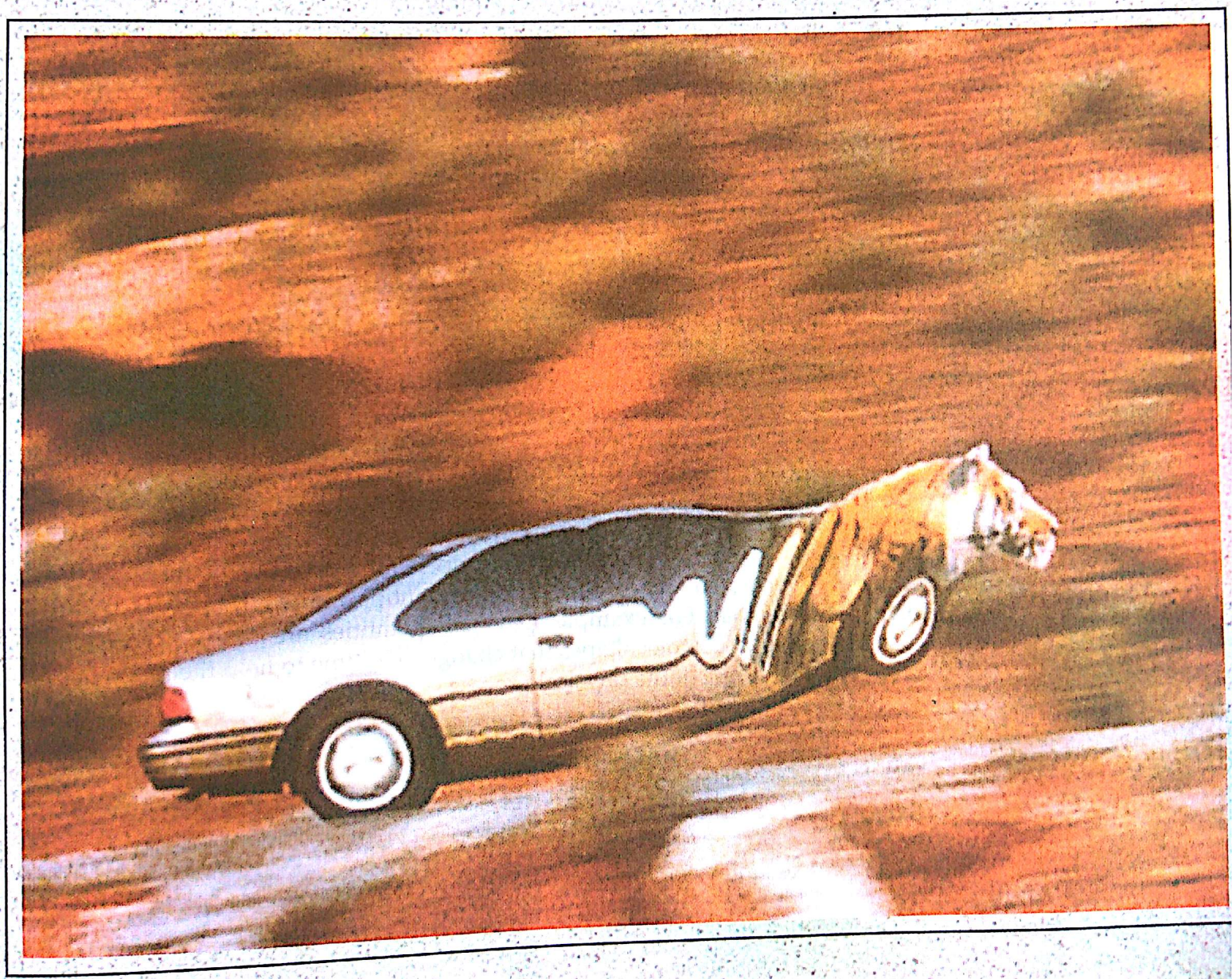
CHAPTER

16 Computer Animation



Some typical applications of computer-generated animation are entertainment (motion pictures and cartoons), advertising, scientific and engineering studies, and training and education. Although we tend to think of animation as implying object motions, the term **computer animation** generally refers to any time sequence of visual changes in a scene. In addition to changing object position with translations or rotations, a computer-generated animation could display time variations in object size, color, transparency, or surface texture. Advertising animations often transition one object shape into another: for example, transforming a can of motor oil into an automobile engine. Computer animations can also be generated by changing camera parameters, such as position, orientation, and focal length. And we can produce computer animations by changing lighting effects or other parameters and procedures associated with illumination and rendering.

Many applications of computer animation require realistic displays. An accurate representation of the shape of a thunderstorm or other natural phenomena described with a numerical model is important for evaluating the reliability of the model Also, simulators for training aircraft pilots and heavy-equipment operators must produce reasonably accurate representations of the environment. Entertainment and advertising applications, on the other hand, are sometimes more interested in visual effects. Thus, scenes may be displayed with exaggerated shapes and unrealistic motions and transformations. There are many entertainment and advertising applications that do require accurate representations for computer-generated scenes. And in some scientific and engineering studies, realism is not a goal. For example, physical quantities are often displayed with pseudo-colors or abstract shapes that change over time to help the researcher understand the nature of the physical process.

