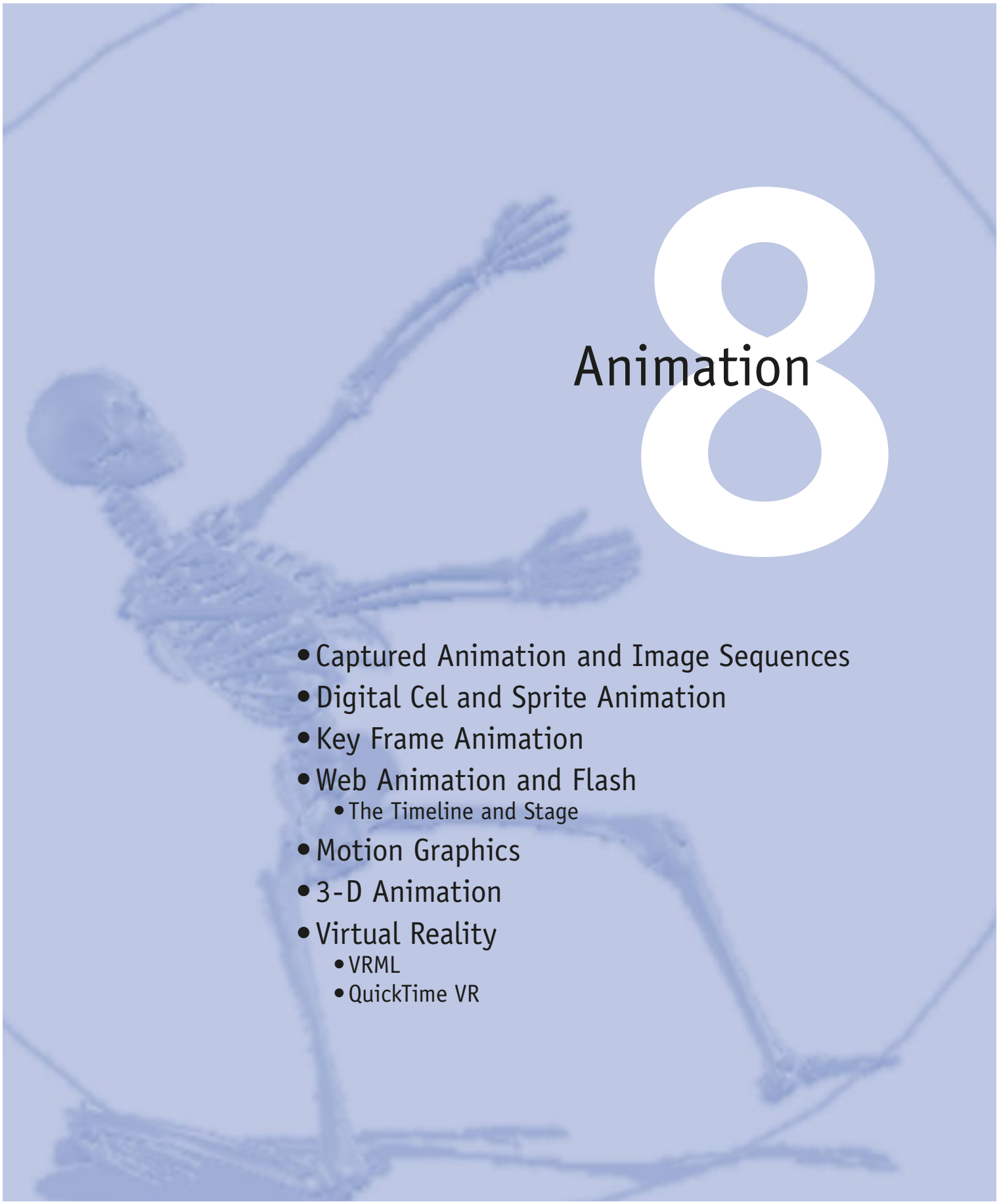


8 Animation

- Captured Animation and Image Sequences
- Digital Cel and Sprite Animation
- Key Frame Animation
- Web Animation and Flash
  - The Timeline and Stage
- Motion Graphics
- 3-D Animation
- Virtual Reality
  - VRML
  - QuickTime VR

8

Animation may be defined as the creation of moving pictures one frame at a time; the word is also used to mean the sequences produced in this way, as in 'a Disney animation' or 'Web animation'. Throughout the twentieth century, animation was used for entertainment, advertising, instruction, art and propaganda on film, and latterly on video; it is now also widely employed on the World Wide Web and in multimedia presentations.

To see how animation works, consider making a sequence of drawings or paintings on paper, in which those elements or characters intended to change or move during the sequence are altered or repositioned in each drawing. The changes between one drawing and the next may be very subtle, or much more noticeable. Once the drawings are complete, the sequence of drawings is photographed in the correct order, a single frame at a time. When the film is played back, this sequence of still images is perceived in just the same way as the sequence of frames exposed when live action has been filmed in real time: persistence of vision causes the succession of still images to be perceived as a continuous moving image. If you wish to convey the illusion of fast movement or change, the differences between successive images in the sequence must be much greater than if the change is to be gradual, or the movement slow.

'Animate' literally means 'to bring to life', which captures the essence of the process: when played back at normal film or video speeds, the still characters, objects, abstract shapes, or whatever, that have been photographed in sequence, appear to come to life.

As film is projected at 24 frames per second, drawn animation in traditional media, as we have just described it, technically requires 24 drawings for each second of film, that is, 1440 drawings for every minute – and even more for animation made on video. In practice, animation that does not require seamlessly smooth movement can

be shot 'on 2s', which means that two frames of each drawing, or whatever, are captured rather than just one. This gives an effective frame rate of 12 frames per second for film, or 15 for NTSC video. Digital animation can actually be played back at these lower frame rates, with the same saving in labour.

If an animation is made solely from drawings or paintings on paper, every aspect of the image has to be repeated for every single frame that is shot. In an effort to reduce the enormous amount of labour this process involves, as well as in a continuing search for new expressive possibilities, many other techniques of animation have been devised. The best known and most widely used – at least until very recently – has been *cel animation*. In this method of working, those elements in a scene that might move – Homer Simpson, for example – are drawn on sheets of transparent material known as 'cel', and laid over a background – the Simpsons' living room, perhaps – drawn separately. In producing a sequence, only the moving elements on the cel need to be redrawn for each frame; the fixed parts of the scene need only be made once. Many cels might be overlaid together, with changes being made to different ones between different frames to achieve a greater complexity in the scene. To take the approach further, the background can be drawn on a long sheet, extending well beyond the bounds of a single frame, and moved between shots behind the cels, to produce an effect of travelling through a scene. The concepts and techniques of traditional cel animation have proved particularly suitable for transfer to the digital realm, and many popular cel-like cartoons on TV are now produced digitally.

Largely because of the huge influence of the Walt Disney studios, where cel animation was refined to a high degree, with the use of multi-plane set-ups that added a sense of three-dimensionality to the work, cel has dominated the popular perception of animation. It was used in nearly all the major cartoon series, from *Popeye* to *The Simpsons* and beyond, as well as in many full-length feature films, starting with *Snow White and the Seven Dwarfs* in 1937. However, from the very beginnings of moving pictures in the 1890s, animation has been successfully created by employing a variety of other means. Many artists do indeed work by drawing each frame separately on paper, while others, even more painstaking, have painted directly on to film, or scratched the emulsion of blackened film stock; others have worked with sand or oil paint on glass, or chalks on paper or card, making changes to the created image between every shot; still others have manipulated front or back lit cut-outs under the camera

– Terry Gilliam's distinctive work for the *Monty Python* TV series is a well-known example of cut-out animation. Sometimes animators have invented a completely new way of working for themselves, such as Alexeieff and Parker's pin screen, in which closely spaced pins are selectively pushed through a board and lit so that the shadows they cast form an image, which is changed between each shot.

A distinct alternative to all of these essentially two-dimensional forms is three-dimensional, or *stop-motion,* animation. This encompasses several techniques, but all use miniature three-dimensional sets, like stage sets, on which objects are moved carefully between shots. The objects may include articulated figures, whose limbs can be repositioned, or solid figures whose parts are replaced, or substituted, between shots, to produce an effect of gestures, walking, and so on. Figures and other objects made out of a malleable modelling material, such as Plasticine, may be used instead; these can be manipulated between shots, to produce both natural movement and otherwise impossible changes and transformations. This latter form of animation – often called *clay animation* – has achieved recent prominence with the work of the Aardman studios whose output includes the *Wallace and Gromit* films.

Although it may be convenient to consider the various techniques of animation separately, hybrid forms of animation are often produced – mixing cel and 3-D, for example. There is also a long tradition of combining animation with live footage. The most celebrated example of this is perhaps *Who Framed Roger Rabbit?* (1988), but a mixture of live action and animation was employed in some of the earliest films ever made, including Georges Méliès' well known 'trick films', and Max Fleischer's *Out of the Inkwell* series of the 1920s, which did much to popularize animation as a form of entertainment. Recently, the eager adoption of digital technology by the film industry has led to a substantially increased use of animation in conjunction with live action, particularly in special effects movies. It is perhaps not always realised by an audience that much of what they perceive as 'special effects' has been achieved by basic animation techniques, whether traditional, as in the 1933 classic *King Kong* and many other monster movies, or digital, as in *The Matrix* and its sequels, for example.

All of the traditional forms of animation have their counterparts in the digital realm. Moreover, digital technology affords new opportunities for using animation and techniques derived from it in new contexts.

Captured Animation and Image Sequences

As we will see, digital technology has brought new ways of creating animation, but computers can also be used effectively in conjunction with the older methods discussed above, to produce animation in a digital form, suitable for incorporation in multimedia productions. Currently, preparing animation in this way – using digital technology together with a video camera and traditional animation methods – offers much richer expressive possibilities to the animator working in digital media than the purely computer-generated methods we will describe later in this chapter.

Figure 8.1 A cel-like digital animation sequence

Instead of recording your animation on film or videotape, a video camera is connected directly to a computer, to capture each frame of animation to disk – whether it is drawn on paper or cel, constructed on a 3-D set, or made using any other technique that does not depend on actually marking the film. Instead of storing the entire data stream arriving from the camera, as you would if you were capturing live video, you only store the digital version of a single frame each time you have set up a shot correctly. Many small utilities are available for performing *frame grabbing* of this sort, and some video editing applications provide the facility – Premiere, for example, offers a Stop Frame command on its Capture menu. Frame grabbers all work in roughly the same way: a recording window is displayed, showing the current view through the camera. You can use this to check the shot, then press a key to capture one frame, either to a still image file, or to be appended to an AVI or QuickTime movie sequence. You then change your drawing, alter the position of your models, or whatever, and take another shot. Frames that are unsatisfactory can be deleted; usually, an option allows you to see a ghost image of the previously captured frame, to help with alignment and making the appropriate changes. When you have captured a set of frames that forms a sequence, you can save it as a QuickTime movie or a set of sequentially numbered image files (see below). The latter option is useful if you want to manipulate individual images in Photoshop or import them into Flash, for example.

