to basic modeling tools, virtually all three-dimensional modeling programs offer a set of basic utilities meant to complement the modeling process. Among them we find such useful techniques as naming objects and getting information about them, duplicating, snapping to grid, mirroring, displaying as a bounding box, calculating volumes, and creating text.


A
F
K
B
G
L
C
H
M
D
I
N
E
J
O



4.5.3 These drawings illustrate the effect of lattices on the shape of a three-dimensional object. Every time the lattice is moved the model is deformed because each of the grid points on the lattice is connected with imaginary springs to the object's points.

Getting Information and Naming Objects

Objects and components of objects in some cases can be named so that we can identify them faster. Many programs will automatically name objects as we create them with names like Cube 1, Cube 2, Cube 3, or Node 1 of 5, Node 2 of 5, and so on. But in some cases where quick and unequivocal identification is required it is best to use unique names. For example, when one of 50 ellipsoids representing balloons is the target of a child's dart, naming it "target balloon" instead of "Ellipsoid 37" could be useful to quickly identify it during the explosion sequence (Fig. 4.6.1).

Another useful feature for quickly assessing detailed information about a specific object—such as its position, dimensions, and orientation—is the Get Information feature that presents information about the active object in numerical form.

Locking

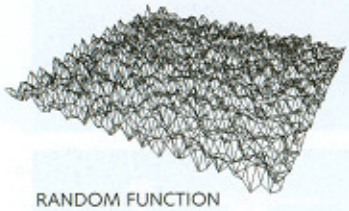
Objects can be locked in a specific position, orientation, size, or spatial range. **Locking** an object or an object's element that is not supposed to move can help streamline the modeling process. In most cases, objects can be switched from the locked position to the unlocked position without losing or modifying any other attributes.

Duplicating and Instancing

Models can be easily duplicated without having to build them from scratch. **Duplicating** creates a single independent copy of the selected model or group of selected models. The copy can be created in the same location as the original or in a new position defined by an XYZ offset value. The duplicating utility can also create multiple copies of an object. The values needed to create multiple copies of an object typically include the number of copies, as well as the XYZ values for translation, rotation, and scaling (Figs. 4.6.2 and 4.6.3). Creating copies of an object creates more three-dimensional elements in the scene, increases the file size, and demands more computing time.

Instancing is an alternative to duplicating that is available in many systems. **Instancing**—also called **cloning** in some systems—creates multiples of an original object by using its numerical description and cloning it elsewhere in the scene. The multiples created with instancing are like "living clones" that continue to be related at all times to the original object. If the original changes shape or is scaled, its dependent instances are also transformed. Because instances of a model do not increase the size of a file they are convenient for creating large armies of objects that look alike and that display a consistent group behavior. Instancing, however, may not be appropriate if a project requires each multiple to undergo a different shape transformation.

STARR FUNCTION



RANDOM FUNCTION

4.5.4 A plane terrain with a resolution of 40 × 40 units is deformed with different mathematical functions: a Starr function, a random function, and on the opposite page versions of the Julia and Mandelbrot fractal functions.

Setting a Face

Two-dimensional outlines drawn with free-form or curve tools are not really three-dimensional objects. When they are first drawn, two-dimensional outlines that are closed are just lines with a hole in the middle. Therefore, in order for two-dimensional outlines to be rendered properly it is necessary to turn them into planes. This process is called **setting a face** to an outline.

Mirroring

Mirroring a three-dimensional model is a useful technique when building an object composed of two identical (or almost identical) halves. The **mirroring** technique is implemented in a number of ways by different software, each with particular requirements and subtle functional differences. Three-dimensional objects can be repositioned in space with the mirroring technique. In such cases the object is entirely moved to where its mirror image would be. In general, however, mirroring works by copying an object, placing the copy in the same location as the original, and finally repositioning the copy. This way, the object remains in its original position, and its copy is placed where the mirror image of the original object would be.