16-1 DESIGN OF ANIMATION SEQUENCES

In general, an animation sequence is designed with the following steps:

- Storyboard layout
- Object definitions
- Key-frame specifications
- Generation of in-between frames

This standard approach for animated cartoons is applied to other animation applications as well, although there are many special applications that do not follow this sequence. Real-time computer animations produced by flight simulators, for instance, display motion sequences in response to settings on the aircraft controls. And visualization applications are generated by the solutions of the numerical models. For *frame-by-frame animation*, each frame of the scene is separately generated and stored. Later, the frames can be recorded on film or they can be consecutively displayed in "real-time playback" mode.

The *storyboard* is an outline of the action. It defines the motion sequence as a set of basic events that are to take place. Depending on the type of animation to be produced, the storyboard could consist of a set of rough sketches or it could be a list of the basic ideas for the motion.

An *object definition* is given for each participant in the action. Objects can be defined in terms of basic shapes, such as polygons or splines. In addition, the associated movements for each object are specified along with the shape.

A *key frame* is a detailed drawing of the scene at a certain time in the animation sequence. Within each key frame, each object is positioned according to the time for that frame. Some key frames are chosen at extreme positions in the action; others are spaced so that the time interval between key frames is not too great. More key frames are specified for intricate motions than for simple, slowly varing motions.

In-betweens are the intermediate frames between the key frames. The number of in-betweens needed is determined by the media to be used to display the animation. Film requires 24 frames per second, and graphics terminals are refreshed at the rate of 30 to 60 frames per second. Typically, time intervals for the motion are set up so that there are from three to five in-betweens for each pair of key frames. Depending on the speed specified for the motion, some key frames can be duplicated. For a 1-minute film sequence with no duplication, we would need 1440 frames. With five in-betweens for each pair of key frames, we would need 288 key frames. If the motion is not too complicated, we could space the key frames a little farther apart.

There are several other tasks that may be required, depending on the application. They include motion verification, editing, and production and synchronization of a soundtrack. Many of the functions needed to produce general animations are now computer-generated. Figures 16-1 and 16-2 show examples of computer-generated frames for animation sequences.

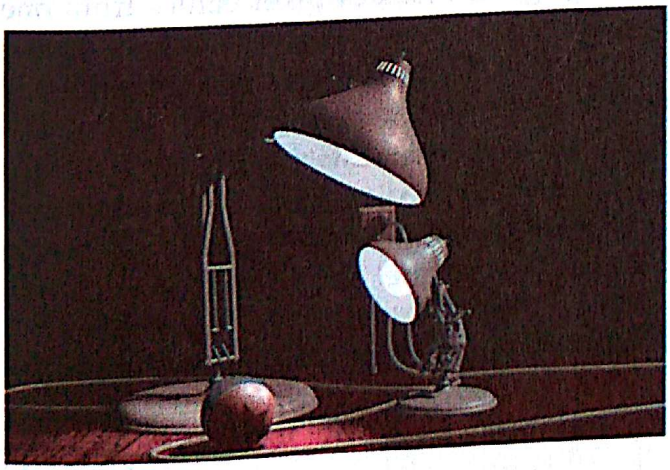


Figure 16-1
One frame from the award-winning computer-animated short film *Luxo Jr.* The film was designed using a key-frame animation system and cartoon animation techniques to provide lifelike actions of the lamps. Final images were rendered with multiple light sources and procedural texturing techniques. (*Courtesy of Pixar. © 1986 Pixar.*)

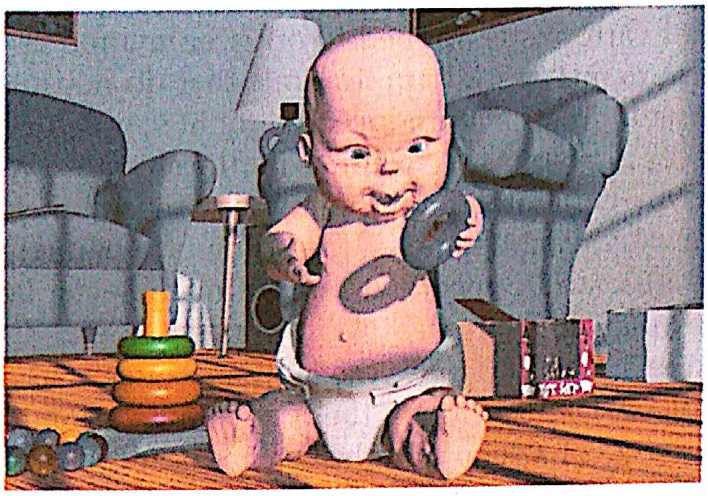


Figure 16-2
One frame from the short film *Tin Toy*, the first computer-animated film to win an Oscar. Designed using a key-frame animation system, the film also required extensive facial expression modeling. Final images were rendered using procedural shading, self-shadowing techniques, motion blur, and texture mapping. (*Courtesy of Pixar. © 1988 Pixar.*)

16-2
GENERAL COMPUTER-ANIMATION FUNCTIONS

Some steps in the development of an animation sequence are well-suited to computer solution. These include object manipulations and rendering, camera motions, and the generation of in-betweens. Animation packages, such as Wavefront, for example, provide special functions for designing the animation and processing individual objects.

