

SOFTIMAGE[®] | 3D

Animating

A Comprehensive User's Guide

The *SOFTIMAGE 3D User's Guide* was written by Susan Duciaume, Camille Lemberg, Suzanne Gerrior, Grahame Fuller, and Edna Kruger; edited by Edna Kruger; graphics created by Richard Z'Graggen; and layout by Paul Verrall.

Special thanks to Rejean Gagne, Lucy Boyd-Wilson, Lafleche Dumais, Yves Boucher, Richard Lampron, Gordon Cameron, Glen Frasier, and Eric Cabot for their assistance in assuring the technical integrity of this guide.

© Copyright 1996 Softimage Inc., a wholly owned subsidiary of Microsoft Corporation. All rights reserved.

Microsoft®, MS®, MS-DOS®, Windows®, and Windows NT® are registered trademarks of Microsoft Corporation in the United States and/or other countries.

SOFTIMAGE® is a registered trademark of Softimage Inc., in the United States, Canada, and/or other countries.

3D Studio Max™ is a trademark of Autodesk, Inc.

Adobe Illustrator® is a registered trademark of Adobe Systems, Inc.

Abekas® is a registered trademark of Abekas Video Systems, Inc.

Alias® is a registered trademark of Alias Research Inc.

Apple® and Macintosh® are registered trademarks of Apple Computer, Inc.

Aurora® is a registered trademark of Chyron Corporation.

AutoCAD® is a registered trademark of AutoDesk, Inc.

Flock of Birds® is a registered trademark of Ascension Technology Corp.

Indigo® is a registered trademark and Indy™ is a trademark of Silicon Graphics, Inc.

Magellan™ is a trademark of Logitech, Inc.

Macromedia® is a registered trademark and Freehand™ is a trademark of Macromedia, Inc.

PostScript® is a registered trademark of Adobe Systems, Inc.

SEGA Saturn™ is a trademark of Kabushiki Kaisha Sega Enterprises.

Targa® is a registered trademark of Truevision Inc.

TIFF™ is a trademark of Adobe Systems, Inc.

UNIX® is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company, Ltd.

V-LAN® is a registered trademark of Videomedia, Inc.

Wacom® is a registered trademark of Wacom Co., Ltd.

WSD® is a registered trademark of Accom, Inc.

All other product names mentioned in this guide may be trademarks or registered trademarks of their respective companies and are hereby acknowledged.

This document is protected under copyright law. The contents of this document may not be copied or duplicated in any form, in whole or in part, without the express written permission of Softimage Inc. This document is supplied as a guide for SOFTIMAGE 3D products. Reasonable care has been taken in preparing the information it contains. However, this document may contain omissions, technical inaccuracies, or typographical errors. Softimage Inc. does not accept responsibility of any kind for customers' losses due to the use of this document.

Document No. 90438-0596

Printed in Canada.

Table of Contents

CHAPTER ONE

Introduction	1
Introduction to Animation	3

CHAPTER TWO

Creating Keyframes	5
Introduction	7
Keyframing	8
Function Curves	12
Before You Start Animating	15

CHAPTER THREE

Playing an Animation	17
Introduction	19
Using the Time Line	20
Controlling the Overall Animation	22
Previewing the Animation	25
Viewing a Flipbook	26

CHAPTER FOUR

Editing an Animation	27
Introduction	29
Editing Function Curves	30
Editing with the DopeSheet Window	53
Using Ghost Frames	64
Using the Spreadsheet Window	65

CHAPTER FIVE

Animating on a Path	67
Introduction	69
Assigning an Object to a Path	70
Substituting a Path	72
Changing the Translation's Timing	73
Converting to and from Paths	74

C H A P T E R S I X

Plotting an Animation 75

Introduction 77

Recording an Object's Values 78

Plotting Trajectories.....79

Plotting Explicit Translation 81

Plotting Rotation.....83

Plotting Shape.....84

Generating Velocity and Acceleration Function Curves 85

C H A P T E R S E V E N

Deformations 87

Introduction 89

Shape and Cluster Animation 91

Animating with Lattices.....100

Animating Curve and Surface Deformation.....105

Quick Stretch 109

Making Waves 119

Deforming Using Custom Effects.....124

C H A P T E R E I G H T

Kinematics.....127

Introduction 129

Anatomy of a Skeleton 130

Animating with Inverse Kinematics 133

Constraining Joints and Effectors.....135

Controlling the Orientation of a Chain 137

Animating with Forward Kinematics 143

C H A P T E R N I N E

Dynamics.....145

Introduction 147

Animating with Dynamics 148

Assigning Physical Properties.....150

Setting Up Dynamics Properties.....153

Adding Physical Forces 154

Creating Dynamic Simulations	158
Creating Obstacles for Collisions	159
The Bouncing Ball	163

C H A P T E R T E N

Constraints	167
Introduction	169
Constraining Object Centres	171
Constraining the Rotation	172
Constraining by Direction	173
Constraining the Scaling	179
Constraining to a Plane	180
Constraining between Clusters and Objects	182
Constraining between Points	185
Creating Constraint Limitations	188
Creating Custom Constraints with Expressions	191
Using Constraints with Inverse Kinematics	193
Using Constraints with Channel Drivers	194
Removing a Constraint	195
Activating and Deactivating Constraints	196

C H A P T E R E L E V E N

Expressions	197
Introduction	199
Writing and Editing an Expression	200
Copying an Expression	202
Creating a Custom Constraint	203
Editing Function Curves	204
Using Expressions with Channels	211

C H A P T E R T W E L V E

Motion Capture and Control	213
Introduction	215
Overview of Channel Connections	216
Selecting and Assigning a Channel Driver	218
Types of Motion Capture Devices	226
Capturing Motion in SOFTIMAGE 3D	229

Saving a Channel Setup File.	237
Copying Function Curve Connections	238
Loading Motion Capture Files	239
Calibrating your Animation Data	241
Controlling Animation and Devices	246
Synchronizing your Animation with Sound	249

C H A P T E R T H I R T E E N

SOFTIMAGE Live	251
Introduction	253
Getting Started	254
Viewing Scenes	256
Controlling Scene Display	259
Animation and Channels	263
Creating a Virtual Set with Real and Computer-Generated Actors	267

C H A P T E R F O U R T E E N

Particles and Custom Effects	277
Introduction	279
Creating Particles with Flock	280
Creating Particles with Custom Effects	284
Animating Custom Effects	286

C H A P T E R F I F T E E N

Animation Files	289
Introduction	291
Saving a Structure Key	292
Retrieving a Structure Key	293

I N D E X

C H A P T E R O N E

Introduction

Introduction to Animation

In SOFTIMAGE 3D, you can animate almost anything. However, the method you choose can vary greatly. In traditional animation, keyframes are drawn showing the animated subject at a critical or extreme moment of a given action. The intermediate frames are then filled in afterward to complete the action. This is called *in-betweening*. The process of keyframing in SOFTIMAGE 3D follows the same principle as traditional keyframing, with one important difference: once you define the keyframes for the animated action, SOFTIMAGE 3D automatically calculates the in-between frames for you.

When you keyframe objects in your scene, SOFTIMAGE 3D creates function curves for each of the parameters you have animated. A function curve, or *fcurve*, is a mathematical representation of the parameter's change over a period of time. You can change your animation by editing the function curves.

You can animate objects and articulated chains using conventional keyframing of translations, rotations, and scaling or by constraining it to another animated object that you translate, rotate, or scale. You can apply inverse kinematics to 2D and 3D chains. Inverse kinematics is a powerful tool, in which you define the goal of the chain, and SOFTIMAGE 3D performs the calculations to create the animation.

You can also use other methods, such as mathematical expressions, plotting, dynamic simulations, and motion capture to animate your objects. When you use these methods, you may not actually need to keyframe your animation because SOFTIMAGE 3D generates the animation based on the values that you have entered. For example, when you create a dynamic simulation, you are specifying the density of the object, the type of dynamic simulation, any physical forces to apply to the object, and the start and end frame of the animation. Once you have entered these variables, SOFTIMAGE 3D generates the animation. Of course, you could achieve these results through traditional keyframing, but it would take much longer.

You can apply deformation techniques to change the geometry of an object over time. Deformation is a powerful modelling tool, made more powerful because you can also animate the various types of

deformation, such as shape and cluster animation, lattice deformation, curve and surface deformation, Quick Stretch, and wave effects.

Here's an example of some of the animation capabilities at your disposal: assume that you want to create a realistic animation of an autumn leaf falling from a tree branch. You can animate the colour of the leaf to change gradually from green to yellow. You can separate the leaf from the tree, record the changing position of the leaf as it falls, and animate the changing shape of the leaf as if it was fluttering through the air. The position of the camera can be changed to follow the leaf's descent. After the leaf strikes the ground, you can change its texture to look like it is gradually dried up and wrinkled, and have its colour fade from yellow to brown. Finally, to simulate the setting sun, the lights can be dimmed until the scene fades to black.