Capturing animation to disk in the manner just outlined not only opens up the possibilities of non-linear editing and post-production that we described in Chapter 7, it also allows animation in traditional media to be combined with purely digital animation and motion graphics; Figure 8.2 shows an example from a work produced in this hybrid fashion.

Figure 8.2 Stop-frame animation captured to disk and combined with digital animation

For certain types of traditional animation, it is not even necessary to use a camera. If you have made a series of drawings or paintings on paper, you can use a scanner to produce a set of image files from them. You can also manipulate cut-outs on the bed of a scanner, almost as easily as under a camera. A film scanner will even allow you to digitize animation made directly onto film stock. You can to use a digital stills camera instead of a video camera, provided it allows you to download images directly to disk. In all of these cases you are able to work at higher resolution, and with a larger colour gamut, than is possible with a video camera.

For drawn or painted animation you can dispense with the external form and the digitization process entirely by using a graphics program to make your artwork, and save your work as a movie or as a sequence of image files. You can use the natural media brushes of Painter or recent versions of Photoshop to produce animation that looks (somewhat) as if it was produced with traditional materials, or you can take advantage of the pixel manipulating facilities of these programs to produce work with a characteristic digital look. Even when you are producing your animation a frame at a time, a program can sometimes save you work, by letting you record macros or scripts for drawing repeated elements or applying filters to many frames.

Sequences of image files provide a very flexible representation of an animation. Individual files can be opened in a graphics program to be altered; single files can be removed from the sequence, replaced or added. The sequence can then be imported into a video editing application and converted into an AVI or QuickTime movie. However, managing a collection of image files can become complicated, especially if you eventually want to import them into a video editing program. In order for this to be possible without tying you to a particular combination of programs, the files' names must conform to some convention. For example, on the Macintosh, Premiere can only import a

sequence of PICT files if they are all in the same folder, and all the files have a suffix consisting of a period followed by the same number of digits, for example Animation.001, Animation.002, … Animation.449. (Failure to provide the necessary leading zeroes will have consequences that you can probably guess at.) If you make any changes to the set of images, you must take care not to disturb the numbering, or to adjust it if necessary.

Several computer programs, including Painter and Flash (which we will describe in more detail later) let you open a movie and modify its individual frames. (Photoshop can open files in a special filmstrip format, which Premiere can export and re-import, for similar purposes.) This offers new possibilities. You can, for example, paint onto or otherwise alter original video material, which is one way of adding animation to live action.† Another option is to trace, frame by frame on a layer, selected elements from a live action video clip, which is subsequently deleted. This process, whether achieved digitally or by older means, is what is properly referred to as *rotoscoping*, and has long been used to create animation that accurately reproduces the forms and natural movements of people and animals.

† In computer graphics circles, this process of painting onto existing video frames is sometimes called 'rotoscoping', but the use of the term is inaccurate, as explained below.

Rotoscoping is named after the *rotoscope*, a device patented by Max Fleischer (of *Betty Boop* and original animated *Popeye* fame) in 1915. Fleischer's device projected movie footage, one frame at a time, onto a light table, giving a back-projected still image over which the animator could place a sheet of animation paper. When the tracing of one frame was complete, the film was advanced to the next by means of a hand crank.

Instead of using a set of still image files to hold an animation sequence, you can sometimes use a single 'image' file to hold several images. While a surprising number of file formats – including, but not exclusively, formats intended for use with animation software – offer this facility, by far the most common is GIF.

GIF files' ability to store a sequence of images has been used to provide a cheap and cheerful form of animation for Web pages. Most Web browsers will display each image contained in a GIF file in turn when they load the file. If the displaying happens fast enough, the images will be seen as an animation. The GIF89a version of the format provides for some optional data items that control the behaviour of an *animated GIF*, as these files are called. In particular, a flag can be set to cause the animation to loop, either for a stipulated number of times or indefinitely, and a minimum delay between frames, and hence a frame rate, can be specified. However, animated GIFs do not provide a very reliable way of adding animated features to Web pages. As with most aspects of a browser's behaviour, the way in which animated GIFs are displayed can be changed by users – looping can be turned off, animation can be prevented, and if image loading is disabled, animated GIFs will not appear at all – and not all browsers offer a proper implementation. The main advantage of animated GIFs is that they do not rely on any plug-in, or the use of scripting (see Chapter 16), so they will be viewable using a wider range of browsers.

Several free or inexpensive utilities are available on the major platforms for combining a set of images into a single animated GIF; Premiere and Flash allow you to save a movie in this form, too, and dedicated Web graphics programs, such as ImageReady and Fireworks, can be used to create animated GIFs from scratch or by altering existing images. Potentially, therefore, GIF files can be used to store any form of animation. However, even when GIF animation is properly implemented and enabled, it has many shortcomings. You cannot add sound; you are restricted to a 256 colour palette; your images are losslessly compressed, which may conserve their quality, but does not provide much compression, a serious consideration that effectively prevents the use of this format for any extended animation sequences. Usually, each frame of an animated GIF is displayed by the browser as it arrives. Network speeds mean that there may be excessive, and probably irregular, delays between frames, making any frame rate that may be specified in the file irrelevant. However, if an animation is set to loop, once it has played through the first time it will have been copied into the browser's local cache (unless it is too big), and subsequent loops will play at a speed only limited by the user's processor and disk (which are completely unknown to the animator). In general, there is little chance of an animated GIF consistently playing back at a sufficiently high frame rate to give smooth animation, unless it is small. Usually, therefore, animated GIFs are not

used for realistic animation, but for more stylized changing images, often resembling neon advertising signs. Possibly for this reason, by association of ideas, Web page advertising is what animated GIFs are most often used for. It is probably fair to say that, because of the ease with which animated advertisements can be incorporated into a Web page by almost anybody, they have been used for some of the worst animation ever produced. (Possibly as a result, the use of animated GIFs has declined in recent years.)

For captured animation of any duration, especially if it is accompanied by sound, the best results will be achieved using a video format. Once you have captured or painted an animation sequence and saved it as, or converted it to, QuickTime, for example, what you have is just an ordinary QuickTime movie, so it can be edited, combined with other clips, have effects applied, and be embedded in a Web page, just like any other video clip. However, animation clips may have some distinctive features which affect the way you deal with them. In particular, certain styles of drawn animation tend to feature simplified shapes and areas of flat colour. (This is *not* invariably the case; the characteristics of the images depend on the individual animator's style.) Material of this type may be more amenable to lossless compression than other types of video. QuickTime's *Animation* codec is designed to take advantage of the characteristics of simple cartoon-style drawn animation, which, as we will see later in this chapter, are often shared by computer-generated 3-D animation. Compression is based on run-length encoding (RLE), and, when the codec is used at its highest quality setting, is lossless. There is also a lossy mode that can be used to achieve higher compression ratios. Because it is based on RLE, this codec can compress areas of flat colour well, which is what makes it suitable for animation in the particular styles just mentioned.

'Digital Cel' and Sprite Animation

Our earlier description of cel animation may have put you in mind of layers, as described in Chapter 3. Layers allow you to create separate parts of a still image – for example, a person and the background of a scene they are walking through – so that each can be altered or moved independently. The frames of an animated sequence can be made by combining a background layer, which remains static, with one or more animation layers, in which any changes that take place between frames are made. Thus, to create an animation, you would begin by creating the background layer in the image for the first frame. Next,

on separate layers, you create the elements that will move; you may want to use additional static layers in between these moving layers if you need to create an illusion of depth. After saving the first frame, you begin the next by pasting the background layer from the first; then, you add the other layers, incorporating the changes that are needed for your animation. In this way, you do not need to recreate the static elements of each frame, not even using a script.

Where the motion in an animation is simple, it may only be necessary to reposition or transform the images on some of the layers. To take a simple example, suppose we wish to animate the movement of a planet across a background of stars. The first frame could consist of a background layer containing the star field, and a foreground layer with an image of our planet. To create the next frame, we would copy these two layers, and then, using the move tool, displace the planet's image a small amount. By continuing in this way, we could produce a sequence in which the planet moved across the background. (If we did not want the planet to move in a straight line, it would be necessary to rotate the image as well as displace it, to keep it tangential to the motion path.)

Using layers as the digital equivalent of cel saves the animator time, but, as we have described it, does not affect the way in which the completed animation is stored: each frame is saved as an image file, and the sequence will later be transformed into a QuickTime movie, an animated GIF, or any other conventional representation. Yet there is clearly a great deal of redundancy in a sequence whose frames are all built out of the same set of elements. Possibly, when the sequence comes to be compressed, the redundant information will be squeezed out, but compressing after the event is unlikely to be as successful as storing the sequence in a form that exploits its redundancy in the first place. In general terms, this would mean storing a single copy of all the static layers and all the objects (that is, the non-transparent parts) on the other layers, together with a description of how the moving elements are transformed between frames.

This form of animation, based on moving objects, is called *sprite animation*, with the objects being referred to as *sprites*. Slightly more sophisticated motion can be achieved by associating a set of images, sometimes called *faces*, with each sprite. This would be suitable to create a 'walk cycle' for a humanoid character, for example (see Figure 8.3). By advancing the position of the sprite and cycling through the faces, the character can be made to walk.

QuickTime supports sprite tracks, which store an animation in the form of a 'key frame sample' followed by some 'override samples'. The key frame sample contains the images for all the faces of all the sprites used in this animation, and values for the spatial properties (position, orientation, visibility, and so on) of each sprite, as well as an indication of which face is to be displayed. The override samples contain no image data, only new values for the properties of any sprites that have changed in any way. They can therefore be very small. QuickTime sprite tracks can be combined with ordinary video and sound tracks in a movie.

We have described sprite animation as a way of storing an animated sequence, but it is often used in a different way. Instead of storing the changes to the properties of the sprites, the changed values can be generated dynamically by a program. Simple motion sequences that can be described algorithmically can be held in an even more compact form, therefore, but, more interestingly, the computation of sprite properties can be made to depend upon external events, such as mouse movements and other user input. In other words, the movement and appearance of animated objects can be controlled by the user. This way of using sprites has been extensively used in computer games, but it can also be used to provide a dynamic form of interaction in other contexts, for example, simulations.