One function available in animation packages is provided to store and manage the object database. Object shapes and associated parameters are stored and updated in the database. Other object functions include those for motion generation and those for object rendering. Motions can be generated according to specified constraints using two-dimensional or three-dimensional transformations. Standard functions can then be applied to identify visible surfaces and apply the rendering algorithms.

Another typical function simulates camera movements. Standard motions are zooming, panning, and tilting. Finally, given the specification for the key frames, the in-betweens can be automatically generated.

16-3
RASTER ANIMATIONS

On raster systems, we can generate real-time animation in limited applications using *raster operations*. As we have seen in Section 5-8, a simple method for translation in the xy plane is to transfer a rectangluar block of pixel values from one location to another. Two-dimensional rotations in multiples of 90° are also simple to perform, although we can rotate rectangular blocks of pixels through arbitrary angles using antialiasing procedures. To rotate a block of pixels, we need to determine the percent of area coverage for those pixels that overlap the rotated block. Sequences of raster operations can be executed to produce real-time animation of either two-dimensional or three-dimensional objects, as long as we restrict the animation to motions in the projection plane. Then no viewing or visible-surface algorithms need be invoked.

We can also animate objects along two-dimensional motion paths using the *color-table transformations*. Here we predefine the object at successive positions along the motion path, and set the successive blocks of pixel values to color-table

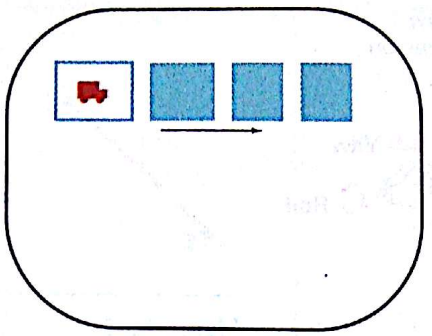


Figure 16-3
Real-time raster color-table animation.

entries. We set the pixels at the first position of the object to "on" values, and we set the pixels at the other object positions to the background color. The animation is then accomplished by changing the color-table values so that the object is "on" at successively positions along the animation path as the preceding position is set to the background intensity (Fig. 16-3).

16-4 COMPUTER-ANIMATION LANGUAGES

Design and control of animation sequences are handled with a set of animation routines. A general-purpose language, such as C, Lisp, Pascal, or FORTRAN, is often used to program the animation functions, but several specialized animation languages have been developed. Animation functions include a graphics editor, a key-frame generator, an in-between generator, and standard graphics routines. The graphics editor allows us to design and modify object shapes, using spline surfaces, constructive solid-geometry methods, or other representation schemes.

A typical task in an animation specification is *scene description*. This includes the positioning of objects and light sources, defining the photometric parameters (light-source intensities and surface-illumination properties), and setting the camera parameters (position, orientation, and lens characteristics). Another standard function is *action specification*. This involves the layout of motion paths for the objects and camera. And we need the usual graphics routines: viewing and perspective transformations, geometric transformations to generate object movements as a function of accelerations or kinematic path specifications, visible-surface identification, and the surface-rendering operations.

Key-frame systems are specialized animation languages designed simply to generate the in-betweens from the user-specified key frames. Usually, each object in the scene is defined as a set of rigid bodies connected at the joints and with a limited number of degrees of freedom. As an example, the single-arm robot in Fig. 16-4 has six degrees of freedom, which are called arm sweep, shoulder swivel, elbow extension, pitch, yaw, and roll. We can extend the number of degrees of freedom for this robot arm to nine by allowing three-dimensional translations for the base (Fig. 16-5). If we also allow base rotations, the robot arm can have a total of 12 degrees of freedom. The human body, in comparison, has over 200 degrees of freedom.

Parameterized systems allow object-motion characteristics to be specified as part of the object definitions. The adjustable parameters control such object characteristics as degrees of freedom, motion limitations, and allowable shape changes.

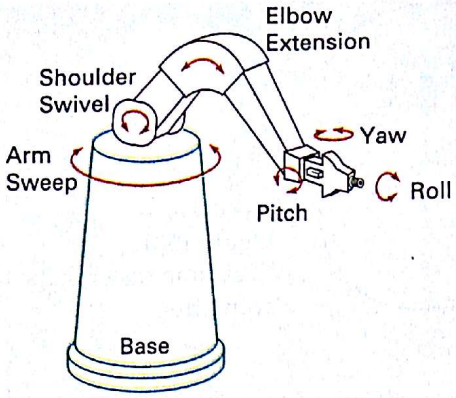


Figure 16-4
Degrees of freedom for a stationary, single-arm robot.

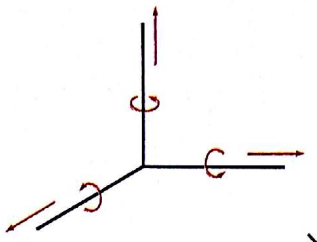


Figure 16-5
Translational and rotational degrees of freedom for the base of the robot arm.