C H A P T E R T W O

Creating Keyframes

Introduction

In traditional animation, *keyframes* are drawn showing the animated subject at a critical or extreme moment of a given action. The intermediate frames are then filled in afterwards to complete the action. This is called *in-betweening*.

The process of keyframing in SOFTIMAGE 3D follows the same principle as traditional keyframing, with one important difference: once you define the keyframes for the animated action, SOFTIMAGE 3D automatically calculates the in-between frames for you. This is called *interpolation*. SOFTIMAGE 3D interpolates the animation parameters you set at each keyframe, generating a function curve. This is the most fundamental animation technique in SOFTIMAGE 3D.

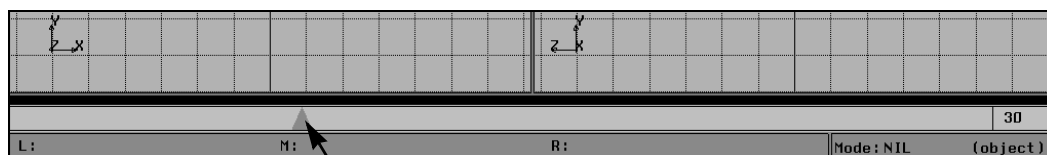
You can keyframe anything in SOFTIMAGE 3D. For example, you can keyframe the visibility of an object, or its textures, or materials. Define the parameters of the object (or the light, the camera, etc.) you want to animate first, define the time position of the keyframe, and then set the keyframe with a **SaveKey** command. You can also define the time position of the keyframe first, then fill in the animation values.

Keyframing

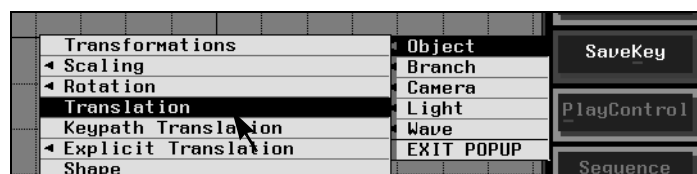
You can animate almost any parameter of any object, materials, textures, lights, and the camera in SOFTIMAGE 3D using keyframing.

To keyframe an object, follow these basic steps. In the example, the translation of an object is being keyframed:

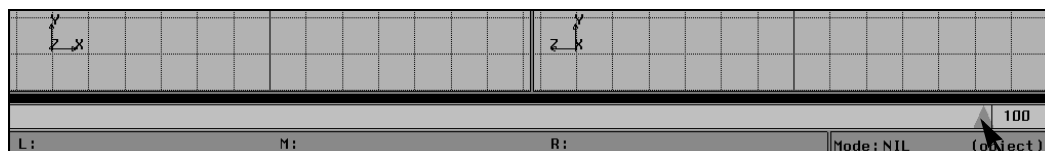
1. Advance to the frame of the sequence where you want an animated action to begin using the time line pointer in the time line at the bottom of the display.



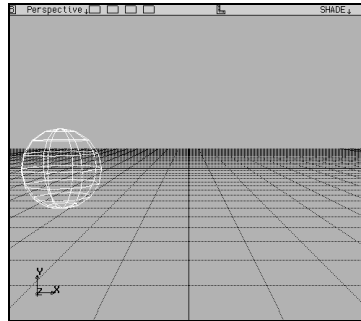
2. Select one or more elements (a light, the camera, an object, etc.) in the scene to represent the initial state or starting point of the action you want to animate.
3. Save a keyframe for the current frame using an appropriate **SaveKey** command, such as **SaveKey → Object → Translation**, to keyframe the element's translation.



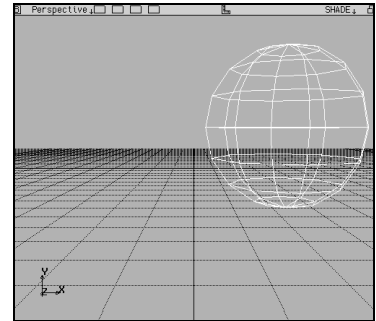
4. Using the time line pointer, advance to the frame of the sequence where you want the animated action to end.



5. Modify the parameters of the selected element to represent the end state of the animated action. For example, here the object is scaled to twice its size, and translated (moved) it to the right.



Frame 1



Frame 100

6. Save a keyframe for the current frame using the same **SaveKey** menu cell command as you chose before. A shortcut for saving the same type of keyframe is to middle-click on the **SaveKey** menu cell (this repeats the last used command).



7. Click the forward icon in the playback box to view the animation.

After keyframing, as you move the pointer along the time line, the scene is updated to follow the in-between values generated for the intermediate frames.

You can, however, override the update by moving the time line pointer with the right mouse button. This is useful for copying the key positions of a model from one frame to another – you simply find a keyframe with the key position you want to copy, advance to the frame where you want the key position to be copied using the right mouse button, and save the new keyframe with the **SaveKey** command.

Translation versus Explicit Translation

When you use the **SaveKey** → **Object** → **Translation** or **Keypath Translation** commands to animate your object, only one function curve is created. This function curve represents the movement of your object as a percentage (%) of the distance travelled along a path.

You can also animate the translation of an object (or camera, light etc.) with explicit translation. When you choose to keyframe explicit translation, the translation of the keyframed object is defined by its

explicit x, y, and z local coordinates in Cartesian space, not a percentage of distance as given with translation or keypath translation.

To save explicit translation keyframes, use the **SaveKey → Object → Explicit Translation** command in the Motion module. You can choose to keyframe only the X, Y, or Z axis of an object, or choose **All** to keyframe all the axes at once.

Animating the rotation and scaling of an object works very much the same way as explicit translation. The object's rotation or scaling can be keyframed along one axis or all three axes at once. Although scaling and rotation are not called explicit, they are by default.

Setting the Time Position First

Another method of keyframing is to set keyframes first, then fill in the values after. This is a slight twist on the typical keyframing sequence of changing the parameters, then saving a keyframe. For example, you may want to use this method to bring an animation file into SOFTIMAGE 3D. This method works exclusively when keyframing scaling, rotation, and explicit translation values.

1. Define the location of keyframes at predefined time positions on the time line, using the appropriate **SaveKey** command (**Scaling**, **Rotation**, or **Explicit Translation**).
2. Choose the **Preferences → Animation** command. The Animation Preferences dialogue box appears.
3. Select the **No Confirmation Required** key editing option. This permits you to make one of two operations:
 - You can edit the keyframe without having to choose the **SaveKey** menu cell. In other words, you can move to a keyframe (which will be coloured) using the time line pointer, then edit the position, rotation, or explicit translation of the element at the keyframe position, and the changes you make automatically update the function curve without having to click the **SaveKey** menu cell.
 - You can load in predefined structure keys using the **Get → From Library → Structure Keys** command. The keyframes you set in step 1 will automatically be updated with the animation data from the saved structure keys. For more information on Structure Keys, see *Animation Files* on page 289.

4. Select the **Coloured Keyframe Feedback** option. This changes the wireframe colour of the object (or the light, the camera, etc.) at keyframes to better indicate which frames are keyframes.

Note

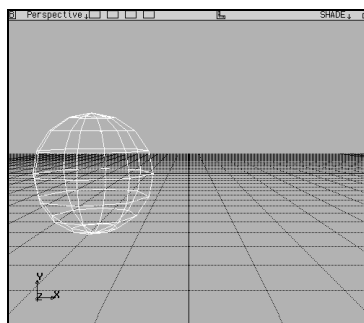
Since the time line pointer must be on a keyframe to allow you to actually change the keyed parameters of an object, use the K button in the playback box to jump from keyframe to keyframe. When you are not on a keyframe, no changes in parameters take effect!

Function Curves

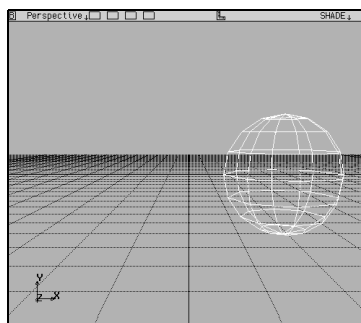
Most of the animation methods mentioned in the introduction of this guide on page 3 generate a function curve for each parameter you animate. A function curve, or *fcurve*, is a mathematical representation of the parameter's change over a period of time. Essentially, the function curve is like a plotted line graph which describes the animation data. The function curve can then be edited, which gives you a great deal of control over your animation – you can change the shape of the curve and SOFTIMAGE 3D automatically recalculates the animation data, producing a different animated effect. This is the basis for most animation in SOFTIMAGE 3D.

For more information about editing function curves, see *Editing an Animation* on page 27.