Key Frame Animation

During the 1930s and 1940s, the large American cartoon producers, led by Walt Disney, developed a mass production approach to animation. Central to this development was division of labour. Just as Henry Ford's assembly line approach to manufacturing motor cars relied on breaking down complex tasks into small repetitive sub-tasks that could be carried out by relatively unskilled workers, so Disney's approach to manufacturing dwarfs relied on breaking down the production of a sequence of drawings into sub-tasks, some of which, at least, could be performed by relatively unskilled staff. Disney was less successful at de-skilling animation than Ford was at de-skilling manufacture – character design, concept art, storyboards, tests, and some of the animation, always had to be done by experienced and talented artists. But when it came to the production of the final cels for a film, the role of trained animators was largely confined to the creation of *key frames*.

We have met this expression already, in the context of video compression and also in connection with QuickTime sprite tracks. There,

Figure 8.3 Sprite faces for a walk cycle

key frames were those which were stored in their entirety, while the frames in between them were stored as differences only. In traditional animation, the meaning has a slightly different twist: key frames are typically drawn by a 'chief animator' to provide the pose and detailed characteristics of characters† at important points in the animation. Usually, key frames occur at the extremes of a movement – the beginning and end of a walk, the top and bottom of a fall, and so on – which determine more or less entirely what happens in between, but they may be used for any point which marks a significant change. The intermediate frames can then be drawn almost mechanically by 'in-betweeners'. Each chief animator could have several in-betweeners working with him‡ to multiply his productivity. (In addition, the tedious task of transferring drawings to cel and colouring them in was also delegated to subordinates.)

† The mass production approach to animation is almost invariably associated with cartoons featuring characters.

‡ This 'him' is not a casual slip: the big cartoon studios of those days did not have what we would consider an enlightened attitude to women as animators.

In-betweening (which is what in-betweeners do) resembles what mathematicians call *interpolation*: the calculation of values of a function lying in between known points. Interpolation is something that computer programs are very good at, provided the values to be computed and the relationship between them can be expressed numerically. Generally, the relationship between two key frames of a hand-drawn animation is too complex to be reduced to numbers in a way that is amenable to computer processing. But this does not prevent people trying – because of the potential labour savings.

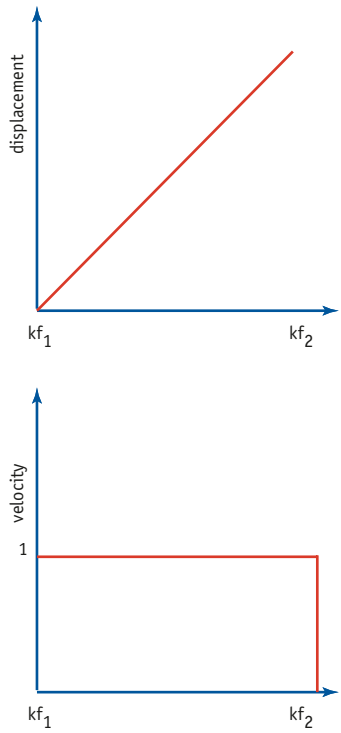


Figure 8.4 Linearly interpolated motion

All digital images are represented numerically, in a sense, but the numerical representation of vector images is much simpler than that of bitmapped images, making them more amenable to numerical interpolation. To be more precise, the transformations that can be applied to vector shapes – translation, rotation, scaling, reflection and shearing – are arithmetical operations that can be interpolated. Thus, movement that consists of a combination of these operations can be generated by a process of numerical in-betweening starting from a pair of key frames. This means that cartoon-like animation can be created digitally in programs like Flash with considerable savings of effort compared with traditional methods.

If we just consider motion in a straight line, the simplest form of interpolation is *linear*. This means that an object moves an equal distance between each frame, the distance moved per frame being the total distance between the object's positions in the starting and ending key frames, divided by the number of frames in the sequence. Putting it

more simply, the symbol moves at a constant velocity, which causes two problems.

First, motion begins and ends instantaneously, with objects attaining their full velocity as soon as they start to move, and maintaining it until they stop. Nothing really moves like that. To produce a more natural movement, programs that implement interpolated motion (including Flash) borrow a technique from hand-made animation: the transition from stasis to movement is made more gradual by using smaller, slowly increasing, increments between the first few frames (i.e. the object accelerates from a standstill to its final velocity), a process referred to as *easing in*. The converse process of deceleration is called *easing out*. Figure 8.4 shows the way the horizontal displacement and velocity of an object changes with time when it is moved from an initial position in key frame 1 (kf_1) of (0, 0) to a final position in key frame 2 (kf_2) of (50, 50), using linear interpolation over 50 frames. Figures 8.5 and 8.6 show how the change might be modified when the motion is eased in or out – we have shown a style of easing that uses *quadratic* interpolation, that is, the acceleration is constant. More complicated styles are possible and might be preferred. When applying easing in Flash, the animator can set the degree of easing using a slider that moves from maximum easing in, through a constant velocity, to maximum easing out. In effect, this moves the displacement curve from one like Figure 8.5, via similar curves with less pronounced bulge, through Figure 8.4 and beyond to Figure 8.6. (That is, the acceleration goes from some maximum positive value, through zero, to a maximum negative value.)

The second problem with linear interpolation can be seen in Figure 8.7, which shows how displacement and velocity change if we now append to our original sequence a second one of 50 frames, during which our object moves from its position in kf_2 of (50, 50) to a new position at (75, 75) in kf_3. Because each sequence is interpolated separately as a straight line, there is a sharp discontinuity at kf_2; as the velocity graph clearly shows, this will appear as a sudden deceleration at that point in the animation. Again, this is an unnatural sort of movement, that will rarely be what is desired. By clever manipulation of the easing slider, it would be possible to smooth out this abruptness, but a more general solution to the problem is available. In Chapter 4, we stressed that Bézier curves' most attractive property is that they can be joined together smoothly by aligning their tangent vectors. By using Bézier curves instead of straight lines to interpolate between key frames, smooth motion can be achieved. Note that we do

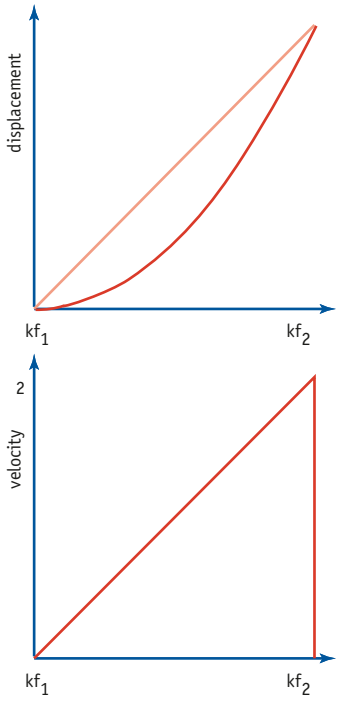


Figure 8.5 Easing in

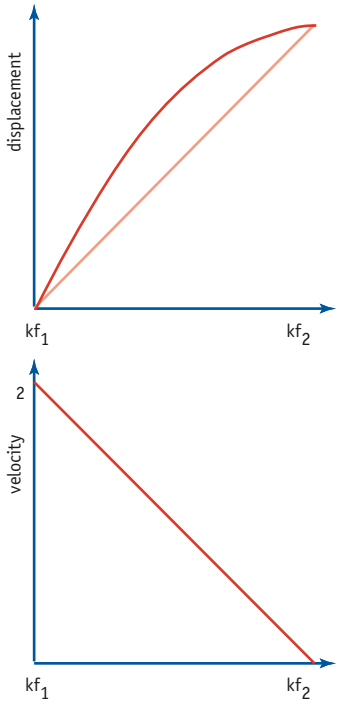


Figure 8.6 Easing out

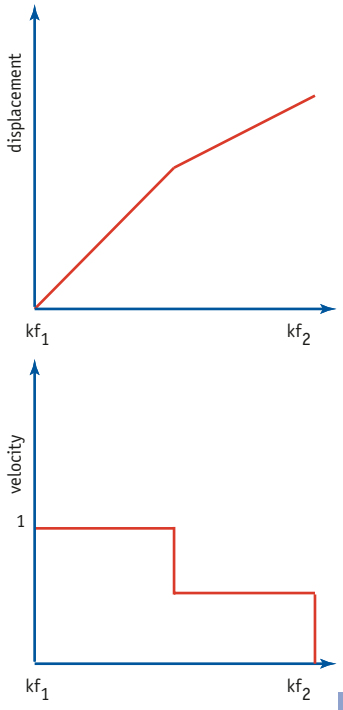


Figure 8.7 Abrupt change of velocity

not mean that objects should follow Bézier shaped paths, but that the *rate* at which their properties change should be interpolated using a Bézier curve. Flash does not offer the option of any form of non-linear interpolation but other, more elaborate, key frame animation systems do. Although we have only considered displacement in one direction, you should be able to see that similar considerations apply to any transformations that might be interpolated.

Web Animation and Flash

Animation can be added to Web pages in the form of animated GIFs or embedded video, but the most popular Web animation format is *Shockwave Flash (SWF)*, which is usually generated using Macromedia Flash. SWF is a vector animation format, which makes it particularly suitable for Web animation, since graphic objects can be compactly represented in vector form, and, as we outlined in the previous section, motion can be represented as numerical operations on the vector data. Thus, SWF animations can have lower bandwidth requirements than video or any bitmapped format. The price paid for this lower bandwidth is that vector animations do not offer the full range of visual possibilities available in bitmaps.

Although SWF files *may* be small, only needing low bandwidth, they are not necessarily so. Flash does allow bitmapped images to be imported into animations, and certain styles of drawing and animation lead to vector animations that are of considerable size. In fact, many Flash animations found on the Web take an unacceptable time to load over a dial-up connection. If bandwidth considerations are important to you, you need to make a conscious effort to avoid using bitmaps, draw simple vector shapes and use interpolated motion as much as possible in your animations.

Flash is more than an animation program. It supports a powerful scripting language, called ActionScript, which makes it possible to add interactivity to animations and to build Web applications with user interfaces created in Flash. We will return to these aspects of Flash in Chapter 16. Scripting can also be used to create animations in Flash, by using algorithms to set the position of movie clip symbols. Instead of creating motion by hand, objects can be made to move according to the laws of physics, or some mathematical description of behaviour, such as the way flocks of birds move as a group.