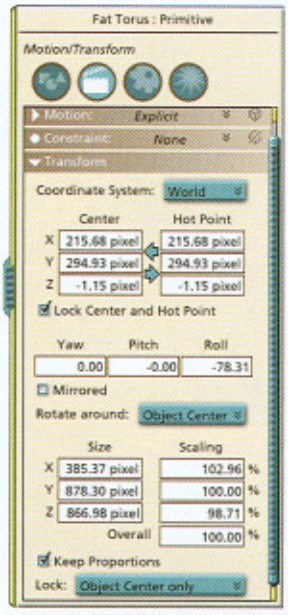
Mirroring works by either providing a scaling value of -1 along the axis on which the mirroring is to take place, specifying a two-dimensional plane (XY, XZ, or YZ) on which the object is to be mirrored, or by establishing an axis of reflection by clicking a line perpendicular to the object to be reflected (one end of the line represents the base point of reflection, the other end represents the beginning of the axis reflection). Mirroring is illustrated in Figures 4.6.2 and 4.6.4.

Setting the Center of Objects

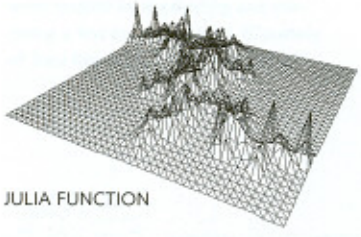
Most three-dimensional programs keep track of where the centers of objects are placed. By default, these centers are automatically placed in the objects' geometric centers. These points become very important—especially during the animation process because many operations are calculated based on the spatial position of the center of the object. These operations include scaling, rotation (global and local—the latter is also known as pivoting), linking, and simulations of motion dynamics related to center of gravity. Being able to interactively reposition the center of an object is a powerful modeling and animation utility (Fig. 4.6.1).

Snapping to the Grid

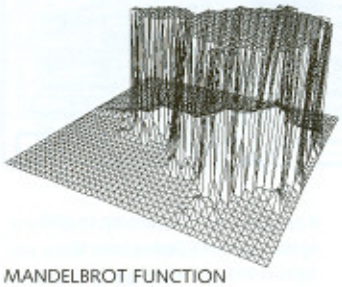
By forcing the object's or its components' points, for example, to **snap to a grid**, three-dimensional modeling programs can help to simplify the construction of regular shapes or precise details within larger shapes (Fig. 4.6.3). Grids can usually be defined by the user.



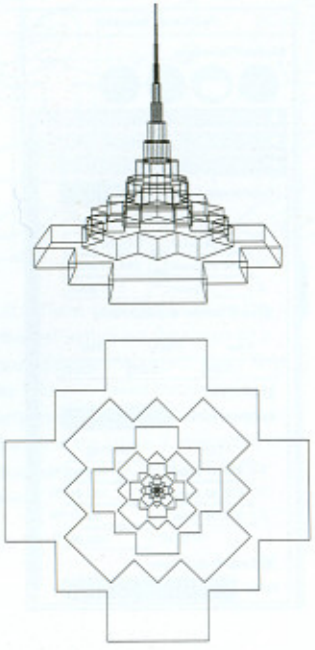
4.6.1 This dialog box provides quick numerical information about the position, orientation, size, and ranges of motion of an object. (Carrara dialog box © 2001–2003 Eovia Corporation.)



JULIA FUNCTION



MANDELBROT FUNCTION



4.6.2 This object was created by duplicating the original shape at the bottom 12 times. Each instance of the original shape was mirrored by rotating 45 degrees and translating one unit on the Z axis.

This includes the size of the grid unit, whether the points snap to the grid, whether the object's center or edges snap to the grid when the object is moved, and whether the snap to grid function is applied to all objects in the scene or only to some.

Setting Text

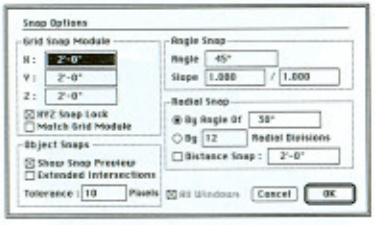
The text tool is capable of automatically producing two-dimensional outlines or three-dimensional objects extracted from the two-dimensional outlines of fonts (or typefaces) installed in the computer system. The sophistication and variety of two-dimensional text outlines varies greatly from software to software, and so do all the additional features associated with letterforms, such as letterspacing, kerning capabilities, and point-editing features (Fig. 4.6.5).