Consider a simple animation of an object moving from the left to right, like this:

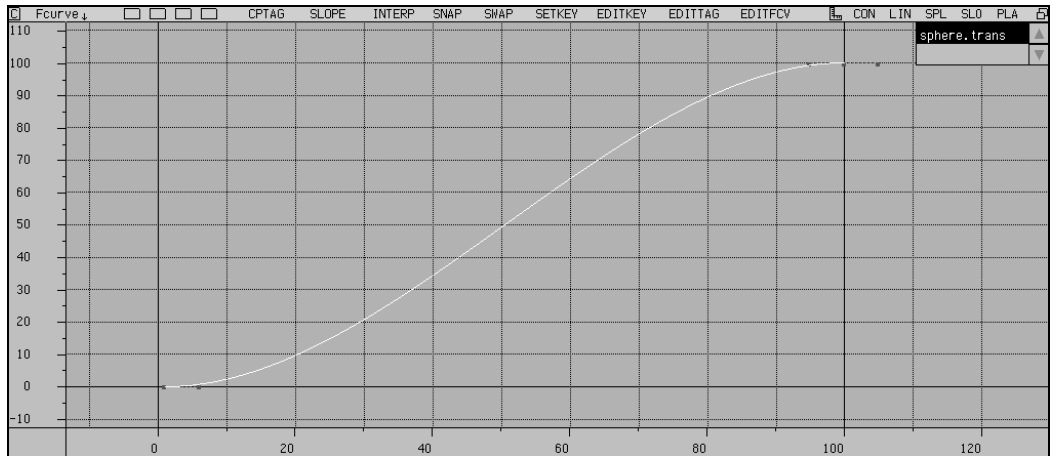


Frame 1



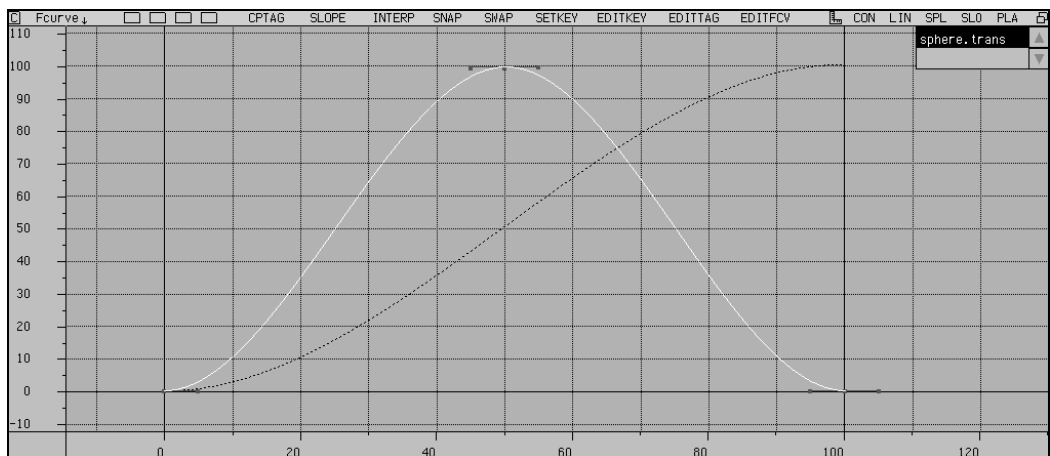
Frame 100

This is one kind of function curve that can be generated from that motion in SOFTIMAGE 3D:

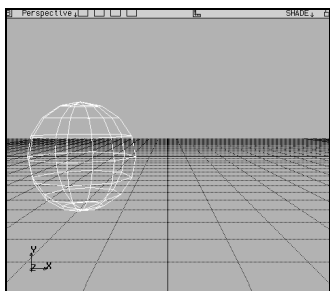


The upturn at the beginning of the curve represents the gradual acceleration of the object from a resting position. The straight segment of the curve shows the object travelling at a constant speed. The levelling off at the top of the curve represents the gradual slowing down of the object until it reaches a dead stop. This process is called *ease-in* and *ease-out*.

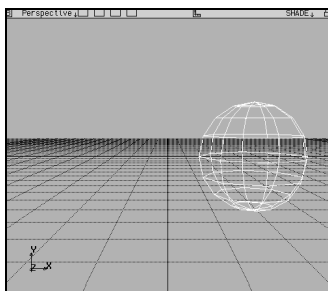
By experimenting with the shape of the function curve, you can alter the speed and direction of the object. For example, if you edited the previous function curve to look like a bell curve:



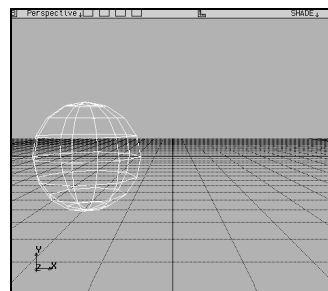
the resulting animation, moves the object to the right, then moves it back toward the starting point, like this:



Frame 1

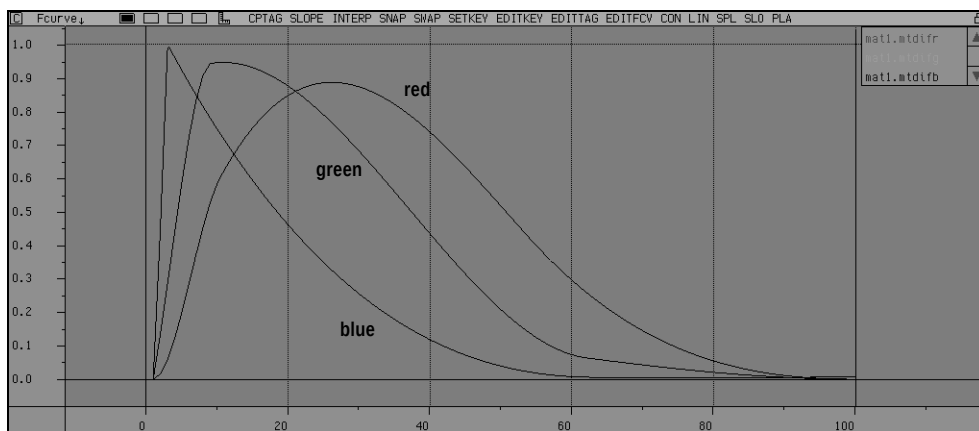


Frame 50



Frame 100

Function curves are also created for parameters that animate properties other than movement of the objects in your scene, such as the material colour of an object. The function curves below represent the red, blue, and green values of an object's material as they change over time. The values range from 0 to 1 and correspond to the values that you set when you keyframed the object's colour.

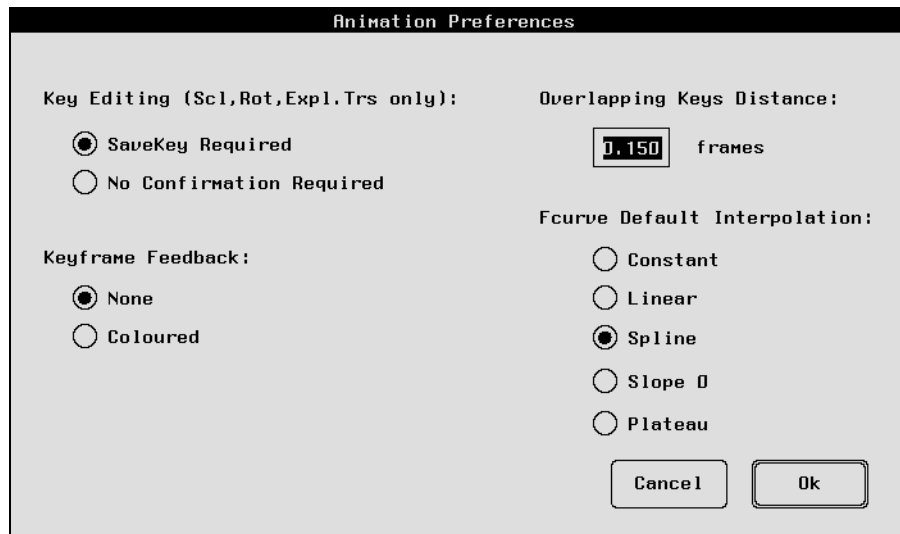


You can change the colour of the object over time by changing the function curve key point values.

Before You Start Animating

In the **Preferences → Animation** command, you can set parameters to create and modify an animation more efficiently. For example, you can select a colour keyframe feedback option to quickly identify keyed objects, or change the default interpolation that SOFTIMAGE 3D uses between key points on the function curve.

1. Choose the **Preferences → Animation** command. The Animation Preferences dialogue box is displayed.



2. Choose the desired options, such setting as the default function curve interpolation, and using coloured keyframes, then click **Ok** to confirm the settings and exit the dialogue box.

See page 10 for an example of how you can set the **No Confirmation Required** option to allow you to automatically update key values of a modified animation.

For more information about the parameters available in the Animation Preferences dialogue box, see page 1154 of the *Reference Guide*.

C H A P T E R T H R E E

Playing an Animation

Introduction

There are many ways to play back the animation that you create. The easiest way to view your animation is to use the time line located at the bottom of your display to quickly view the animation.

For a more accurate view, you can also render a scene and play it back with the flipbook. As in traditional animation, a flipbook lets you get a better idea of how your final animation will look. This allows you to view your animation with all the textures, materials, etc. applied to the objects in your scene.