The Timeline and Stage

An animation being created in Flash is organized using a *timeline*, a graphical representation of a sequence of frames, similar to the timeline in video editing applications. Animations can be built up a single frame at a time, by inserting key frames into the timeline sequentially.

Flash's *stage* is a sub-window in which frames are created by drawing objects. Objects can be created on the stage using some built-in drawing tools, similar to but less comprehensive than those of Illustrator or Freehand, or they can be imported from those applications. Bitmap images, in formats including JPEG and PNG, may also be imported and auto-traced to make vector objects; images can be used in bitmap form within a Flash frame, but cannot then be rotated or scaled without potentially degrading the image. Comprehensive support is provided for text; characters in outline fonts can be decomposed into their component outline paths, which can be edited or animated separately. Layers can be used to organize the elements of a frame; they also play a key role in interpolating motion. Figure 8.8 shows the stage and timeline as they appear in Flash. The Flash interface also contains a toolbox, containing the vector drawing tools, and a host of palettes, for colour mixing, alignment, applying transformations, setting typographic options, and so on.

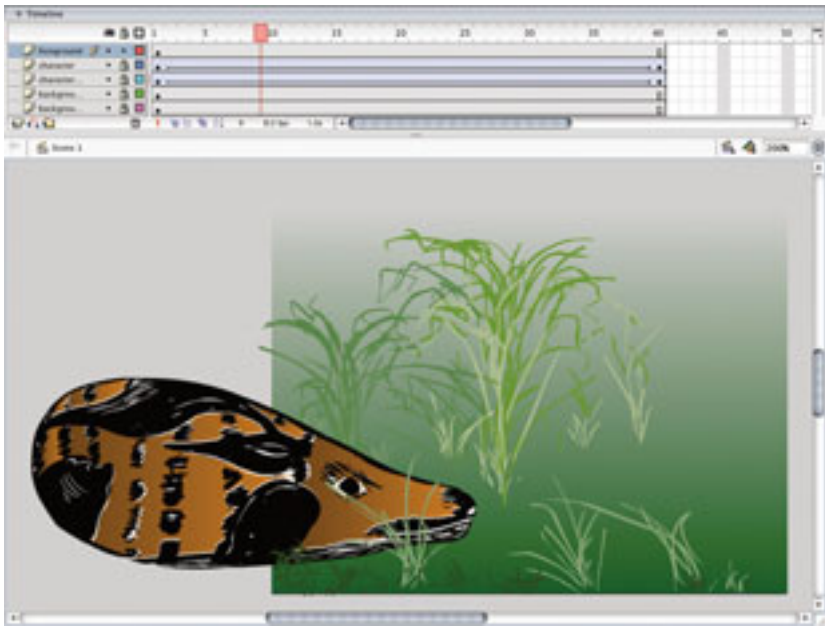


Figure 8.8 The timeline (top) and stage (below) in a simple Flash movie

When a movie is first created, it contains a single empty key frame. When a key frame is added to the timeline immediately after an existing key frame, it starts out with a copy of the contents of the preceding key frame. Since most animation sequences exhibit only small changes between frames, an efficient way of working on animations created a frame at a time is by adding key frames incrementally at the end of the current sequence and making changes to their contents. To assist with this sort of animation, Flash provides an *onion-skinning* facility; when this is turned on, up to five preceding frames are displayed semi-transparently under the current frame. This makes it easier to see the changes between frames, and to align objects correctly.

As well as key frames, the timeline can also hold simple frames. These contain no objects of their own; when the movie is played back, they continue to display the contents of the most recent key frame. That is, they hold on the key frame. You can add frames and key frames independently to different layers, so one layer may hold a static background image with moving elements on layers above it. The background layer will have just one key frame at the beginning, while the moving layers will have key frames at every point where an object moves. This may be every single frame.

Symbols and Tweening

Graphical objects can be stored in a library in a special form, called a *symbol*, that allows them to be reused. Multiple *instances* of a symbol may be placed on the stage. They will all be fundamentally identical, but transformations can be applied, to change the size and orientation of each instance. Instances remain linked to the symbol; if the symbol is altered, every instance is automatically altered too.

Since interpolated animations, almost by definition, reuse objects, interpolating (or *tweening*, as Flash puts it) the motion of an object turns it into a symbol. You can create tweened motion in several ways. In the simplest, a key frame is selected in the timeline, and an object is drawn on the stage. The command Create Motion Tween is selected from the Insert menu; this sets up the tweening and, as a side-effect, stores the object in the library as a symbol. Another key frame is created at the end of the tweened sequence, and the symbol is moved to a new position in this new frame. The tweening is shown on the timeline as an arrow between the two key frames, as you can see in Figure 8.8, where the two character layers have been tweened to move the creature across the screen in a straight line, as shown in Figure 8.9. (The figure only shows every fourth frame of the anima-

tion.) An animation may be built up as a sequence of automatically tweened segments, between key frames that have been arranged by hand, by repeating this process.

Moving a single symbol about the stage in a straight line offers little in the way of artistic gratification. Tweening can be applied to different layers, with key frames in different places, though, allowing the independent animation of many symbols, each of which may be part of a single character. To further ease the work of animating motion, an object can be moved along a path drawn on a hidden layer; this *motion path* need not be a straight line, so movements that would have to be constructed using many key frames if only rectilinear movements were possible can be achieved in a single tweened sequence with just two key frames. Finally, although the process we have described is referred to as 'motion tweening', an object's size, orientation, opacity and colour may also be interpolated in the same way.

As well as motion tweening, Flash supports *shape tweening*, or *morphing*, as it is commonly known. This is a form of interpolation where the shapes of graphical objects are transformed between key frames, for example, a square can be turned into a circle. In the animation shown at the beginning of this chapter and in Figures 8.10 to 8.12, the motion of the spray and ripples on the sea was created by shape tweening.

There are, in fact, three different sorts of symbol in Flash. *Graphic* symbols are simply reusable vector objects; the symbols created for motion tweening are graphic symbols by default. *Button* symbols are a specialized type of symbol, used for adding interactivity to Flash movies; we will describe them in Chapter 16. *Movie clip* symbols are self-contained animations with their own timelines, that play within the main movie. For example, the dolphins leaping picturesquely out of the water in the animation of the sailing ship were added to the basic animation of the ship, sea and clouds by creating a movie clip symbol of a single dolphin jumping, as in Figure 8.10; instances of this single dolphin symbol were added to the main movie, to make a school of dolphins to accompany the ship on its voyage.

Since movie clip symbol instances have their own timelines, they will continue to play, even when the main movie has been stopped. Figure 8.12 shows that the schools of dolphins carry on leaping, even when the ship has been frozen (by stopping the movie in the player). Compare these frames with the ones at the beginning of the chapter, to see how the compound animation has been built out of the independent movement of its elements. Figure 8.13 shows how the

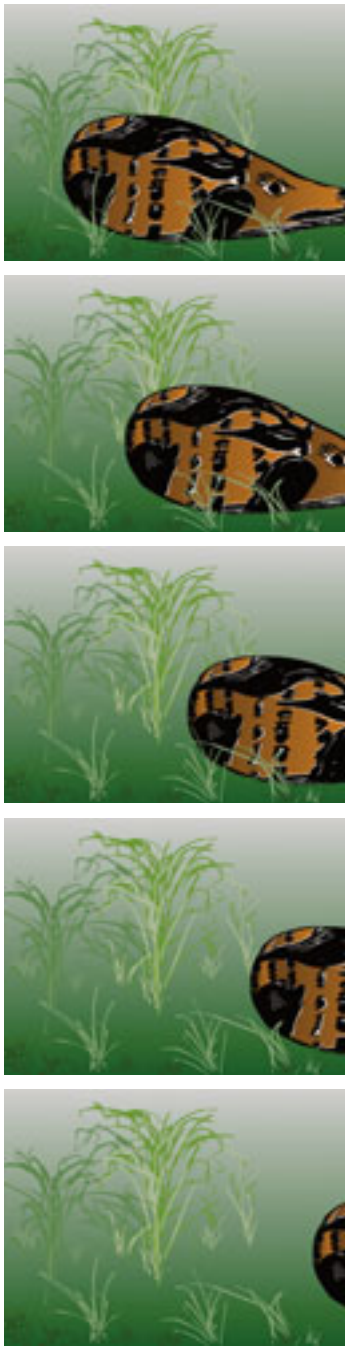


Figure 8.9 A simple animation created by motion tweening

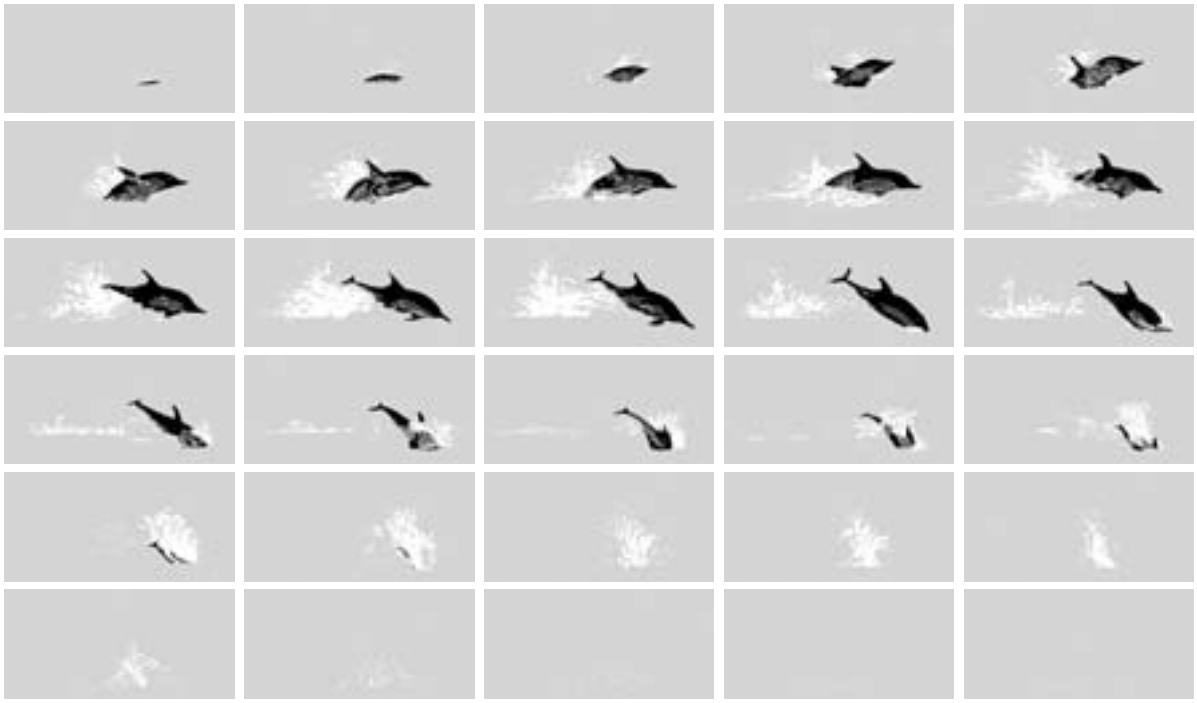


Figure 8.10 Movie clip symbol of a leaping dolphin (superimposed on grey for clarity)