Scripting systems allow object specifications and animation sequences to be defined with a user-input *script*. From the script, a library of various objects and motions can be constructed.

16-5 KEY-FRAME SYSTEMS

We generate each set of in-betweens from the specification of two (or more) key frames. Motion paths can be given with a *kinematic description* as a set of spline curves, or the motions can be *physically based* by specifying the forces acting on the objects to be animated.

For complex scenes, we can separate the frames into individual components or objects called *cels* (celluloid transparencies), an acronym from cartoon animation. Given the animation paths, we can interpolate the positions of individual objects between any two times.

With complex object transformations, the shapes of objects may change over time. Examples are clothes, facial features, magnified detail, evolving shapes, exploding or disintegrating objects, and transforming one object into another object. If all surfaces are described with polygon meshes, then the number of edges per polygon can change from one frame to the next. Thus, the total number of line segments can be different in different frames.

Morphing

Transformation of object shapes from one form to another is called **morphing**, which is a shortened form of metamorphosis. Morphing methods can be applied to any motion or transition involving a change in shape.

Given two key frames for an object transformation, we first adjust the object specification in one of the frames so that the number of polygon edges (or the number of vertices) is the same for the two frames. This preprocessing step is illustrated in Fig. 16-6. A straight-line segment in key frame k is transformed into two line segments in key frame $k + 1$. Since key frame $k + 1$ has an extra vertex, we add a vertex between vertices 1 and 2 in key frame k to balance the number of vertices (and edges) in the two key frames. Using linear interpolation to generate the in-betweens, we transition the added vertex in key frame k into vertex 3' along the straight-line path shown in Fig. 16-7. An example of a triangle linearly expanding into a quadrilateral is given in Fig. 16-8. Figures 16-9 and 16-10 show examples of morphing in television advertising.

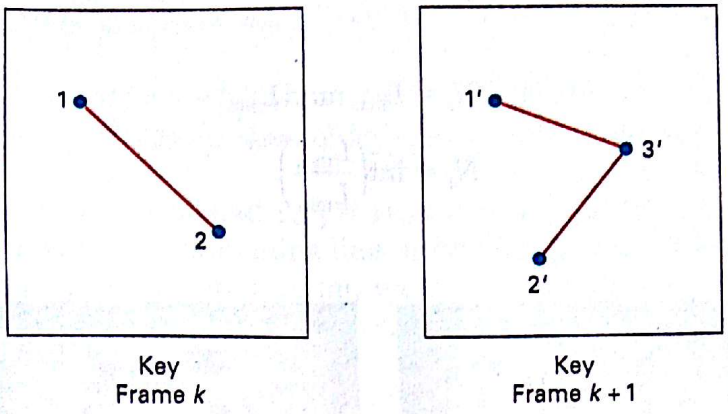


Figure 16-6
An edge with vertex positions 1 and 2 in key frame k evolves into two connected edges in key frame $k + 1$.

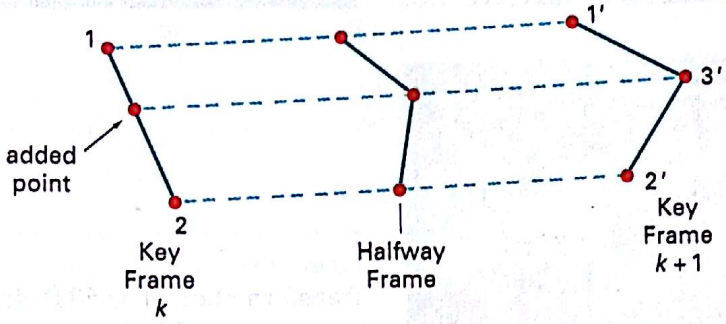


Figure 16-7
Linear interpolation for transforming a line segment in key frame k into two connected line segments in key frame $k + 1$.

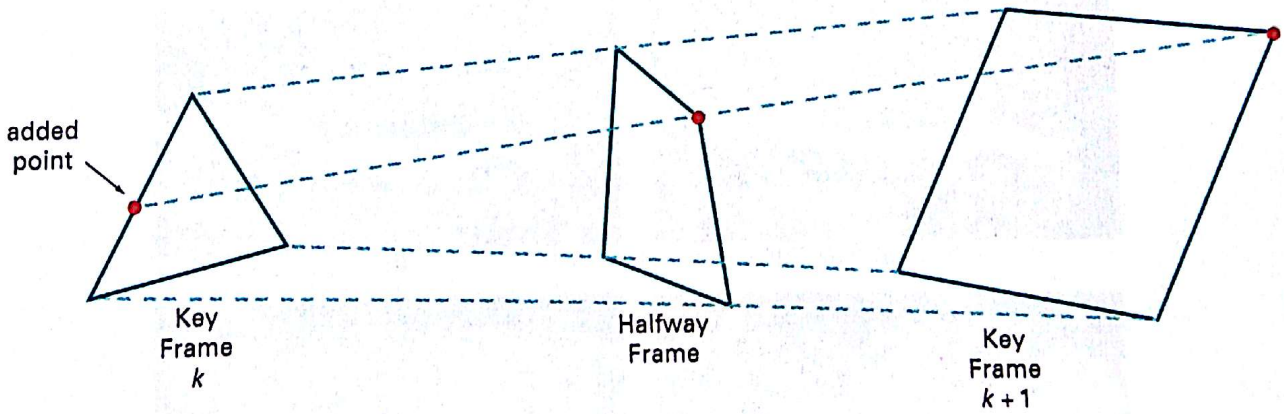


Figure 16-8
Linear interpolation for transforming a triangle into a quadrilateral.