Most of today's three-dimensional modeling programs extract the text outlines from spline-based descriptions (often in the PostScript language) resident in the system software. Some three-dimensional programs extract this outline information from a font database—sometimes in curve format, other times in polygonal format—that is provided with the three-dimensional program itself. The shapes of the letterforms are usually smooth and detailed when the outline information is brought in as a series of curves. However, when the outline information is brought in as a series of polygonal lines, the resulting shape may be jagged and unrefined, especially in the portions of the outline with the most curvature.

Most text tools work in conjunction with the keyboard. Any character that can be typed from the keyboard will show up in the three-dimensional environment as a two-dimensional outline. When an extrusion value is specified for the two-dimensional outline, the letterform can be three-dimensional. As with all objects modeled with extrusion, bevelling can be applied to letterforms modeled with extrusion-based text tools (Fig. 5.4.1–5.4.2). (See Chapter 5 for additional information on bevelling.)

Volume Calculation

The calculation of volumes and unfolding of planes are two modeling specialized techniques that can be useful when designing three-dimensional models that eventually get fabricated out of real materials. Volume calculation tools allow users to find out the total volume and area of the inside, outside, or parts of any three-dimensional object. Knowing the exact volume of liquid that can be contained in a new bottle design can be very important to an industrial designer. Likewise, an engineer in charge of supervising the actual production of the bottle needs to know the volume of glass needed to fabricate the bottle. Some of the volume calculation tools can also be used to extract data related to the object's mass or its center of gravity and inertia. This information can be used later in the animation of models using motion dynamics animation software.



4.6.3 Controls for the snap to grid option. (form•Z Dialog box. © 1991–1995 auto•des•sys, Inc.)

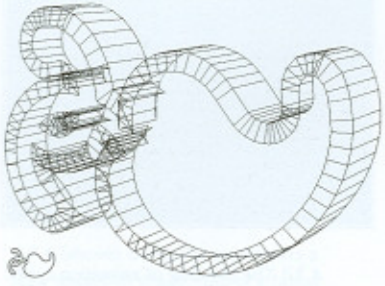
Being able to unfold the plans that bound a three-dimensional object can be quite useful when it is necessary to fabricate either a cardboard scale model or prototype of the three-dimensional object or the final object itself in more durable materials, such as plastic or sheet metal. Figure 15.8.1 shows three-dimensional objects that were built with a variety of modeling techniques and then unfolded into two-dimensional patterns.

Bounding Box

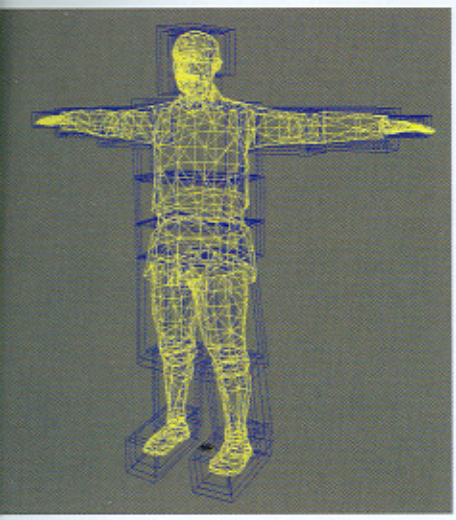
When modeling a scene with multiple complex objects many computer systems may slow down because of the huge number of calculations needed to redraw the image of the models on the screen. Using bounding boxes to represent objects is a convenient technique for speeding up their display. **Bounding boxes** are usually rectangular, and they are defined by the points most distant from the center of the model. Bounding boxes can also be used to define the collision volumes in a videogame (Fig. 4.6.6) or a dynamics simulation (Fig. 13.3.1). Bounding boxes are not to be confused with lattices, which can be used to distort the three-dimensional objects contained within them. Bounding boxes and the lattices used to deform free-form objects look similar but behave differently.

Making objects invisible or **ghosting** them are two options similar to the bounding box. Making objects invisible removes them from the display but not from the information contained in the file. Objects made invisible with this method are usually not displayed regardless of the rendering method until made visible again—usually by just clicking a choice in a checkbox. The ghosting option offered by some programs is similar to the bounding box. In some implementations of the ghosted display the model is represented with dotted lines, and the display of the ghosted model is only updated when the mouse button is released at the end of an interactive manipulation. In other implementations of ghosting it does not speed the display of images on the screen but instead facilitates working with complex models by making ghosted portions unselectable.