Figure 8.11 Movie clip symbols playing in sync with other action in the main movie

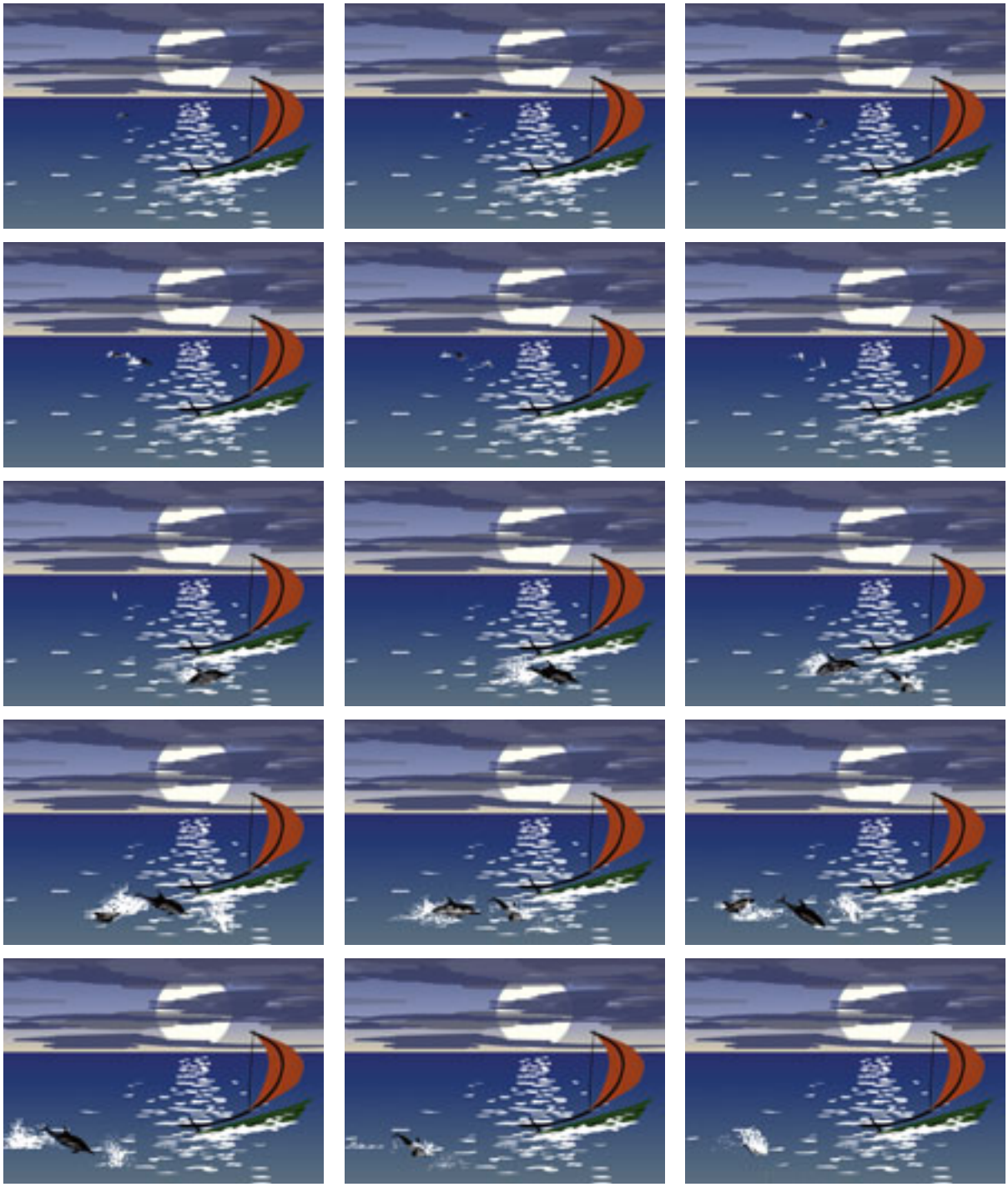


Figure 8.12 Movie clip symbols playing independently of the main movie

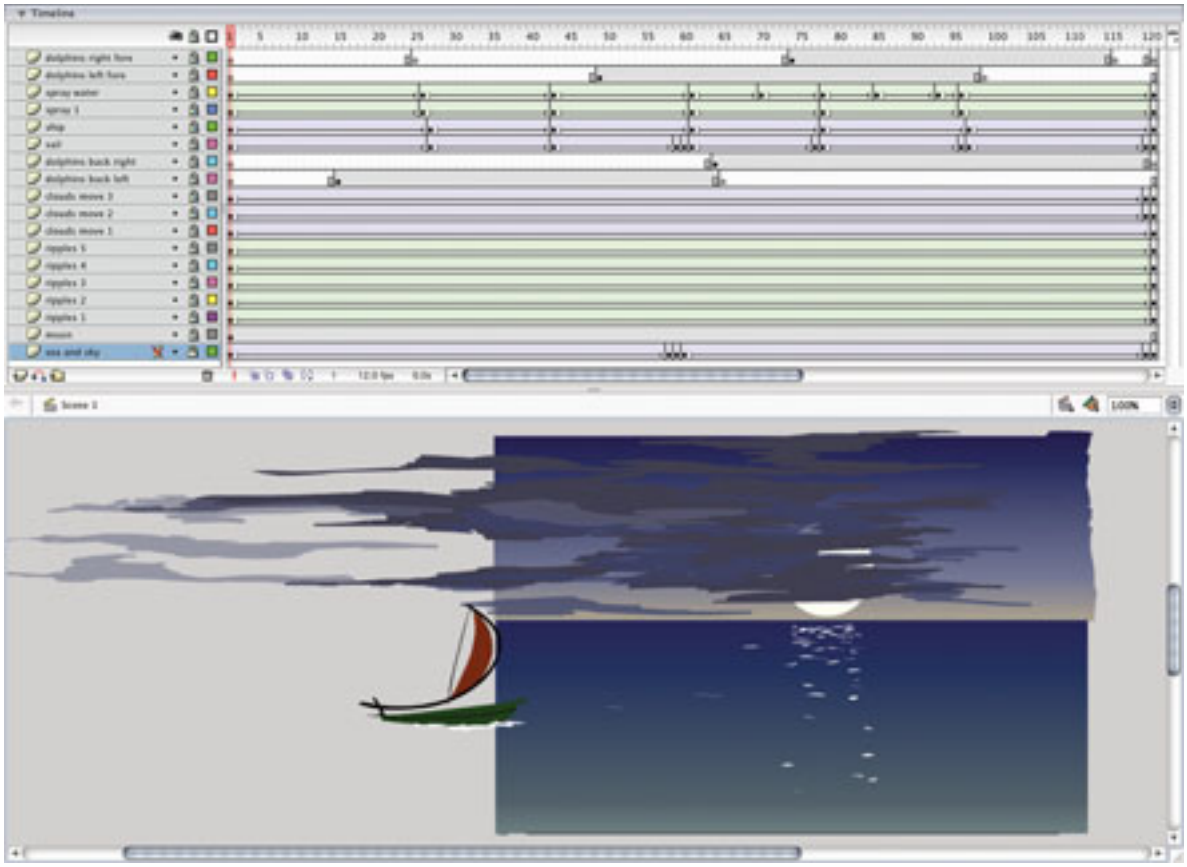


Figure 8.13 Timeline and stage for a complex Flash animation

composition was created out of many layers, with shape and motion tweening, as well as movie clips. Note how the images on the stage extend beyond the frame; tweening causes them to move into shot in the completed animation.

The nature of vector drawing and tweening leads to a compact representation of animations created in this way. An SWF file consists of *items*, which are divided into two broad classes: *definitions* and *control items*. The former are used to store definitions of the symbols used in an animation into a dictionary; the latter are instructions to place, remove, or move a symbol (identified by its name in the dictionary). Placement and movement are specified using transformation matrices, so that the position and any scaling or rotation are specified at the same time. An SWF file is thus rather like a program, comprising as it does definitions of some objects and instructions that manipulate them. SWF data is encoded in a binary form and compressed, resulting in very small files.

Motion Graphics

Interpolation between key frames can be applied to bitmapped images. Since bitmaps do not contain identifiable objects, the use of layers to isolate different elements of an animation is essential. The analogy with cel animations is more or less complete – each layer is like a transparent sheet of acetate with something painted on it. Layers can be moved independently, so an animation can be constructed by placing different elements on different layers, and moving or altering the layers between frames. Where the movement or alteration is easily described algorithmically, it can be interpolated between key frames, just as in-betweeners interpolate between a chief animator's key frames. Typically, between key frames, a layer may be moved to a different position, rotated or scaled. These geometrical transformations are easily interpolated, but since we are now concerned with bitmapped images, they may require resampling, and consequently cause a loss of image quality, as we explained in Chapter 5.

AfterEffects is the leading desktop application for animation of this kind. Because of their shared provenance, AfterEffects works well in conjunction with Photoshop and Illustrator. A Photoshop image can be imported into AfterEffects, with all its layers – including adjustment layers – and alpha channels intact; an Illustrator drawing can be imported and rasterized, again with its layers intact. A common mode of working, therefore, is to use the tools and facilities of Photoshop or Illustrator to prepare the elements of an animation on separate layers, and import the result into AfterEffects where the layers are animated. Photoshop images should be prepared at an appropriate resolution and size for your intended delivery medium. If they are to be scaled down, they must be large enough to accommodate the maximum reduction that will be applied. Illustrator files can be either rasterized when they are imported and then treated as bitmaps, or continuously rasterized for each frame in the animation. This means that if they are scaled, for example, no detail will be lost.