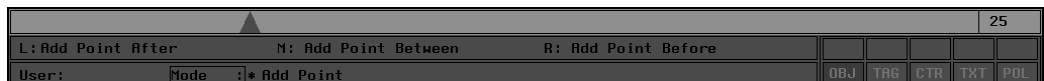
Depending on your needs, you may want to view a wireframe or fully rendered playback. You can control how your animation plays back by editing settings in the **PlayControl** command in the Motion module. You can also change the offsets and scale of your animation by using the **Sequence** command in the Motion module. These commands allow you to specify the speed, frame rate, etc. that is used when you play back your animation, or to scale and offset the animation. The ability to scale and offset is especially useful if your animation is too long or too short, or you simply want to add frames at the beginning of your animation, before any existing keyframes.

There are also a number of tools for other ways of viewing scenes and animation sequences. For more information, see the *Using Tools User's Guide*.

Using the Time Line

The time line is part of the status bar located at the bottom of the SOFTIMAGE 3D display.

The top line in the status bar is the time line. Its width represents the length of an animation sequence. The triangular pointer (time line



pointer) can be moved back and forth along the time line causing the software to display the animated sequence in the windows. To move the pointer, either click anywhere along the time line to move it to that position, or drag the pointer with the mouse.

Dragging the pointer while holding down the left mouse button updates the sequence frame by frame.

Dragging the pointer while holding down the right mouse button lets you move to a different frame *without* updating the scene (the currently displayed frame remains in the windows). This is especially useful if you want to quickly copy a keyframe to another frame in the animation sequence. To do this, go to the keyframe you want to copy, drag the pointer with the right mouse button to the frame to which you want to copy the animation, and then choose a command from the **SaveKey** menu at that frame.

To stop playback at any time, click any mouse button.

Using the Playback Box

The playback box allows you to view animation using various techniques. It is located at the right end of the time line.

Playing Forward and Backward



Forward arrow plays back the scene from starting to ending frame according to the settings defined in the **PlayControl** menu cell. Click the forward arrow to start. By middle-clicking, you can play your animation forward from the current frame.

To stop playback, click the forward arrow again.



Backward arrow plays back the scene from ending to starting frame according to the settings defined in the **PlayControl** menu cell. Click the backward arrow to start. By middle-clicking, you can play your animation backward from the current frame.

To stop playback, click the backward arrow again.

Note

Real time playback is achieved by setting the frame step to zero in the **PlayControl** menu cell. However, with a complex scene, frames are skipped to maintain real time. For more information, see *Defining the PlayBack* on page 22 and *Playing Back Quickly* on page 23.



F lets you play the animation one frame at a time. Click F to activate it and click either the backward or forward arrow. The next frame displayed depends on the frame-step setting in the **PlayControl** menu cell. To deactivate it, click “F” again. You can use the right mouse button with the F button to move one frame at a time without updating the scene or the left and right arrow keys.



K lets you play the animation one *keyframe* at a time. Click K to activate it and click either the backward or forward arrow. The current frame moves to the next/previous keyframe for the selected object(s). You can use the right mouse button with the K button to move one keyframe at a time without updating the scene.

Note

When the Fcurve window is open, only the keyframes for the selected function curves are played.



Setting Frame Numbers

S text box sets the starting frame number. Click in the box to insert a number. Middle or right-click to replace the starting frame number.



E text box sets the end frame number. Click in the box to insert a frame number. Middle or right-click to replace the end frame number.

Looping the Animation



Click L to activate the loop mode and then click either the backward or forward arrow. By middle-clicking the playback arrows, you can play the animation from the current frame and then loop it from the start frame to the end frame (or end frame to start frame if played backwards). By right-clicking the playback arrows, you can play back your animation in a “flip-flop” manner which alternates between forward and backward.

To deactivate the loop option, click L again.

Controlling the Overall Animation

You can control how your animation plays back by editing settings in the Playback Controls dialogue box. You can also activate the **Fast Playback** mode in the window's Layout dialogue box. When this mode is set, the wireframe display is played back faster for the selected window. For information on setting these options, see the previous page and page 30 of the *Reference Guide*.

You can also use the **K** button in the playback box to view your animation by the selected object's keyframes. If the Fcurve or DopeSheet windows are open, only the animation generated by the selected function curves is played.

Defining the PlayBack

The **PlayControl** menu cell allows you to set parameters to define the playback of a scene. You can enter the start frame, end frame, and the frame step. You can also insert a pause between frames, and set the playback speed.

Note Playback control parameters are saved with the scene. Also, the renderer does not consider frames outside the range of the starting and ending frames set and will not render them even if they are specified in the **Render Setup** dialogue box.

Playing Back a Wireframe Animation in Real Time

1. Select the window you want use to see your real time play back, and click the “L” (ruler icon) in the window title bar. This opens the Layout dialogue box for that window.
2. Click the **Fast Playback** option and click **Ok** to exit the dialogue box.
3. Choose the **PlayControl** menu cell in the Motion module. The Playback Controls dialogue box appears.
4. Enter the frame rate you desire (for example, 24, 25, or 30 frames per second) and select the **Refresh Scene** option. Click **Ok** to exit the dialogue box.
5. Play back your scene using the playback box. Your animation still does not play back at the frame rate you have specified. However, all the frames have been saved in memory, and subsequent playbacks will be faster.

6. Choose the **PlayControl** menu cell again, and change the frame step to 0 (zero) and select the **Keep Timing** option.

Setting the frame step to zero allows SOFTIMAGE 3D to skip frames if it cannot keep the frame rate you have entered and the **Keep Timing** option makes sure that SOFTIMAGE 3D actually uses the frame rate you have specified.

7. Playback your scene again – it's in real time.

Note

When using this mode, the frame rate values that are displayed are incorrect.

Playing Back Quickly

Selecting the **Fast playback** option in a window's Layout dialogue box lets you quickly play back a wireframe animation for the selected window. The animation is saved to buffer and when played again, it plays back much faster (the frame rate is maintained). When **Fast playback** is active in one window, other windows are not updated during playback.

Tip

Choose the **PlayControl** menu cell in the Motion module and set the **Frame Step** to 1 for the first playback and then specify the frame rate.

The **r Supra** key resets the fast playback so that if it is interrupted, all frames are recalculated. As you drag the time line pointer, the other windows are updated as soon as you release the pointer.

Fast playback remains active in Shade or Depthcue windows. In a Shade window, the animated objects are displayed in wireframe until you stop the playback. In a Depthcue window, the animated objects are displayed in wireframe and without depthcue until you stop the playback.

Tip

To display your animation in real-time, see the previous section.

Offsetting and Stretching Your Animation

Ever finish an animation and find that it is a little bit too short?

The **Sequence** commands in the Motion module allow global control over animation by offsetting or stretching (scaling) motion of objects. These controls can be applied to selected objects or all objects.

To double the length of your scene, and add 200 frames before your current animation of 100 frames, follow these steps:

1. Choose the **Sequence** ➔ **All** command in the Motion module.
2. Type 200 in the **Offset** text box. This adds 200 frames before the beginning of your animation.
3. Type 2 in the **Scale** text box. This value is the number by which the current length of your scene is multiplied.

Your scene is now 400 frames in length and the stretched animation begins at frame 200. The playback box does not change the number of frames that are displayed, so to view your entire animation, either change the number in the playback box or open the Fcurve window to view the scaled and offset function curves.



Be careful about scaling your animation: some function curves may not scale properly with your scene. If you do encounter problems, use the DopeSheet to scale a segment of your scene. For more information, see *Scaling Segments* on page 58.

For more details about the **Sequence** menu cell, see page 1255 in the *Reference Guide*.

Previewing the Animation

The **Update** commands allow you to view the animation for all objects or for a selected object only. The **Update → Selection** and **Update → All** commands are displayed as toggles, allowing you to immediately see which update mode is currently active.

These commands are very useful for previewing animation in real-time, or near real-time when working with complex animation sequences.

To preview the animation for a particular object in your scene:

1. Select the object.
2. Choose the **Update → Selection** or **All** command.
3. Play back your animation using the time line. Only the animation of the selected object is played back.

Viewing a Flipbook

Once you have rendered an image sequence, you can use the **FlipBook** menu cell or the **flipbook** standalone to display the animation on the screen. The flipbook is similar to traditional flipbooks because it plays back a sequence of images. However, the SOFTIMAGE 3D flipbook plays back in real time, as close to 30 frames per second as possible (depends on processes such as complexity, memory, etc.).

For more information about rendering a sequence to play back with the flipbook in SOFTIMAGE 3D, see the *Rendering User's Guide*.

To view an animation sequence:

1. Render your animation sequence using the **Render** menu cell in the Matter module. See *Rendering a Scene* on page 21 of the *Rendering User's Guide* for more information.
2. Choose the **FlipBook** menu cell in the Tools module. The Flipbook dialogue box is displayed.
3. Select the desired sequence you want to view, and enter the start frame, last frame, and the frame step for the sequence.
4. Define the other parameters in the dialogue box, such as the display speed (such as 24 frames per second), or the horizontal or vertical zooming factors.
5. Click Ok. The flipbook window appears loading all the frames that you specified.
6. When all the frames are loaded, use the playback box or the time line at the bottom of the display to play the animation sequence forward or backward.