We can state general preprocessing rules for equalizing key frames in terms of either the number of edges or the number of vertices to be added to a key frame. Suppose we equalize the edge count, and parameters L_k and L_{k+1} denote the number of line segments in two consecutive frames. We then define

$$L_{max} = \max(L_k, L_{k+1}), \qquad L_{min} = \min(L_k, L_{k+1}) \tag{16-1}$$

and

$$N_e = L_{\max} \bmod L_{\min}$$
$$N_s = \text{int}\left(\frac{L_{\max}}{L_{\min}}\right) \tag{16-2}$$

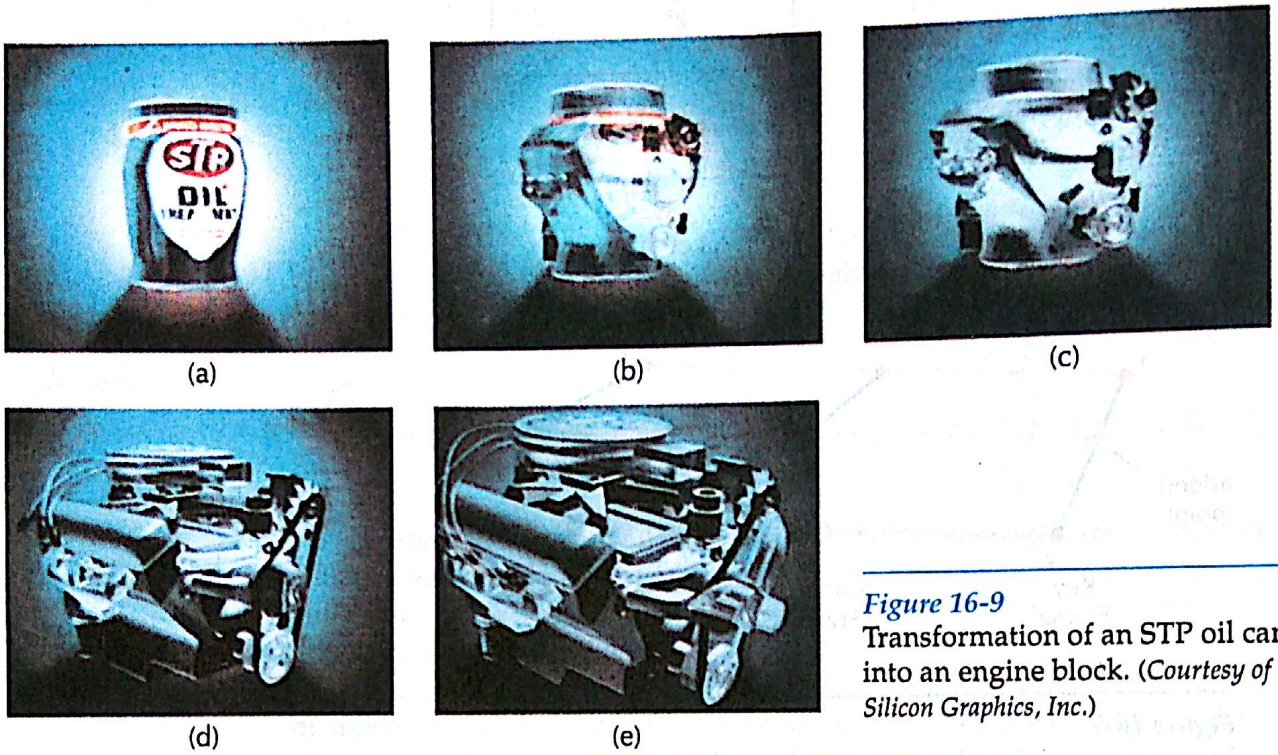


Figure 16-9
Transformation of an STP oil can into an engine block. (*Courtesy of Silicon Graphics, Inc.*)