The simplest animations are made by repositioning layers, either by dragging them or by entering coordinates, and interpolating motion between key frames. By combining layers and adding effects and filters that also vary over time, moving graphic designs are obtained. The countdown sequence shown in Figure 8.14 was made by importing a set of still images into AfterEffects and animating them in this way. Apart from the interpolated motion of the complete bitmaps, no moving elements were used. The effects that can be achieved using

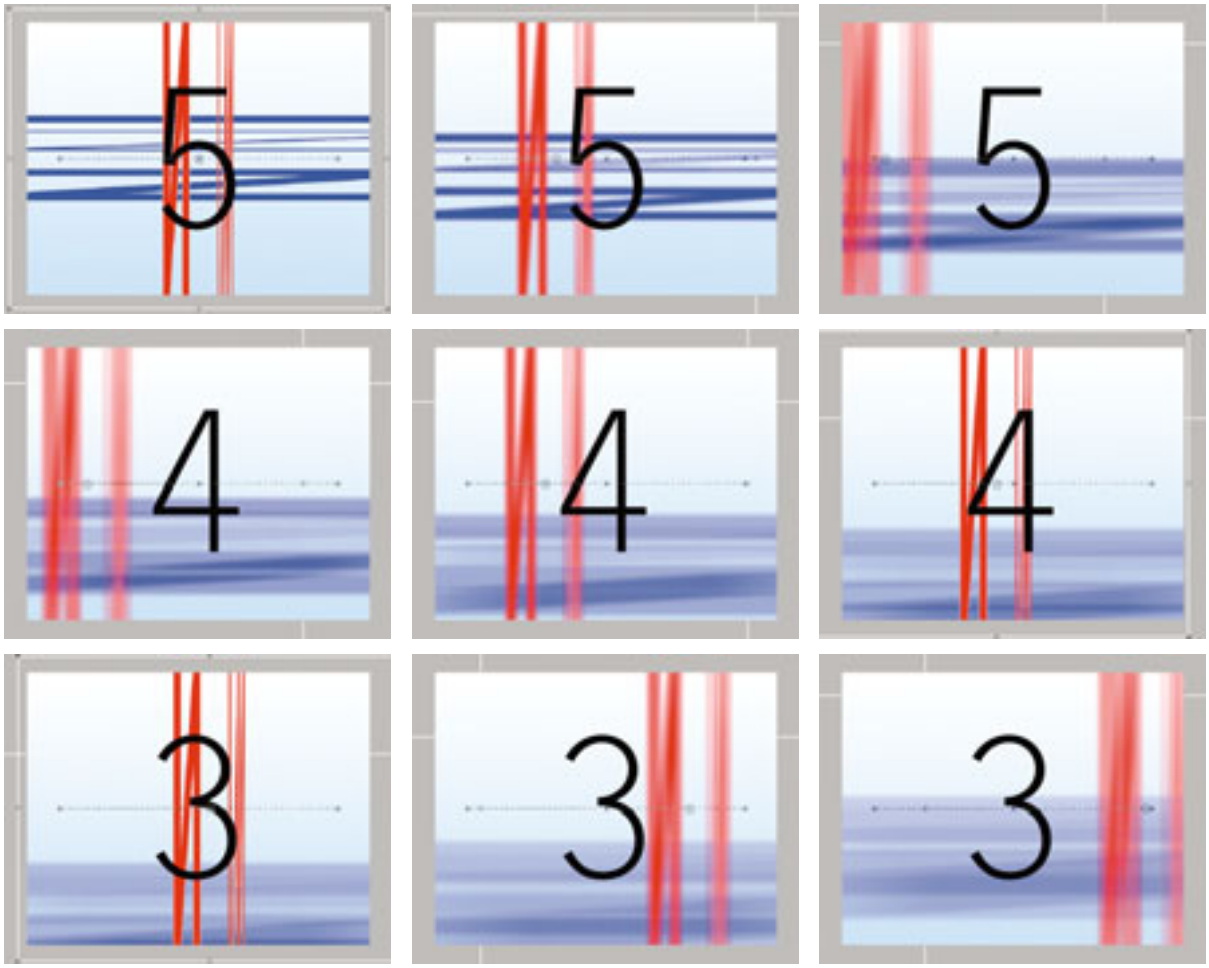
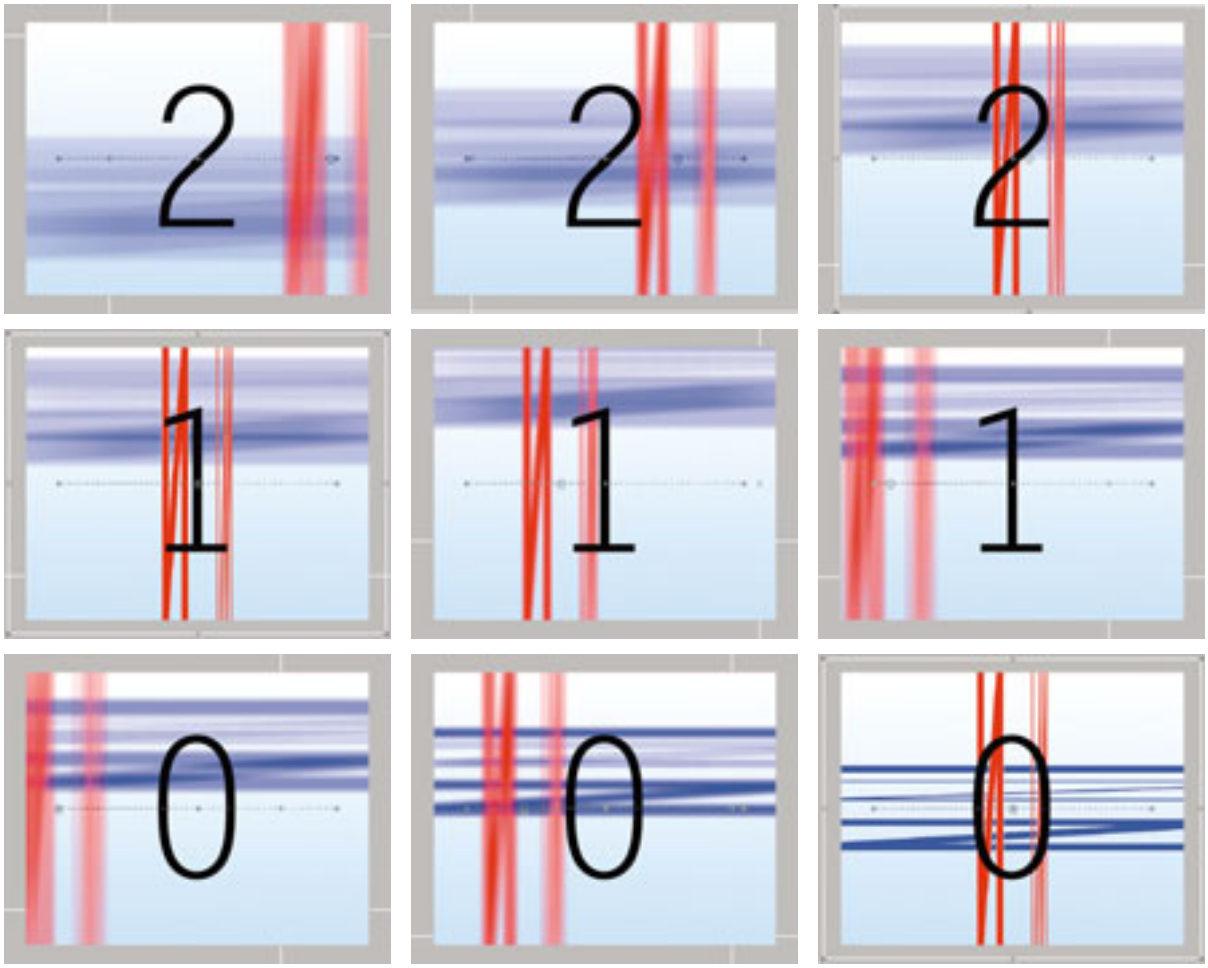


Figure 8.14 Simple motion graphics

motion and time-varying filters on bitmapped images have more in common with graphic design than with mainstream cartoons or art animations. They are often known by the more suggestive name of *motion graphics*. Many of the techniques first appeared in title sequences for feature films, and credit sequences remain a typical application.

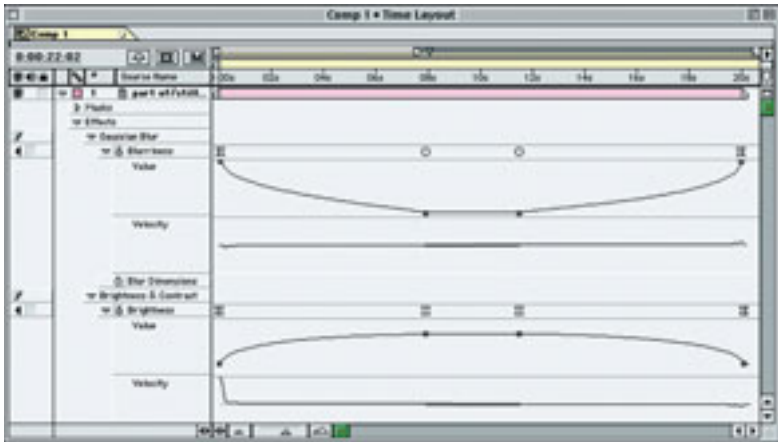
Interpolation can be applied to other properties of a layer. In particular, its angle can be varied, so that it appears to rotate. Angles may be set by hand in key frames and interpolated, or the rotation may be determined automatically in conjunction with movement, to maintain the orientation of a layer with respect to its motion path. Scaling, which may be used as a perspective effect to convey the impression of



approaching or receding movement, or as a zoom in or out, can also be set in key frames and interpolated.

AfterEffects supports both linear and Bézier interpolation, in both space and time. Linear interpolation leads to abrupt changes in direction; with Bézier interpolation the changes in direction are smooth. These are different forms of spatial interpolation, which are set by moving the layer in the window that shows the image. Temporal interpolation affects the rate of change of position with respect to time. Again, this may be linear, with a constant velocity and instantaneous starting and stopping, as discussed earlier in connection with Flash, or Bézier, where the acceleration is smooth. The temporal and

Figure 8.15 Interpolating filters



spatial interpolation methods are independent: you can use linear temporal interpolation with Bézier motion paths, and vice versa.