Tip

When the flipbook is in a paused state, you can use the hardware zoom on the flipbook by using the number keys (1 to 0) on the keyboard. For example, pressing 2 on the keyboard enlarges your flipbook image to twice the size of the default image size.

You can also use the **flipbook** standalone to view a flipbook for your animation sequence. See the *Standalones* HTML files on the *On-Line Documentation CD* for more information about using the **flipbook** standalone.

C H A P T E R F O U R

Editing an Animation

Introduction

Once you have keyframed your animation, you can tweak the function curves of animated parameters to refine the animation. Function curves represent the relationship between the animated parameter. You can edit an animation using the Fcurve window, expressions, or the DopeSheet window.

In the Fcurve window, you can change the position of key points of a function curve, change its slope, or change its interpolation type. You can also perform basic editing techniques such as cutting, copying, and pasting either the entire function curve or just segments of it. If you want an animated parameter to follow precise values at certain frames of your animation, you can numerically edit the function curves. Using the Fcurve window, you can create animations such as a walking cycle for a character.

You can use perform more global changes to your animation in the DopeSheet window. Using the DopeSheet, you can modify your animation sequence by editing keyframes with standard editing operations such as moving, copying, and cutting. From the DopeSheet window, you can scale your animation, and certain relations between animated objects, such as constraints, can be activated or deactivated.

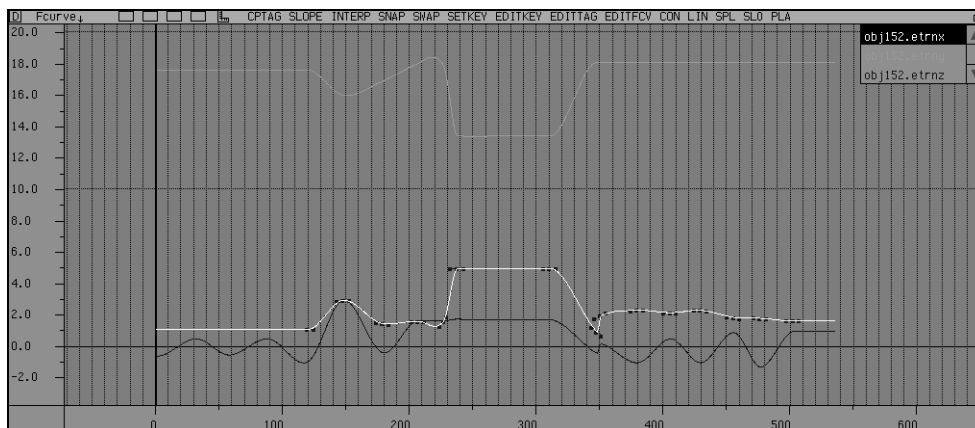
You can display your animation using Ghost view mode. This is useful when editing your animation, since the Ghost view mode allows you to display a series of snapshots of your animated objects at keyframes behind and ahead of the current frame.

The Motion mode in the Schematic window displays links between animation and objects in the scene. You can use this view on its own or cumulatively with other Schematic view modes (such as Model or Matter) to see animation on its own, or to see the relationships between them and other elements in your scene. For more information, see *Modes* on page 66 in the *Reference Guide*.

Another tool that is useful for editing your animation is the Spreadsheet window. The Spreadsheet allows you to query the contents of your scene, and display only the elements that you wish to edit. You can perform global or specific changes in the Spreadsheet. This feature of SOFTIMAGE 3D is described in the *Working with SOFTIMAGE 3D User's Guide* on page 84.

Editing Function Curves

Function curves allow you to animate the parameters of an element. A function curve is a graphic representation of the relationship between time and the value of an animated parameter. Selected function curves are displayed in the Fcurve (function curve) window, which is also sometimes known as the function curve editor. In it, you can modify the function curves to change the animation.



Keyframes are represented on a function curve by blue squares which are only visible when the function curve is selected. These keyframes are called *key points* on the function curve. The interpolation is shown as a line connecting the squares. Time (in frames) is displayed along the horizontal axis of the window and parameter values are displayed along the vertical axis.

Each displayed function curve name is the same colour as the associated function curve, and is listed in a box in the upper-right corner of the window.

Displaying Function Curves

The `FcrvSelect` commands in the Motion module display the function curves of an animated parameter in the Fcurve window.

For example:

1. Select an object for which you have already animated the rotation (chosen **SaveKey** → **Object** → **Rotation** → **All**). You can also use **Multi** menu cell to select many objects.

2. Choose the **FcrvSelect** → **Object** → **Rotation** → **All** command.

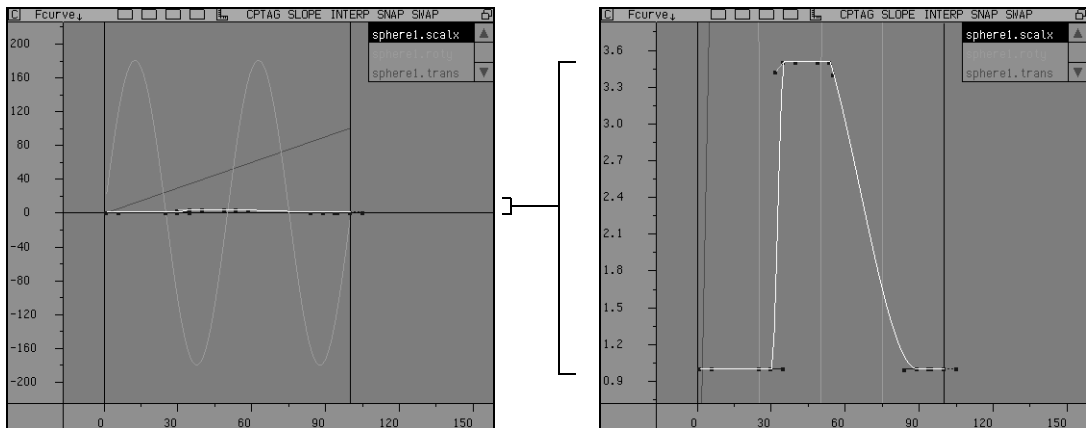
The Fcurve window opens displaying the object's rotation function curves for all three axes.

You can now edit the rotation function curves for your object. For more information about editing function curves in different ways, see page 35 through page 50.

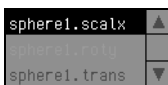
Zooming and Moving in the Fcurve Window

There are zoom operations designed especially for use in the Fcurve window:

- To zoom in on the horizontal axis of the Fcurve window, hold down the z and x Supra keys simultaneously. Use the middle mouse button to zoom in and the right mouse button to zoom out.
- To zoom in on the vertical axis of the Fcurve window, hold down the z and s Supra keys simultaneously. Use the middle mouse button to zoom in and the right mouse button to zoom out.
- To drag the camera on the horizontal axis of the Fcurve window, hold down the z and x Supra keys simultaneously. Use the left mouse button to drag the camera left or right.
- To drag the camera on the vertical axis of the Fcurve window, hold down the z and s Supra keys simultaneously. Use the left mouse button to drag the camera up or down.



Zoom on vertical axis.



Selecting Function Curves

To select a function curve displayed in the window, hold down the space bar and click directly on the function curve, or click on its name in the list of function curves in the upper-right corner. Once you have selected a function curve, it is highlighted in white, and its key points are visible. For a complete list of the abbreviations of the function curve names you see in the list, see *List of Attributes by Function Curve Name* on page 50 of Appendix D.

Grouping Function Curves

You can use the **Select → Set Named Selection** command to give a name to a group of selected function curves. This is useful when you are working with many function curves, and you want to separate them, for example, to group all the rotation function curves for a skeleton hierarchy.

1. Choose the **Multi** menu cell and select all the function curves you're interested in, then choose **Select → Set Named Selection**. The Set Named Selection dialogue box appears.
2. Enter the name of your selection, select the **Selected Fcurves** option and then click **Create**.
3. The name you have assigned to the group of function curves appears in the list of function curves in the right corner of the Fcurve window. Each time you click on the name, a group of function curves is selected.



You can toggle Named selection on and off with the **FcrvSelect → Options → Show Fcv Groups** command.

Deleting Function Curves

The **FcrvReset** commands in the Motion module delete the function curve for the selected parameter. The parameter value (which now has no animation) remains at its current value.



Once you have deleted a function curve, you cannot undo it. Be careful!

For example, to delete all the function curves for a light's colour, choose the **FcrvSelect → Light → Colour** command.

Clearing Function Curves from the Window

The **FcrvSelect → Clear Mode** command in the Motion module allows you to clear the function curves for the animated parameters currently displayed in the Fcurve window without deleting them. This is useful for when you have completed editing one group of function curves and want to begin working with others.

The **FcrvSelect → Clear Mode** command is a toggle that allows you to set the clear mode on or off. When selected (the default), the **Clear Mode** command automatically clears the Fcurve window when you select a new function curve.

If the **FcrvSelect → Clear Mode** command is not selected, you can select many different objects at the same time. For example, if you are animating a scene in which the camera moves in relation to an object in your scene, you can use this command to see and edit the function curves for both.