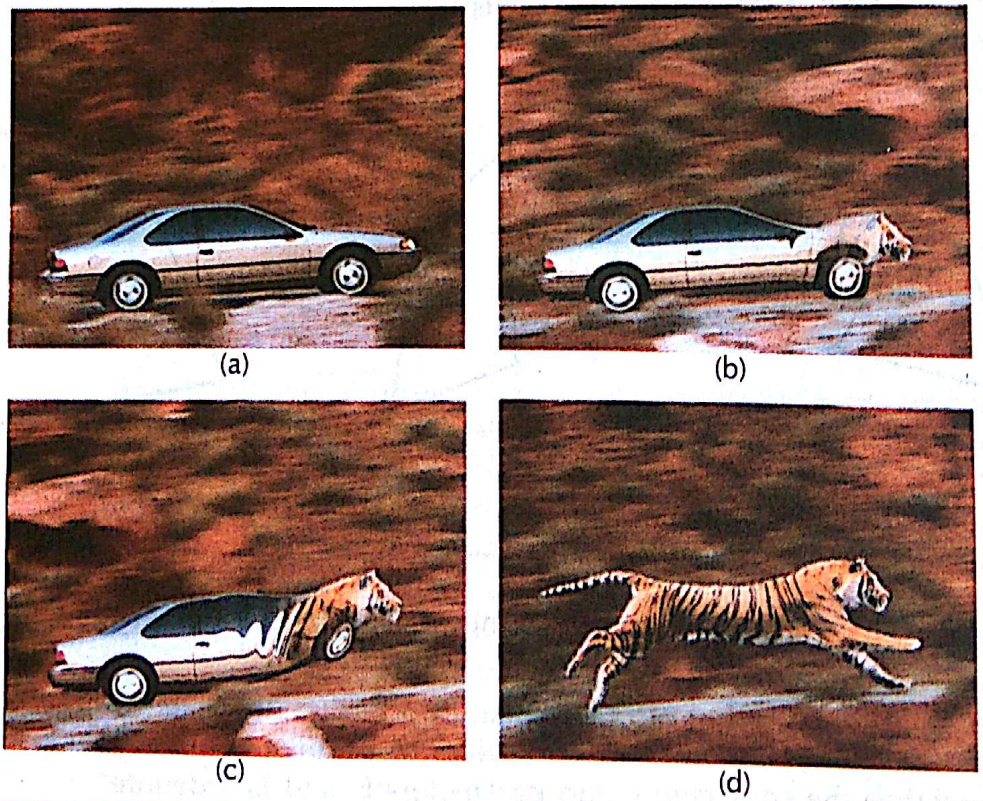


Figure 16-10
Transformation of a moving automobile into a running tiger. (*Courtesy of Exxon Company USA and Pacific Data Images.*)

Then the preprocessing is accomplished by

1. dividing N_e edges of $keyframe_{min}$ into $N_s + 1$ sections
2. dividing the remaining lines of $keyframe_{min}$ into N_s sections

As an example, if $L_k = 15$ and $L_{k+1} = 11$, we would divide 4 lines of $keyframe_{k+1}$ into 2 sections each. The remaining lines of $keyframe_{k+1}$ are left intact.

If we equalize the vertex count, we can use parameters V_k and V_{k+1} to denote the number of vertices in the two consecutive frames. In this case, we define

$$V_{max} = \max(V_k, V_{k+1}), \qquad V_{min} = \min(V_k, V_{k+1}) \tag{16-3}$$

and

$$N_{ls} = (V_{max} - 1) \bmod (V_{min} - 1)$$
$$N_p = \text{int}\left(\frac{V_{max} - 1}{V_{min} - 1}\right) \tag{16-4}$$

Preprocessing using vertex count is performed by

1. adding N_p points to N_{ls} line sections of $keyframe_{min}$
2. adding $N_p - 1$ points to the remaining edges of $keyframe_{min}$

For the triangle-to-quadrilateral example, $V_k = 3$ and $V_{k+1} = 4$. Both N_{ls} and N_p are 1, so we would add one point to one edge of $keyframe_k$. No points would be added to the remaining lines of $keyframe_{k+1}$.

Simulating Accelerations

Curve-fitting techniques are often used to specify the animation paths between key frames. Given the vertex positions at the key frames, we can fit the positions with linear or nonlinear paths. Figure 16-11 illustrates a nonlinear fit of key-frame positions. This determines the trajectories for the in-betweens. To simulate accelerations, we can adjust the time spacing for the in-betweens.

For constant speed (zero acceleration), we use equal-interval time spacing for the in-betweens. Suppose we want n in-betweens for key frames at times t_1 and t_2 (Fig. 16-12). The time interval between key frames is then divided into $n + 1$ subintervals, yielding an in-between spacing of

$$\Delta t = \frac{t_2 - t_1}{n + 1} \tag{16-5}$$

We can calculate the time for any in-between as

$$tB_j = t_1 + j\,\Delta t, \qquad j = 1, 2, \ldots, n \tag{16-6}$$

and determine the values for coordinate positions, color, and other physical parameters.

Nonzero accelerations are used to produce realistic displays of speed changes, particularly at the beginning and end of a motion sequence. We can model the start-up and slow-down portions of an animation path with spline or