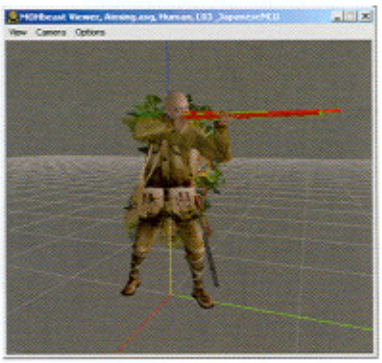
4.6.4 The components in this engine were created by duplicating and mirroring a few custom objects. (Courtesy of Toru Kosaka, STUDIO EggMan.)



4.6.5 Two-dimensional text outlines can easily be converted into three-dimensional type.



4.6.6 The collision detection volumes in dark blue define the areas of vulnerability of a soldier in the game *Medal of Honor*, and trigger hit reactions. The larger boxes are used for the easy playing mode while the smallest boxes offer a greater degree of difficulty. (© 1999 Electronic Arts Inc. All rights reserved.)



4.7.1 The blending of animation cycles as calculated by the game engine is previsualized on a PC in real-time with the *MOH Beast* utility. (© 1999 Electronic Arts Inc. All rights reserved.)

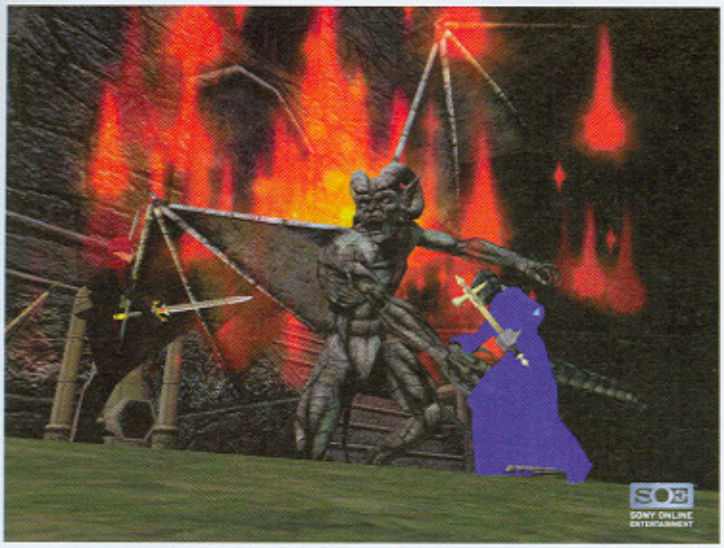
4.7 Real-Time Polygonal Models

The ability to render polygonal models in real-time is almost as old as three-dimensional computer animation, but today's systems offer more realistic rendering and more complex models than what was possible a decade ago. Gaming is the area that takes the most advantage of real-time rendering, whether it takes place on game platforms or computers. Interactive online websites with three-dimensional navigation and functionality also take advantage of polygonal models rendered in real-time. Both gaming and online websites require an approach to modeling for real-time that emphasizes efficiency and portability.