The degree of control over the interpolation of these spatial properties offered by AfterEffects is considerable. Using a conventional Bézier pen tool, the graphs showing how a value varies with time may be redrawn. Key frames are inserted automatically when control points are added to the graph. Absolute values may be entered numerically, allowing complete control over positioning and the rate of movement. Nevertheless, the type of motion that can be produced by interpolating the position, angle, and size of a single layer is restricted. Objects appear to move as a whole, with an unrealistic gliding motion, resembling that seen in simple, low-budget, cut-out animation, as favoured for some pre-school entertainment and education on television. Key frame animation of bitmapped images is therefore more frequently used for stylized motion. As we mentioned in Chapter 7, travelling mattes are often made by animating a still image in AfterEffects. Another popular application is the animation of text. Individual characters or words can be placed on layers and animated, just like any other layer, or text may be placed on a path, as in Illustrator, and then moved along that path over time.

As our countdown example demonstrates, bitmapped representation allows other properties of the image besides its position, angle and size to be altered over time. So, in addition to geometrical transformations, more radical time-based alterations of the layers can be achieved. As we described in Chapter 5, bitmapped images can be treated with many different effects and filters. Most of these filters have parameters, such as the radius of a Gaussian blur, or the bright-

ness of glowing edges. Such parameters can be made to change over time, using the same mechanism of interpolation between key frames as is used for interpolating motion. Doing so allows some unique effects to be generated.

For example, Figure 8.15 shows a sequence of frames extracted from the title sequence of a short film. The title, 'part of', emerges from darkness as a vague blur, becomes sharper and brighter until it reaches maximum clarity, where it is held for a few seconds before receding back into the darkness. This was achieved by applying a time-varying Gaussian blur to the text, in conjunction with varying the brightness. The actual text was a single still image, made in Photoshop, that was otherwise unaltered. Figure 8.15 also shows the graphs of the Gaussian blur and brightness values that were used. The blur starts with a very high value, which renders the text illegible; in the same key frame, the brightness is substantially reduced. The values are interpolated to a point where the blur is removed and the brightness brought right up. Bézier interpolation is used to ensure a smooth fade up. The values are held constant between the middle two key frames, as shown by the flat portions of the graphs, and then, to make the title fade back into nothing, a symmetrical interpolation is used to a final key frame where the values are identical to those at the beginning.

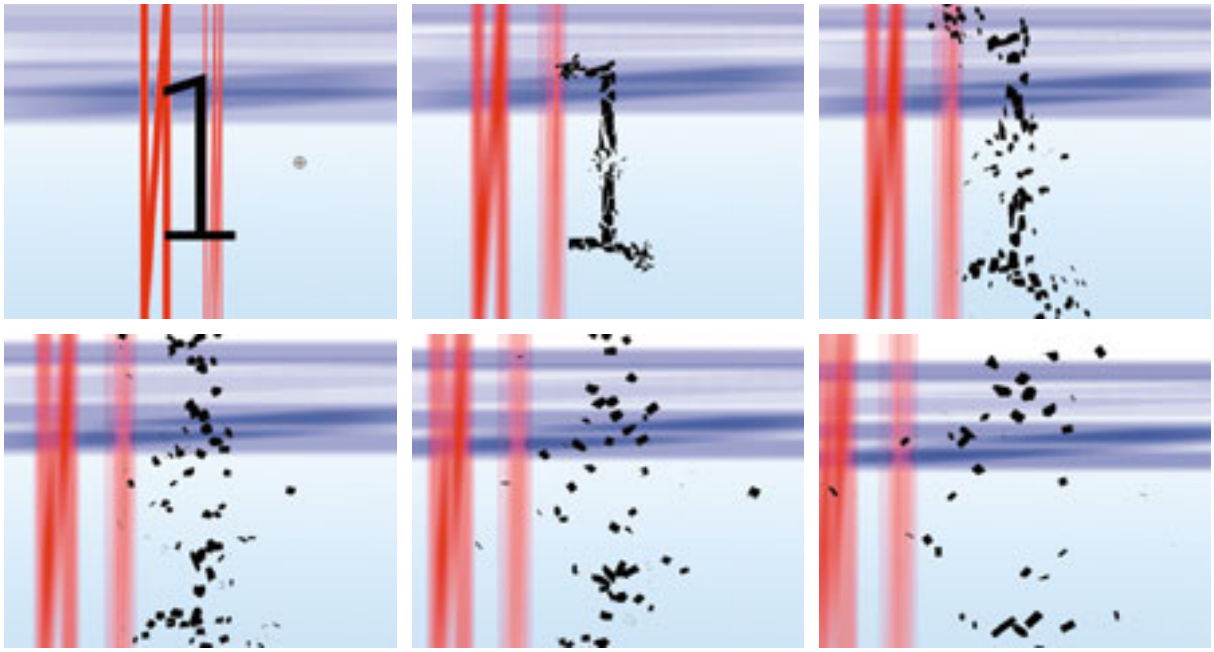
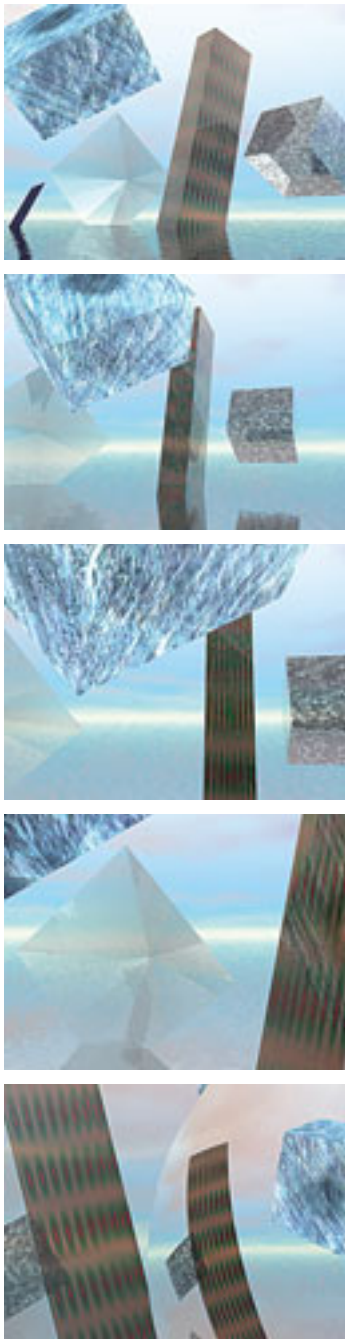


Figure 8.16 A purely temporal effect



As well as varying the parameters of still image filters and effects over time, you can also apply new effects which become possible when a temporal dimension is added to images. Figure 8.16 shows part of an alternative version of the countdown animation, with a shatter effect applied to the numerals.

3-D Animation

3-D animation is easy to describe, but much harder to do. No new concepts beyond those already introduced in this chapter and in Chapter 4 are needed to understand the essence of the process. The properties of 3-D models are defined by numerical quantities. Changing the numbers changes properties such as an object's position in space, its rotation, its surface characteristics, and even its shape. The intensity and direction of light sources and the position and orientation of a camera are also numerically defined. In order to animate a three-dimensional scene, therefore, all that is necessary is to set up an initial scene and render it as the first frame of the animation, make some changes to parameters, render the next frame, and so on. Because the values that are being changed are numerical, some kinds of change can be interpolated between key frames; a timeline can be used as a convenient way of organizing the animation, and motion paths in three dimensions (often 3-D Bézier splines) can be used to describe movement. Because 3-D models must be rendered as 2-D images, which implies the presence of a viewpoint or camera, as well as moving objects in a scene, we can move the camera, making it fly through a landscape or round some objects, as in Figure 8.17.

Whereas simple 3-D animations, such as tumbling logos and rotating globes, really can be made very easily – and there are dedicated packages available for such tasks – high-quality photo-realistic animations, such as those employed in television advertisements, music videos, and film special effects, require huge resources: time, processor power and memory, dedicated software, and above all, highly skilled specialized personnel. Multimedia production can rarely afford these resources. For this reason our description of 3-D animation is limited – readers interested in a fuller treatment of the subject should consult the relevant references given in the bibliography.

There are several factors that make 3-D animation more difficult than it might appear. The first is the difficulty that most people have in visualizing in three dimensions. When we add time, there are four dimensions to be handled through the medium of a two-dimensional computer screen. This difficulty is exacerbated by the second prob-

lem, which is the amount of processing power needed to render a 3-D animation. Advanced shading algorithms, such as ray tracing, take a long time to process a single image. In animation, we need at least 12, and up to 30, images to be processed for every second of completed animation. This makes generating fully rendered previews impossible on anything but the most powerful workstations or networks of distributed processors.

Large budgets, patience and practice can overcome these problems, but what remains is the necessity to change a large number of parameters in such a way as to produce convincing movements. At the very highest level of 3-D computer-generated animation, the solution that is adopted is to provide a rich interface giving the animator complete control over movement. For animating figures, this resembles the type of control used by a puppeteer to control a mannequin – body suits equipped with motion sensors, like those used to control animatronic puppets, are even used sometimes. This is in marked contrast to the approach taken to animation by computer programmers, whose well-trained instinct is to try to automate everything. In an attempt to overcome the limitations of simple interpolation schemes, considerable research efforts have been expended on ways of producing convincing motion in three dimensions automatically.

One of the key approaches is to provide certain kinds of behaviour that can be applied to objects and the way they interact. A simple type of behaviour consists of making one object point at another. This is most useful when the pointing object is a camera or light. If a camera is pointed at an object, it will maintain that object in its field of view, no matter where it moves; a spotlight pointed at an object will automatically follow it, as a real spotlight follows a dancer on a stage, for example. Actual objects in the scene can also be made to point at each other: a sunflower can be made to point always at the sun, for example. A variation on this behaviour is to have one object track another, i.e. follow its motion at a distance. This can be used crudely to animate chase scenes. Like pointing, it can be applied to a camera, allowing it to follow an object or character through a scene, even in places where it would be physically impossible for a real camera to go.