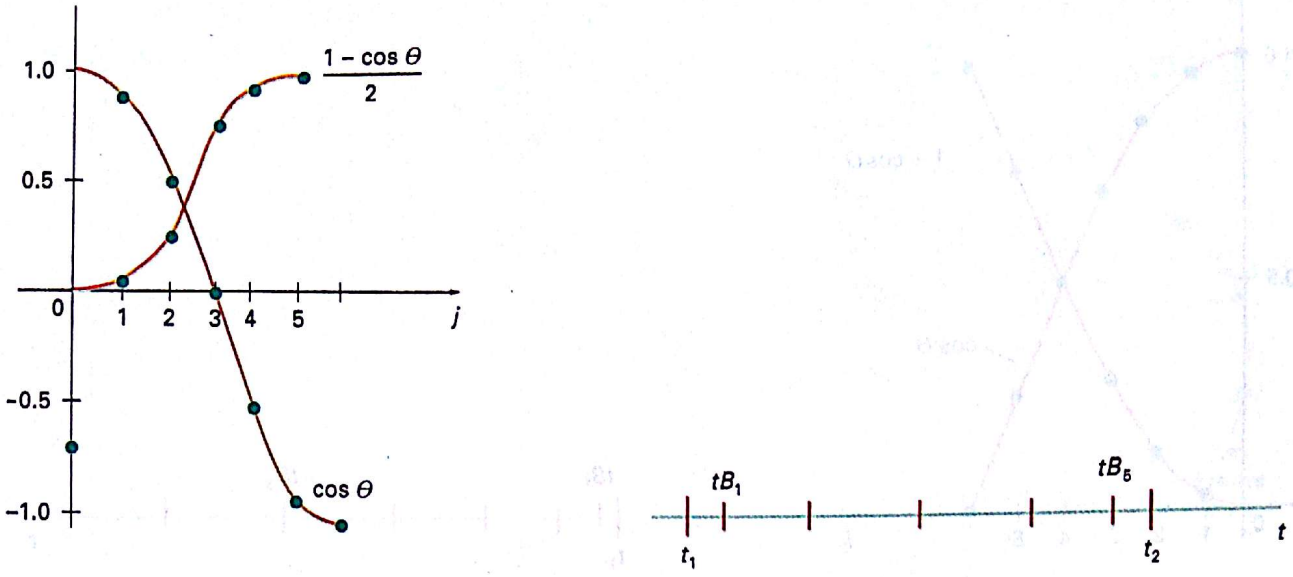


Figure 16-15
A trigonometric accelerate–decelerate function and the corresponding in-between spacing for $n = 5$ in Eq. 16-9.

$$\frac{1}{2}(1 - \cos\theta), \qquad 0 < \theta < \pi/2$$

The time for the jth in-between is now calculated as

$$tB_j = t_1 + \Delta t\left\{\frac{1 - \cos[j\pi/(n+1)]}{2}\right\}, \qquad j = 1, 2, \ldots, n \tag{16-9}$$

with Δt denoting the time difference for the two key frames. Time intervals for the moving object first increase, then the time intervals decrease, as shown in Fig. 16-15.

Processing the in-betweens is simplified by initially modeling "skeleton" (wireframe) objects. This allows interactive adjustment of motion sequences. After the animation sequence is completely defined, objects can be fully rendered.

16-6 MOTION SPECIFICATIONS

There are several ways in which the motions of objects can be specified in an animation system. We can define motions in very explicit terms, or we can use more abstract or more general approaches.

Direct Motion Specification

The most straightforward method for defining a motion sequence is *direct specification* of the motion parameters. Here, we explicitly give the rotation angles and translation vectors. Then the geometric transformation matrices are applied to transform coordinate positions. Alternatively, we could use an approximating