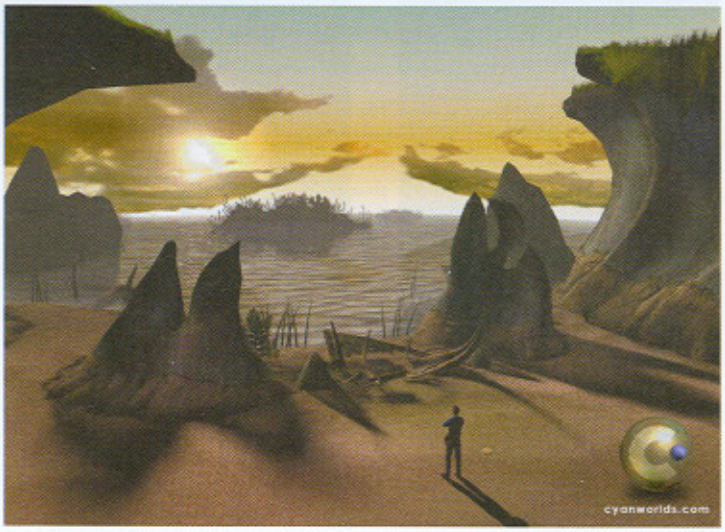
Improved hardware rendering, as explained in Chapter 6, has fueled the feasibility and popularity of ever more complex real-time polygonal models. Powerful graphics cards and GPUs, faster clock speeds, and increased memory continue to raise the bar for real-time polygonal models. But there are always limits to what a powerful graphics system can do, and for that reason it is important to **optimize polygonal models** for real-time rendering. Designing and building polygonal models and their textures for real-time rendering is about compromise and optimization (Fig. 4.7.6). There is always a compromise between visual detail and speed in a real-time environment. Real-time models may range from a few hundred polygons to thousands of polygons. *Spyro the Dragon*, for example, has 352 polygons (Figs. 3.6.4, 10.5.10 and 11.1.1) while *Dawn* (Fig. 6.10.1) is built with 203,741 triangles. A "heavy" model with too much geometrical complexity could slow down the real-time rendering and the frame rate would drop, lessening the illusion of natural motion. Same would be true for a model with image maps, textures, too large for the available memory, graphics card, or GPU in question. Keep in mind that the geometry and size of image maps that might be optimal for one environment might be overkill or not enough for another. This is a common challenge when "porting" or adapting a computer game from one platform to another, from platform to PC or vice versa: models and image maps sometimes require significant revision and optimization.

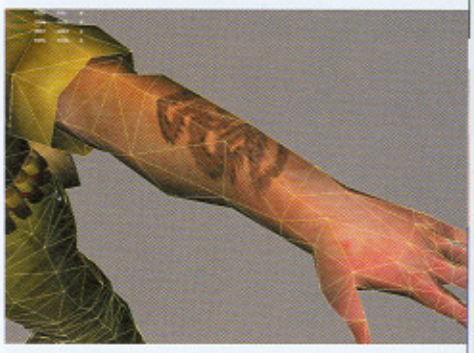
Most computer or platform games use their own custom **rendering engine** to render polygonal models in real-time (Fig. 4.7.1). The engine is fed with highly optimized polygonal and image map information. Because different game engines use a variety of different rendering and animation techniques the polygonal model and image maps formats may vary between engines (Figs. 3.6.4, 4.7.5, 7.4.6, and 11.5.4). For this reason it is always a good idea to keep a lowest-common denominator model that can be adapted and converted for different game platforms and engines.

Most online websites that use real-time polygonal models require a **player** or plug-in. Most multiplayer gaming websites use their own formats and engines (Figs. 4.7.2–4.7.3, and 4.7.8). But a wide variety of off-the-shelf three-dimensional players for real-time rendering is also available. These can usually be downloaded to the



4.7.2 The multiplayer game *EverQuest* has characters ranging between 500 and 3,000 polygons, a minimum of two 256 x 256 textures, and bump maps. (EverQuest® courtesy Sony Online Entertainment Inc. EverQuest is a registered trademark of Sony Computer Entertainment America Inc. © 1999-2002 SCEA Inc. SOE and the SOE logo are registered trademarks of Sony Online Entertainment Inc. All rights reserved.)