Choosing the Interpolation Type

When you change the interpolation type, you change the shape of your function curve. The interpolation is the way the animation is defined between key points on a function curve. The affects the acceleration and deceleration of the animation, as well as other factors.

For example, you can choose linear interpolation which creates sudden changes in movement between key points, or you can use spline interpolation to have ease-in and ease-out between key points, which allows you to create smooth animations.

Tip You can set the default interpolation type used in the **Preferences → Animation** dialogue box. See see page 1154 of the *Reference Guide* for more information.

You can select the following interpolation types for any function curve.

Constant/CON controls the interpolation between key points on a function curve by repeating the value of a keyframe for each of the frames after it. The movement is characterized by sudden changes and then static positions.





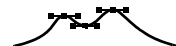
Linear/LIN creates regular intervals between key points making each segment of the function curve appear as a straight line. The movement is characterized by sudden changes in movement at each key point.



Spline/SPL is the default. It uses a spline interpolation between key points showing an acceleration (ease-in) and a deceleration (ease-out) according to the slopes at the key points. This results in a smooth transition of motion at the key points.



Slope 0/SLO creates a type of slope that allows you to constrain some or all key points of a function curve to a slope of 0. The handles (tangents) of the key points all remain horizontal (that is, with a slope of 0).



Plateau/PLA automatically sets the slope of a key point to zero if the next key has the same value, or if that key is a local minima or maxima. A local minima/maxima is a key point where both previous and next keys have either smaller (maxima) or larger (minima) values. This is useful to hold key point values to prevent the automatic interpolation from going above or below your keyframed values.

To change the interpolation:

1. Select an animated object.
2. Display the appropriate function curve using an **FcrvSelect** menu cell command. The Fcurve window opens displaying that function curve or curves.
3. Select a function curve in the Fcurve window.
4. Choose the **FcrvEdit** → **Interp Mode** command and one of its subcommands (Constant, Linear, Spline, Slope, or Plateau). You can also use one of the title bar options in the Fcurve window (CON, LIN, SPL, SLO, or PLA).

Choosing the Interpolation Type for Fcurve Segments

You can also set the interpolation mode for each segment of the function curve.

1. Select an animated object.
2. Display the function curve using the appropriate **FcrvSelect** command.

3. Choose the **FcrvEdit** → **Interp Management** command, the INTERP option in the Fcurve window title bar, or press the i Supra key.

The mouse buttons are configured so that:

- Left-clicking the key point sets the segment (to the right of the key point) to spline interpolation.
- Middle-clicking the key point sets the segment to linear interpolation.
- Right-clicking the key point sets the segment to constant.

Moving and Scaling Function Curves

You can do global editing of the function curve, including translation (offset) and scaling from within the Fcurve window.

You can also change translation and scaling (in x and y) with the **Trans** and **Scale** menu cells as you would for any object. In this case, changes in scale (in x and y) are from zero.

1. Select a function curve in the Fcurve window.
2. Choose the **FcrvEdit** → **Edit Fcurve** command or select EDITFCV in the Fcurve window title bar.

The mouse buttons are configured so that:

- The left mouse button translates the function curve. Click any key point, hold down the mouse button, and drag the function curve to offset it horizontally (frames) or vertically (movement amplitude).

 You can snap your function curve to grid units by choosing the Layout ruler icon for the Fcurve window and selecting the **Grid Lock** options for x and y.

- The middle mouse button scales the function curve horizontally (frames). If you click on the first key point, all the key points scale to the last key point. If you click on the last key point, all the key points scale to the first key point. If you click on any key point between the end key points, they scale toward end key points without exceeding them.
- The right mouse button controls the vertical scaling (movement amplitude). It works similarly to the middle mouse button for end key points, but with in-between key points, it scales all the key points according to the average value.

Compressing Time with Function Curves

In this example, you are changing the length of the animation by editing the function curves of an object.

1. Create an object and translate it in y.
2. Since you are changing the translation, save explicit translation keyframes for it (**SaveKey → Object → Explicit Translation → All**) at frames 1, 15, 30, 45, 60, 75, 90, and 105.
3. Choose **FcrvSelect → Object → Explicit Translation → All** to open the Fcurve window.
4. Select the y position function curve (**object.etrny**) and tag the key point at frame 1 by pressing the **t Supra** key.
5. Choose **FcrvEdit → Tag Key Points → Inverse** to switch the state of tagged and untagged key points.
6. Select the Layout option in the Fcurve window (the L ruler icon).
7. Set the **Grid size** and **Grid display** to 5 units in x and select the **Grid Lock** option.
8. Scale down tagged key points by choosing **FcrvEdit → Edit Tag Key Points**.
9. Then middle-click on the key point at frame 105 and drag it to the left until this key point is at frame 100. Each tagged key point is scaled toward the first tagged key point.
10. Repeat these steps until only one key point is tagged.
11. Untag the first tagged key point (frame 1).
12. Scale down all tagged key points so that the last one is moved down by 5 frames.
13. By the end of the process, the last key point should be at frame 70.
14. Change the end frame to 85.

Copying Function Curves

Using the **FcrvCopy** commands, you can copy the function curve(s) of animated objects, light, material, or texture parameters from a source object to the currently selected object. Copying function curves to various objects can save precious time and effort while animating.

1. Select the target object to receive the selected parameter function curve.
2. Choose the **FcrvCopy** → **Object** command and the appropriate subcommand for the function curve you want to copy.
3. Pick the object from which the parameter function curve is to be copied.

For more details about the parameters available to copy, see **FcrvCopy** or **FcrvSelect** in the *Reference Guide*.

Cutting, Copying, and Pasting Function Curve Segments

You can edit a displayed function curve by cutting segments, copying one segment to another location, or replacing one segment with another. You can refine the modified function curve by deleting points.

Note

You can scale the function curve segment in x or y, as well as specify the y pivot.

1. Display the desired function curve(s) using the **FcrvSelect** menu cell and one of its commands.
2. The **Fcurve** window opens to display the function curves. If more than one function curve is displayed, select the one you want to edit.
3. Choose the **FcrvEdit** → **Copy/Paste Fcurve Segment** command.

The **Copy/Paste Fcurve Segment** dialogue box is displayed. An empty buffer window in the upper-right corner of the dialogue box is where the selected function curve is displayed.

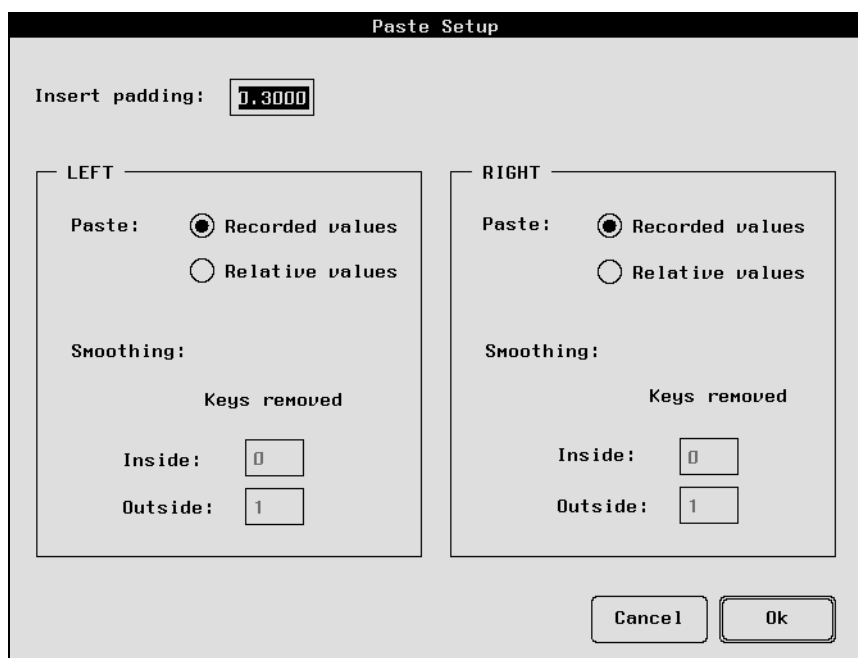
4. The operation which is selected by default is the **Copy** option. You can also use the **Cut** option to remove the function curve segment from the function curve and place it in the buffer.

5. Click the **Process** button at the bottom of the dialogue box.

The selected segment is copied into the function curve buffer and displayed in the buffer window. The Buffer Info area displays information about the copied function curve segment, and the **Start** and **End** text boxes display the first and last frames of that function curve.

6. Click the **Setup** button.

The Paste Setup dialogue box is displayed, where the editing parameters for your operations can be specified.



The image shows a 'Paste Setup' dialog box with a title bar. At the top, there is a label 'Insert padding:' followed by a text box containing '0.3000'. Below this, the dialog is divided into two main sections: 'LEFT' and 'RIGHT'. Each section contains a 'Paste:' group with two radio buttons: 'Recorded values' (selected) and 'Relative values'. Below the 'Paste:' group is a 'Smoothing:' group with a 'Keys removed' label and two text boxes: 'Inside:' with '0' and 'Outside:' with '1'. At the bottom right of the dialog are 'Cancel' and 'Ok' buttons.