Some 3-D animation systems incorporate behaviour based on the physical laws of motion. For example, they allow the user to specify that an object should accelerate from zero under the influence of an external force whose magnitude is specified as a parameter. Taking

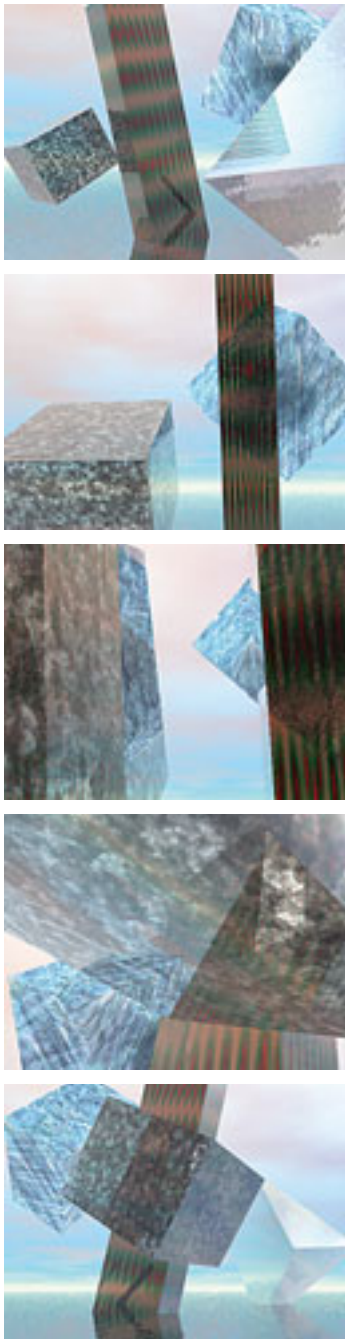
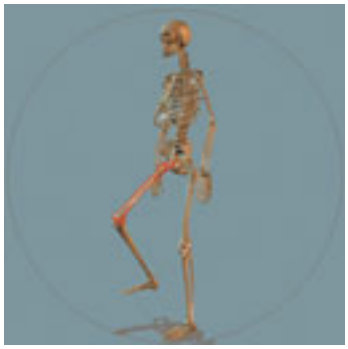


Figure 8.17 Moving the camera around a 3-D scene



this further, moving objects can be made to collide realistically, or bounce off solid surfaces. These behaviours are based on simple laws of physics that encapsulate the possible motion in a few equations. Unfortunately, many realistic types of movement cannot be so easily described, and other methods must be used.

Kinematics is the study of the motion of bodies without reference to mass or force. That is, it is only concerned with how things can move, rather than what makes them do so. In animation, it is most useful in connection with jointed structures, such as the limbs of human or animal figures. Because they are joined together, the various parts of your arm, for example, can only move in certain ways. To produce realistic movement, a 3-D model of an arm must obey the same *kinematic constraints* as a real arm: if the upper arm is raised, the lower arm and hand must come with it, for example. Whereas, in reality, it is the motion of the upper arm that propels the rest of the arm, in an animation system, modelling movement in this way – from the beginning of a chain of connected elements to the end – is not very helpful to the animator. It is more useful to be able to position the object which is at the end of the chain – a hand, say – and then make the other elements – the rest of the arm – move to accommodate it. It is usually the extremities that impose limitations on movement; when walking, a foot must stay above the ground, resting on it at each step, for example. It takes considerable understanding of the way limbs move to ensure that this will happen correctly by moving the thigh, so it is preferable for the software to work out the movements of the leg from the animator's placement of the foot, and so on. This type of modelling is called *inverse kinematics*, since it works backwards from effect to cause. Inverse kinematics can be applied to any structure that is modelled as a linked chain of objects. It is routinely provided by 3-D animation programs that support such structures. Poser, for example, can be set to automatically apply inverse kinematics to the arms and legs of figures.

A little experimentation will show you that computation of movement using inverse kinematics is not entirely straightforward. In particular, the kinematic constraints on arms and legs do not uniquely determine the possible movements that the limbs can make to accommodate movements of the extremities. Try lifting your right hand to touch the tip of your nose with your first finger. How many different ways can your right elbow move while you do so? In order to fix a particular type of movement, extra constraints, such as minimizing the

potential energy of the whole structure, must be added. To produce movements that defy these constraints, while still being physically possible, inverse kinematics must be abandoned, and the parts must be positioned by hand.

Virtual Reality

Originally, the phrase 'virtual reality' was used to describe an immersive sensory experience of a synthetic world. Head-mounted displays, which are sensitive to head movements, are used to project images on the user's eyes, modifying them as the head is moved, so that the user appears to be inside a 3-D world, looking around. Data gloves track hand movements, allowing the display to incorporate an image of the user's arm; haptic interfaces provide tactile feedback, so that users can touch and feel objects in the virtual world. Taken to the extreme, virtual reality of this sort would be the ultimate in multimedia, stimulating all the senses at once.

The high cost of the interface hardware required by immersive virtual reality (together with the understandable reluctance on the part of most of the adult population to immerse their body in strange electronic devices) has confined it to flight and industrial simulations, and specialist games arcades. A more modest vision of virtual reality (VR), as 3-D graphics that can be explored, has evolved. Even this version of VR has not yet achieved widespread acceptance, largely because the heavy demands it makes on processor power lead to disappointing results on desktop computers. Two VR technologies deserve a brief mention, since they can be incorporated in Web pages with some success, and promise to become more important as computer power catches up with the vision of VR enthusiasts.

VRML

The *Virtual Reality Modeling Language (VRML)* was created on a wave of enthusiasm for VR and the World Wide Web in 1994. The intention was to provide a mechanism for distributing virtual worlds over the Internet, using Web browsers as the interface. To this end, VRML was a text-based language, that allowed 3-D objects and scenes to be described in a programming language-like notation. VRML 1.0 did little more; the main additional feature was the capability of embedding hyperlinks in scenes, using URLs. Subsequently VRML 2.0 added support for interactivity, via scripting of the sort to be described in Chapter 16, and allowed for video and audio to be embedded in VRML worlds (so, for example, a television set could be made to show a movie). VRML became an ISO standard in December 1997.

Figure 8.18 'Knee bone's connected to the thigh bone...'

VRML allows the specification of objects, in terms of their geometry (whether they are a cube, cylinder, sphere, and so on) and the material of which they are composed. Textures can be mapped onto the surfaces of objects, which can be placed in 3-D space using transformations. Scenes can be lit in a variety of ways, by specifying the type and position of light objects. The basic language thus provides a way of describing the sorts of scenes that can be constructed with conventional 3-D modelling programs, although it lacks some of the more elaborate features, such as NURBS and metaballs. The description is explicit: for example, a terrain might be modelled as an elevation grid, a type of VRML object that specifies a set of points, forming a grid, each at a different height. In VRML, the dimensions of the grid and the height of each grid point must be explicitly specified. Constructing VRML scenes by hand is thus a painstaking and error-prone business. Most 3-D modellers will generate VRML as output from their normal interactive modelling tools, however, which provides an easier way of constructing scenes.

It might appear that VRML is no more than an alternative representation of the output of a modeller, and, as far as the language goes, this is more or less the case. It does, as we noted earlier, provide additional features for interactivity and embedding multimedia, but the main distinctive feature lies not so much in the language as in the way it is displayed. Once a VRML file has been downloaded into a suitable browser – either a Web browser with an appropriate plug-in, or a dedicated VRML browser – the user can explore the 3-D world it describes. That is, they can move the viewpoint through the space of the model, as if they were moving about in it. To that extent, VRML deserves to be considered a form of virtual reality.

To create the illusion of moving through a 3-D space, VRML must be rendered in real-time. As we have stated several times, realistic rendering of 3-D models is a computationally intensive task, which is usually only feasible with special hardware, such as 3-D accelerator cards. This is one of the main obstacles to the widespread use of VRML, although a lack of commitment to the format by major software vendors may be more significant. At the time of writing, a host of competing – mostly proprietary – formats for delivering 3-D models over the World Wide Web is available, with none, as yet, achieving any widespread use.

QuickTime VR

QuickTime VR (or *QTVR*, for short), part of QuickTime, offers a very basic VR experience. There are two types of QuickTime VR movies: panoramic movies and object movies. The former presents a 360° view of a scene – the interior of a room, or a valley surrounded by mountains, for example. Using their mouse, a user can drag the scene, as if looking around. It is also possible to zoom in or out, in order to view the scene in more or less detail. Object movies, in contrast, allow the user to examine an object from different angles, as if by walking round it, again by dragging with the mouse. QTVR movies of either sort may contain *hot spots*, which are active areas that contain links to other movies. Clicking on a hot spot causes the linked movie to replace the current one. A typical use of hot spots is to allow a user to go through a door from one room into another.

QTVR movies can be generated from some 3-D programs, such as Bryce. They can also be made from photographs, allowing them to represent real scenes and objects. To achieve the best results, a special rotating rig is used to hold the camera (for panoramas) or an object. A sequence of pictures is taken, with the rig being rotated a fixed amount between each picture. These are then scanned (or read in to the computer if a digital camera is being used) and special software 'stitches' them together and renders the result as QTVR. The purpose of the rig is to ensure that the individual pictures fit together perfectly. If an ordinary tripod is used for panoramas, or a record turntable or similar device for object movies, there may be discontinuities; stitching software will attempt to compensate for these, but the result may be distortion of the images.

Since QTVR is part of QuickTime, panoramas and object movies can be combined with audio and video. Most usefully, they can be viewed by any software that uses QuickTime; in particular, the QuickTime plug-in for Web browsers allows QTVR to be embedded in Web pages, in the same way as video clips can be (see Chapter 12).

QuickTime VR and VRML might be considered travesties of the original vision of immersive virtual reality, but they have the advantage of being implementable without special interface devices or powerful workstations. They offer the possibility of new approaches to interfaces to multimedia, based on the organization of media in three-dimensional spaces.

Exercises

1. What are the advantages and disadvantages of using a scanner or a digital stills camera to capture traditional art work as animation sequences? For what types of animation, if any, would you have to use a video camera connected to a computer?
2. How could you incorporate drawn animation into a live-action video sequence without using a special effects program?
3. If an animation sequence is to be saved as QuickTime, what factors will influence your choice of codec? Under what circumstances would it be appropriate to treat the animated sequence exactly like a live-action video sequence?
4. When would it be appropriate to use an animated GIF for an animation sequence? What problems are associated with animated GIFs?
5. In what ways is a sprite animation track radically different from a video track containing animation?
6. For what type of work would sprite animation be particularly suitable and why?
7. What problems are associated with using basic linear interpolation to do 'in-betweening' between key frames?
8. How would you use Flash's easing facility to set up a movement that eases in *and* out?
9. Describe the motion of an object whose position is animated in AfterEffects using Bézier interpolation for the motion path, and linear interpolation for the velocity.
10. Create a very simple title for a video clip as a single image in a bitmapped graphics application such as Photoshop or Painter, and save it as a still image file. Using whatever tools are available (Premiere, AfterEffects, etc.), create a pleasing 10-second title sequence by simply applying time-varying effects and filters to this single image. (If you want to go for a more sophisticated result, and have the necessary tools, you might create your original image on several layers and animate them separately.)
11. Do the models generated by 3-D applications contain enough information to be used in conjunction with a haptic interface to provide tactile feedback to a user? If not, what extra information is needed?