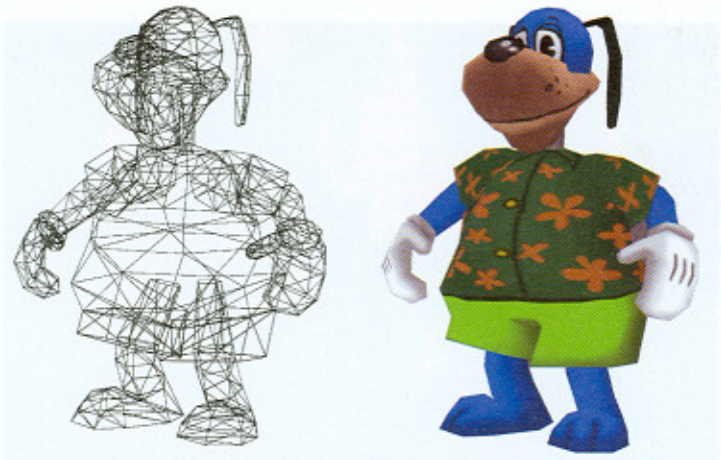




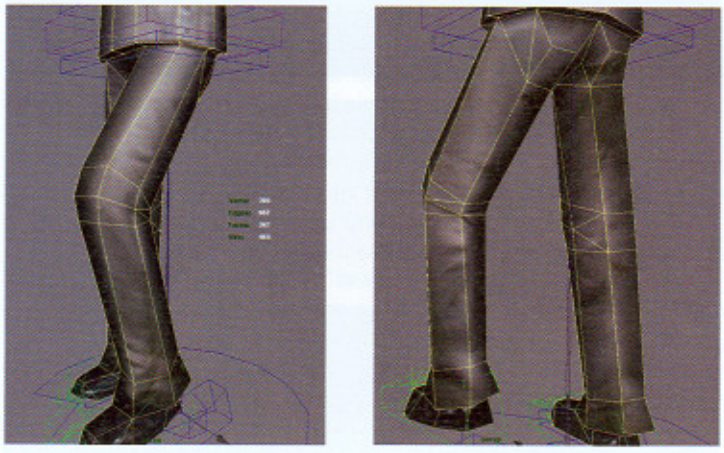
4.7.3 *URU, Ages Beyond Myst* (left) is the multiplayer online version of *Myst*. The high-resolution avatar is roughly 5,000 polygons, but level of detail (LOD) techniques are used to lower its resolution for long shots. The environment is several thousand polygons, but the total may surpass 15,000 polygons when the degree of tessellation of the water increases due to ripple and flow. (© 2002 Cyan Worlds, Inc. All rights reserved.)

4.7.4 (above) Using UV coordinates a high resolution image map of a tattoo is placed on a dozen polygons on the arm. (© 1999 Electronic Arts Inc. All rights reserved.)

user's computer from the World Wide Web, sometimes free of charge. Some of the most popular off-the-shelf formats and players include Shockwave 3D, Viewpoint, Cult 3D, Wild Tangent, Pulse 3D, and more recently MPEG-4. This file format is used primarily for encoding moving images, but it also provides tools for encoding three-dimensional polygonal meshes that can be used for low resolution real-time character facial and body animation. MPEG-4 uses visual lip configurations, called **visemes**, that are equivalent to speech phonemes. Three-dimensional files in general may include compressed geometry, connectivity, shading normals, color, and texture coordinates.



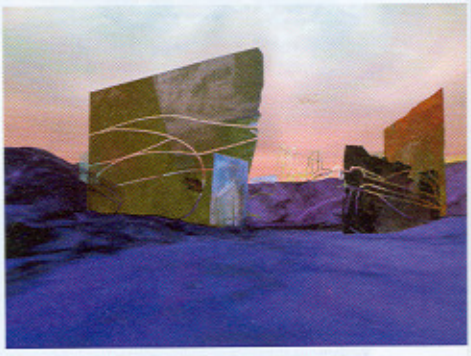
4.7.5 Sample character in wireframe and shaded versions from Disney's *Toontown* Online multiplayer game. Model is within the 1,000 polygon average, including a 392-polygon head, 476-polygon torso, and 158-polygon legs. (© Disney Enterprises, Inc.)



4.7.6 This polygonal mesh has been designed to optimize the folding of the pants at the knees. (© 1999 Electronic Arts Inc. All rights reserved.)

4.7.7 First-level playground in the *Toontown* multiplayer game, with two avatars that players can create interactively from a catalog of parts (torso, legs, head, muzzle) and clothes. (© Disney Enterprises.)