For more information about each option in this dialogue box, see *Setup* on page 695 of the *Reference Guide*.

7. Set the parameters as desired, then click **Ok** to confirm the settings and return to the Copy/Paste Fcurve Segment dialogue box.
8. Select the **Insert** or **Replace** option, depending on what you want to do with the function curve segment. For details on using the operations, see *Operations* on page 692 of the *Reference Guide*.

9. Select the function curve segment on which you want to perform the operation by typing the first and last frame numbers of that segment in the **Start** and **End** text boxes. Select another function curve if you want to paste on something other than the current function curve.
10. Click **Process** to update the buffer window and perform the operation.

The displayed function curve updates to reflect the modifications.

Saving and Retrieving Function Curve Segments

You can use the Fcurve buffer commands to save and retrieve function curve information. This is useful for pasting a function curve segment to another element in either the same scene or a different one instead of trying to recreate that part of the animation. You can also paste a function curve segment to a function curve on an external software or device. Pasting a function curve to another element also allows you to save time, since you can use function curve segments that you use often.

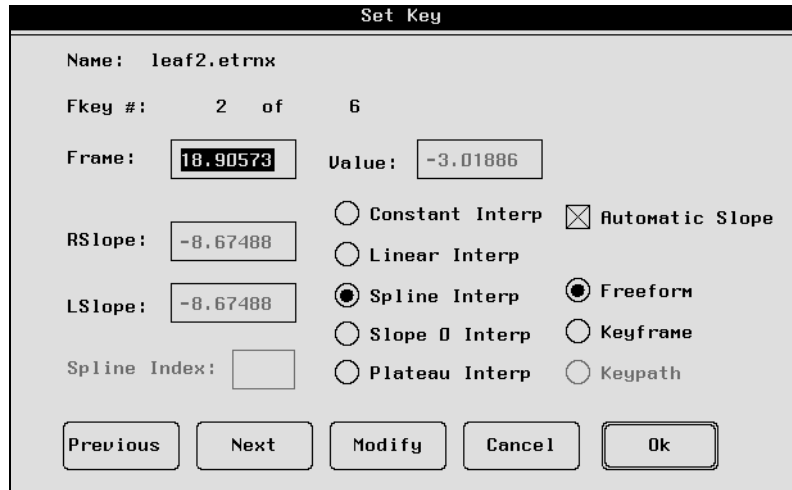
1. To save a function curve segment, you must first copy the segment using the **FcrvEdit → Copy/Paste** command.
2. This places the function curve segment in the buffer. To save the function curve that you placed in the buffer, choose the **Save → Fcurve Segment** command in the Motion module.
3. You can retrieve this function curve segment by using the **Get → Fcurve Segment** command in the Motion module.
4. To view the contents of the buffer, choose the **Info → Fcurve Buffer** command to see information about the function curve.

Editing Function Curves Precisely

You can numerically edit a function curve using parameters in a dialogue box. This gives you real precision by allowing you to edit the function curve to specific values.

1. Select a function curve in the Fcurve window.
2. Choose the **FcrvEdit → Set Key** command, or select **SETKEY** from the Fcurve window title bar, then click the key point on the function curve that you want to edit.

The Set Key dialogue box is displayed.



The Set Key dialogue box is a window with a title bar labeled "Set Key". It contains the following fields and controls:

- Name:** leaf2.etrnx
- Fkey #:** 2 of 6
- Frame:** 18.90573
- Value:** -3.01886
- RSlope:** -8.67488
- LSlope:** -8.67488
- Spline Index:** (empty field)
- Interpolation options:**
 - ☐ Constant Interp
 - ☐ Linear Interp
 - ☒ Spline Interp
 - ☐ Slope 0 Interp
 - ☐ Plateau Interp
 - ☒ Automatic Slope
 - ☒ Freeform
 - ☐ Keyframe
 - ☐ Keypath
- Buttons:** Previous, Next, Modify, Cancel, Ok

3. Modify the **Frame** number and parameter **Value** to be precisely what you want.
4. If you like, modify other parameters such as the interpolation type, or what type of key point it is (**Freeform**, **Keyframe**, or **Keypath**).
5. Click **Modify** to accept these values.
6. If another key point needs modification, click **Previous** or **Next** to display the previous or next key point.
7. Click **Ok** to accept the settings or **Cancel** to exit the dialogue box.

Viewing Function Curve Information

You can use the **Info → Fcurve** command to obtain quick information about a function curve, such as to verify whether or not your key points are directly on frames. If you need to edit any of the numerical values in the **Info → Fcurve** dialogue box, double-click on a key point to open the Set Key dialogue box.



You can also display either a segment of a function curve between two tagged key points, or just one key point on the function curve. To do this, use the **FcrvEdit → Tag Key Points** command to tag the desired key points. The segment between the first and last tagged point is displayed in the buffer window, but information on the key point appears in

the Buffer Info area in the Copy/Paste Fcurve Segment dialogue box.

To view information about a function curve:

1. In the Fcurve window, select a function curve to display information on a specific function curve.
2. Choose the **Info → Fcurve** command. The Info Fcurve dialogue box appears, displaying the name of the function curve, type of interpolation, and keyframes. For each keyframe, parameters such as frame, value, and zero slopes are displayed.

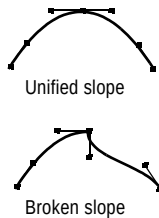
You can also use the **FcrvEdit → Set Key** command to edit key point values as described in the previous section. See page 682 in the *Reference Guide* for more information about editing the contents of the Info Fcurve dialogue box.

Displaying and Hiding Slopes

The **FcrvSelect → Options → Draw Slopes** command is a toggle that makes slopes on the function curve visible or not in the Fcurve window. It is active by default. This command is useful if many keys are created, such as on a highly sampled function curve from motion capture, and you want to view the key points and not the slopes.

Unifying and Breaking Slopes

You can set the relationship between the left and right slope of key points when the function curve interpolation mode is set to Spline (the default).



1. Select a function curve using the appropriate **FcrvSelect** command.
2. Choose the **FcrvEdit → Slope Management** command, or click **SLOPE** in the Fcurve window title bar, or press the **s** Supra key.

The mouse buttons are configured so that:

- To unify the key point, left-click it. This gives the smoothest possible transition at key points for any given position of the slopes.
- To break the key point, middle-click it. This allows you to control the tension of the right and left slopes separately. You can then drag the handles of the key point to modify the shape of each slope.
- To set a key point to release, right-click it. Slopes are unified and their positions are recalculated to give the smoothest curve. Dotted

lines identify when a slope is released. This is the default mode for the key points.

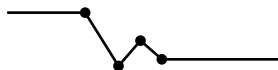
Controlling the Slope's Shape

Controlling the Shape before and after Key Points

The **FcrvEdit** → **Extrap Mode** → **Constant** command allows you to control the shape of the function curve before and after the first and the last key points.

Constant, the default, extrapolates the value of the y-axis from the first frame to the first keyframe and the value of the y-axis from the last keyframe to the last frame. This results in a hold of motion before and after the movement.

1. Select a function curve in the Fcurve window.
2. Choose the **FcrvEdit** → **Extrap Mode** → **Constant** command.



Another way of controlling the slope before and after a key point:

1. Select the key point.
2. Click **SLOPE** in the Fcurve window title bar.
3. Middle-click the key point to break the slope.
4. Move the handles independently to change the shape of the curve.

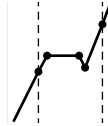
Extrapolating Increments between Key Points

The **FcrvEdit** → **Extrap Mode** → **Gradient** command extrapolates the increment calculated between the first and second key points (for a linear interpolation only) and applies it from the first frame to the first keyframe. It then extrapolates the increment calculated between the last and previous key points and applies it from the last keyframe to the last frame.

For spline interpolation, the slope of the first or last key is used. This results in a continuation of the motion at the same speed, except for

translation on open paths as the object stops when it reaches the ends of the path.

1. Select a function curve in the Fcurve window.
2. Choose the **FcrvEdit** → **Extrap Mode** → **Gradient** command.

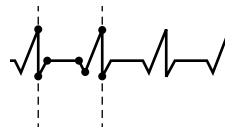


Repeating the Function Curve Shape

The **FcrvEdit** → **Extrap Mode** → **Cycle** command repeats the function curve shape/pattern defined by the key points. This results in the cycling of that pattern from the first keyframe to the last frame.

If you have one or more keys points on your function curve that are not on a frame, you will get strange effects because the cycle is evaluated at each frame and not at an intermediate time. Choose **FcrvEdit** → **Set Keys on Frames** to fix this – it puts non-integer key points on integer frames.

1. Select a function curve in the Fcurve window.
2. Choose the **FcrvEdit** → **Extrap Mode** → **Cycle** command.



All cycles are converted to keyframes. The function curve is repeated until the end frame you have specified in the playback box is reached.