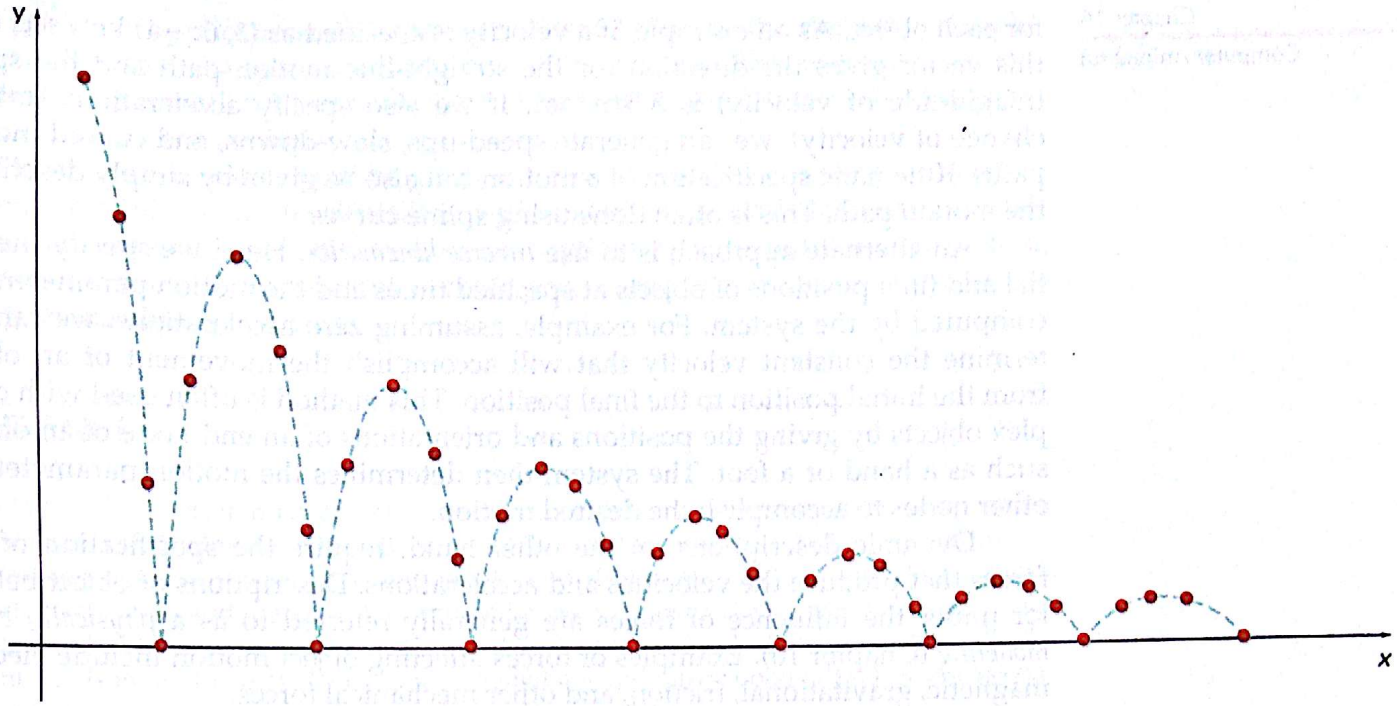


Figure 16-16
Approximating the motion of a bouncing ball with a damped *sine* function (Eq. 16-10).

equation to specify certain kinds of motions. We can approximate the path of a bouncing ball, for instance, with a damped, rectified, *sine* curve (Fig. 16-16):

$$y(x) = A\,|\sin(\omega x + \theta_0)|\,e^{-kx} \qquad (16\text{-}10)$$

where A is the initial amplitude, ω is the angular frequence, θ_0 is the phase angle, and k is the damping constant. These methods can be used for simple user-programmed animation sequences.

Goal-Directed Systems

At the opposite extreme, we can specify the motions that are to take place in general terms that abstractly describe the actions. These systems are referred to as *goal directed* because they determine specific motion parameters given the goals of the animation. For example, we could specify that we want an object to "walk" or to "run" to a particular destination. Or we could state that we want an object to "pick up" some other specified object. The input directives are then interpreted in terms of component motions that will accomplish the selected task. Human motions, for instance, can be defined as a hierarchical structure of submotions for the torso, limbs, and so forth.

Kinematics and Dynamics

We can also construct animation sequences using *kinematic* or *dynamic* descriptions. With a kinematic description, we specify the animation by giving motion parameters (position, velocity, and acceleration) without reference to the forces that cause the motion. For constant velocity (zero acceleration), we designate the motions of rigid bodies in a scene by giving an initial position and velocity vector

for each object. As an example, if a velocity is specified as (3, 0, −4) km/sec, then this vector gives the direction for the straight-line motion path and the speed (magnitude of velocity) is 5 km/sec. If we also specify accelerations (rate of change of velocity), we can generate speed-ups, slow-downs, and curved motion paths. Kinematic specification of a motion can also be given by simply describing the motion path. This is often done using spline curves.

An alternate approach is to use *inverse kinematics*. Here, we specify the initial and final positions of objects at specified times and the motion parameters are computed by the system. For example, assuming zero accelerations, we can determine the constant velocity that will accomplish the movement of an object from the initial position to the final position. This method is often used with complex objects by giving the positions and orientations of an end node of an object, such as a hand or a foot. The system then determines the motion parameters of other nodes to accomplish the desired motion.

Dynamic descriptions on the other hand, require the specification of the forces that produce the velocities and accelerations. Descriptions of object behavior under the influence of forces are generally referred to as a *physically based modeling* (Chapter 10). Examples of forces affecting object motion include electromagnetic, gravitational, friction, and other mechanical forces.

Object motions are obtained from the force equations describing physical laws, such as Newton's laws of motion for gravitational and friction processes, Euler or Navier–Stokes equations describing fluid flow, and Maxwell's equations for electromagnetic forces. For example, the general form of Newton's second law for a particle of mass m is

$$\mathbf{F} = \frac{d}{dt}(m\mathbf{v}) \tag{16-11}$$

with $\mathbf{F}$ as the force vector, and $\mathbf{v}$ as the velocity vector. If mass is constant, we solve the equation $\mathbf{F} = m\mathbf{a}$, where $\mathbf{a}$ is the acceleration vector. Otherwise, mass is a function of time, as in relativistic motions or the motions of space vehicles that consume measurable amounts of fuel per unit time. We can also use *inverse dynamics* to obtain the forces, given the initial and final positions of objects and the type of motion.

Applications of physically based modeling include complex rigid-body systems and such nonrigid systems as cloth and plastic materials. Typically, numerical methods are used to obtain the motion parameters incrementally from the dynamical equations using initial conditions or boundary values.

SUMMARY

A computer-animation sequence can be set up by specifying the storyboard, the object definitions, and the key frames. The storyboard is an outline of the action, and the key frames define the details of the object motions for selected positions in the animation. Once the key frames have been established, a sequence of in-betweens can be generated to construct a smooth motion from one key frame to the next. A computer animation can involve motion specifications for the objects in a scene as well as motion paths for a camera that moves through the scene. Computer-animation systems include key-frame systems, parameterized systems, and scripting systems. For motion in two-dimensions, we can use the raster-animation techniques discussed in Chapter 5.