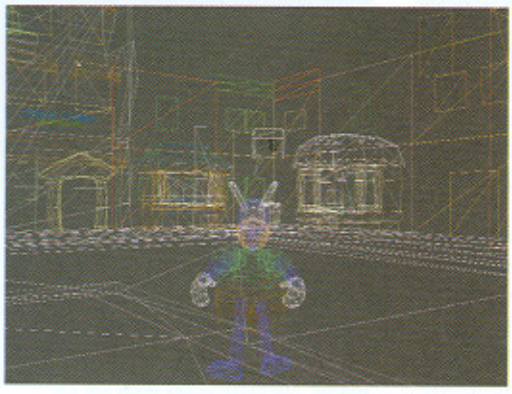
In addition to using powerful graphics cards or simplifying the geometry and the image maps, there are a few tricks and shortcuts that can be used to accelerate real-time rendering. A few of the most common are levels of detail, billboards, and Flash animations. Levels of detail are useful in situations where the same three-dimensional character or prop appears very close and very far from the camera. Levels of detail, also called LODs, consist of having the same model at different resolutions that can be seamlessly loaded as the object or character moves closer or away from the camera (Fig. 3.6.7). Image maps with different levels of detail can also be used to highlight details such as the tattoo in Figure 4.7.4. Billboards are a few polygons that define a flat surface used to project image maps. Billboards have the same function that painted backgrounds have on a theatre stage or a movie set (Figs. 4.7.7 and 4.7.9). (See Chapter 9 for more information on billboards for visual effects.) Using Flash MX files to deliver three-dimensional animations online is another common shortcut. In this case the three-dimensional animation is rendered



and saved as a two-dimensional SWF file or Flash movie. The advantages of this approach, or "cheat," are that Flash files are generally compact and download fast, and also that the Flash player is ubiquitous worldwide. Three-dimensional animations saved in the SWF format are usually contained in small windows, 200 × 200 pixels for example, that can be imported into a Flash movie and easily played within larger scenes.

Computer and platform games are the most pervasive applications of real-time rendering, but certainly not the only one. Fine artists are increasingly creating interactive worlds and virtual reality installations using off-the-shelf players or repurposed game engines; the latter approach is called **game moding**. Figure 12.7.3 shows a virtual reality installation that requires users to wear a VR helmet. Figure 4.7.8 shows *Purbeck Light Years*, a real-time computer animation and immersive virtual reality based on paintings and drawings. The land mesh in this installation was created from a satellite image of Dorset, England, and the grayscale values used as a displacement map on a 8,192-polygon terrain. The ground and sky are covered with bitmap textures. To simulate day or nighttime the ground and sky textures blend gradually into other textures. The skies are chosen at random to represent the unpredictable nature of real weather, and during the daytime there is a 30% chance of rain. The textures for the rectangular vertical planes, or billboards, are from drawings and paintings of Corfe Castle and an alpha transparency channel is used to display their irregular shapes. A flocking system governs the movement of birds, a theatrical element in the project. A polar coordinate system (Fig. 3.4.8) is used to keep the castle in the middle of the scene at all times.

4.7.8 In addition to the 8,192-polygon terrain in *Purbeck Light Years* there are 128 models in this Shockwave 3D world, each consisting of 2 polygons for a world total of 8,448 triangles. A higher resolution model, 131,072 polygons, was used to capture these images. The castle image maps are 32-bit color and 1024 × 512 pixels, and the maps for the planes, ground and sky are 512 × 512 pixels. (© 2003 Jeremy Gardiner. Programming by Anthony Head, research by Veronica Falçao.)



4.7.9 Wireframe and shaded versions of a corner in *Toontown*, where real-time environments typically range between 2,000 and 3,000 polygons excluding characters. The Panda3D proprietary engine can display geometry in real-time using either OpenGL or DirectX. (© Disney Enterprises, Inc.)

CHAPTER 4

Key Terms

Arcs
B-spline
Bézier curve
Bounding boxes
Capping
Cardinal spline
Children
Circles
Cloning
Cones
Control points
Control vertices
Controlled curves
Cubes
Curvature
Curved lines
Cylinders
Deformation with lattices
Degree of curve
Diameter
Die
Direct point manipulation
Dodecahedron
Duplicating
Extrusion
Free-form extrusion
Free-form modeling
Free-form three-dimensional objects
Game moding
Geometric primitives
Ghosting
Icosahedron
Instancing
Knots
Lathe
Linear spline
Locking
Lofting
Mathematical functions
Mirroring
Nonrational curve
Non-uniform rational b-splines
NURBS
Octahedron
Optimize polygonal models
Parents
Player, plug-in
Polygonal lines
Polygons
Radius
Rational curve
Regular polyhedra
Rendering engine
Revolve
Seed outline
Setting a face
Snap to grid
Spheres
Spirals
Splines
Surfaces of revolution
Sweeping
Tangent points
Tension
Terrains
Tetrahedron
Torus
Two-dimensional shapes
Virtual sculpting
Visemes
Weight