Freezing a Function Curve Cycle

The **FcrvEdit** → **Extrap Mode** → **Freeze Cycle** command freezes the selected function curve by duplicating the keys onto the cycled part of the function curve. The function curve can then be edited.

If you have one or more keys points on your function curve that are not on a frame, you will get strange effects because the cycle is evaluated at each frame and not at an intermediate time. Choose **FcrvEdit** → **Set Keys on Frames** to fix this – it puts non-integer frames on integer frames.

1. Select a function curve in the Fcurve window.
2. Choose the **FcrvEdit** → **Extrap Mode** → **Freeze Cycle** command.

You must enter the start and end frame which you want to use to freeze the cycle. The cycle is frozen as new key points on the function curve.

Editing Key Points

To edit keys points on a selected function curve:

1. Select a function curve in the Fcurve window.
2. Choose the **FcrvEdit** → **Edit Key Point** command or select **EDITKEY** in the Fcurve window title bar.
3. In the Fcurve window, edit the key points on the selected function curve using the mouse buttons that are configured as follows:
 - The left mouse button **moves** a key.
 - The middle mouse button **inserts** a key.
 - The right mouse button **deletes** a key.



Removing lattice, shape, and cluster key points in the Fcurve window does not actually remove the respective lattice, shape, or cluster from memory, only its location in time. To remove lattice, shape, or cluster keys, refer to the **Lattice**, **Shape**, and **Cluster** commands.

Moving, Inserting, and Deleting Key Points

To move a key point, follow the previous steps, then pick a key point with the left mouse button and, while holding down the button, drag the mouse to move the point. The curve is modified accordingly and the x and y values are displayed dynamically in the status line. You can also use the left mouse button on a slope handle to modify the curves slope around the associated key point.

Freeform key points (shown in blue) and raw data key points (shown in green) can be moved along the x and y-axes.

Keypath key points saved with the **SaveKey** → **Object** → **Keypath Translation** command can only be moved along the x-axis (time) because this type of key defines a fixed relationship between the keyframe and a point on a path.

Note

For keypath translation, if you select a translation key, then select the SETKEY option in the Fcurve window, and select the **Keyframe** option in the dialogue box, the key changes to yellow. Every subsequent translation keyframe saved with the **SaveKey → Object → Keypath Translation** command is now a Keyframe key and cannot be moved in the Fcurve window.

To insert a key point, follow the steps in the previous section, and move the cursor to the desired location on the function curve in the Fcurve window. Click the middle mouse button, and a new key point appears. Hold the button and drag to fine-tune the shape of the curve. When satisfied, release the button.

Tip

To insert a key point directly on the function curve without modifying its shape, hold the Shift key and middle-click where you want a key point.

To delete a key point, follow the steps in the previous section and click on the key point with the right mouse button.

Setting Key Points on Frames

You can set key points positioned between frames (non-integer frame numbers) onto the frames. Keys points that are not set at a time value that corresponds to a frame are moved to the closest frame. This is determined by the **Overlapping Key Distance** option in the Animation Preferences dialogue box (see **Preferences → Animation** on page 1154 of the *Reference Guide*).

This command is especially useful when working with rotations with discontinuous angles (-180 and 180 degrees).

Tip

To keep key points at integer frame numbers, you can also select the **Grid Lock** option for X in the Fcurve window Layout dialogue box (click the L-shaped ruler icon). This prevents you from moving key points to a non-integer frame.

1. Select the function curve(s) you want to edit in the Fcurve window.
2. Choose the **FcrvEdit → Set Keys on Frames** command.

All key points that are not on frames are snapped to frames. For example, if you have a key point that is located at frame 1.56, then it is moved to frame 2.

Locking and Unlocking Key Points

You can lock key points on a function curve to protect them from being moved inadvertently.

1. Select a function curve in the Fcurve window.
2. Choose the **FcrvEdit → Lock Key Points** command that you want.
 - **Point:** Click on each single key point to lock it.
 - **Rect:** Draw a rectangle by defining opposite corners and including key points to lock.
 - **All:** Locks all key points.
 - **Clear:** Unlocks all key points.
 - **Inverse:** Toggles locking status of key points.

Tagging and Untagging Key Points

You can tag (select) key points to edit a function curve locally. The tagged key points are highlighted in red. You can also use the **t Supra** key to tag key points individually.

1. Select a function curve in the Fcurve window.
2. Choose the **FcrvEdit → Tag Key Points** command and one of these subcommands:
 - **Point/Rect** tags a key point or perform a rectangular tag.
 - **All** tags all key points on the function curve.
 - **Clear** untags all tagged points.

Inverting the Key Point State

You can change the current state of the key points (tagged or untagged).

1. Select a function curve in the Fcurve window.
2. Choose the **FcrvEdit → Tag Key Points → Inverse** command to invert the current tag state of all key points between tagged and untagged.

Deleting Tagged Key Points

1. Select a function curve in the Fcurve window.
2. Choose the **FcrvEdit → Tag Key Points → Delete** command to delete the tagged key points.

Copying, Cutting, and Pasting Tagged Key Points

You can copy, cut, or paste tagged key points.

1. Tag the desired key points on a function curve in the Fcurve window.
2. Choose the **FcrvEdit → Copy/Paste Tag Keys** command or select CPTAG in the Fcurve window title bar.
 - The left mouse button copies the tagged key points to a buffer.
 - The middle mouse button cuts the tagged key points.
 - The right mouse button pastes the tagged key points that are copied to the buffer keys at the selected insertion point.

Moving and Scaling Tagged Key Points

You can do global editing on key points of the function curve.

1. Select a function curve in the Fcurve window.
2. Choose the **FcrvEdit → Edit Tag Key Points** command or select EDITTAG in the Fcurve window title bar.

The mouse buttons are configured so that:

- The left mouse button translates tagged key points.
- The middle mouse button scales tagged key points horizontally.
- The right mouse button scales tagged key points vertically.

Using Auxiliary Function Curves

Each function curve has a corresponding auxiliary curve that is shown in dotted lines in the Fcurve window. The SNAP option of the Fcurve window title bar lets you keep a “copy” of the original function curve as you are modifying it. With the SNAP option, you can take a “snapshot” of the current function curve’s position. When you edit the function curve, this snapshot remains as the auxiliary function curve. Using these snapshots as a visual reference, you can edit your function curve progressively.

If you want to return to the auxiliary function curve at any time, select the SWAP option, which lets you return to the previous SNAP version of your function curve (the auxiliary function curve), which is shown in dotted lines. This is a very useful feature for experimenting with different modifications on a function curve. It is essentially a way of “undoing” an edit to return to the previous version.

1. Select a function curve in the Fcurve window.
2. Choose the **FcrvEdit → Snapshot Fcurve** command or SNAP in the Fcurve window title bar. This saves an auxiliary function curve.
3. Edit the function curve as desired. The auxiliary function curve, which has preserved the shape of the original function curve, appears as a dotted line.
4. To return to the auxiliary function curve, choose the **FcrvEdit → Swap Fcurve** command or SWAP in the Fcurve window title bar.

Converting Function Curves

A function curve in your scene may have key points that are placed at various frames throughout your animation. SOFTIMAGE 3D evaluates the function curve and interpolates between the key points. Unlike this, a raw function curve has one key point per frame. The following are some reasons why you may want to convert function curves:

- When you use motion capture to animate an object, the data that you load into SOFTIMAGE 3D usually has raw function curves. You may want to remove some of the key points to edit the animation.
- Conversely, if you keyframe an animation in SOFTIMAGE 3D, and you want to transfer data to an external device such as a camera, you may want to convert your function curve to a raw function curve to preserve exact motion.

Converting a Function Curve to Raw

You can convert a selected function curve to a raw function curve which has one key point per frame.

1. Select a function curve in the Fcurve window.
2. Choose the **FcrvEdit → Convert to Raw Fcurve** command.

Converting Raw to Spline Function Curve

You can convert the selected raw function curve to a spline interpolation type function curve.

1. Select a raw function curve in the Fcurve window.
2. Choose the **FcrvEdit → Fit Spline to Fcurve** command. The Fit Spline to Fcurve dialogue box is displayed.
3. Set the desired **Tolerance** value. A smaller value yields a more precise conversion.

Associating and Disassociating Function Curves

The **FcrvCopy → Associate Anim.** or **Disassociate Anim.** commands allows you to associate or disassociate a predefined set of animation function curves from an object or hierarchy, material, camera, etc., to the selected object.

An object that is animated is associated to an animation element (visible in the Schematic window when the Motion mode is active) that contains a list of function curves and list of objects using that set of function curves. It can be read directly from disk using the **Get → Element** command. See page 761 in the *Reference Guide* for more information on this command.

To associate or disassociate a function curve:

1. Select the object from which you want to associate or disassociate the animation function curves. You must be in the Schematic window and its Motion mode must be active.
2. Choose the **FcrvCopy → Associate Anim.** **Disassociate Anim.** command.
3. Pick the animation element.
 - Use the left mouse button to associate all nodes in a hierarchy to the animation element.
 - Use the middle mouse button to associate selected nodes of the selected hierarchy root to the animation element.

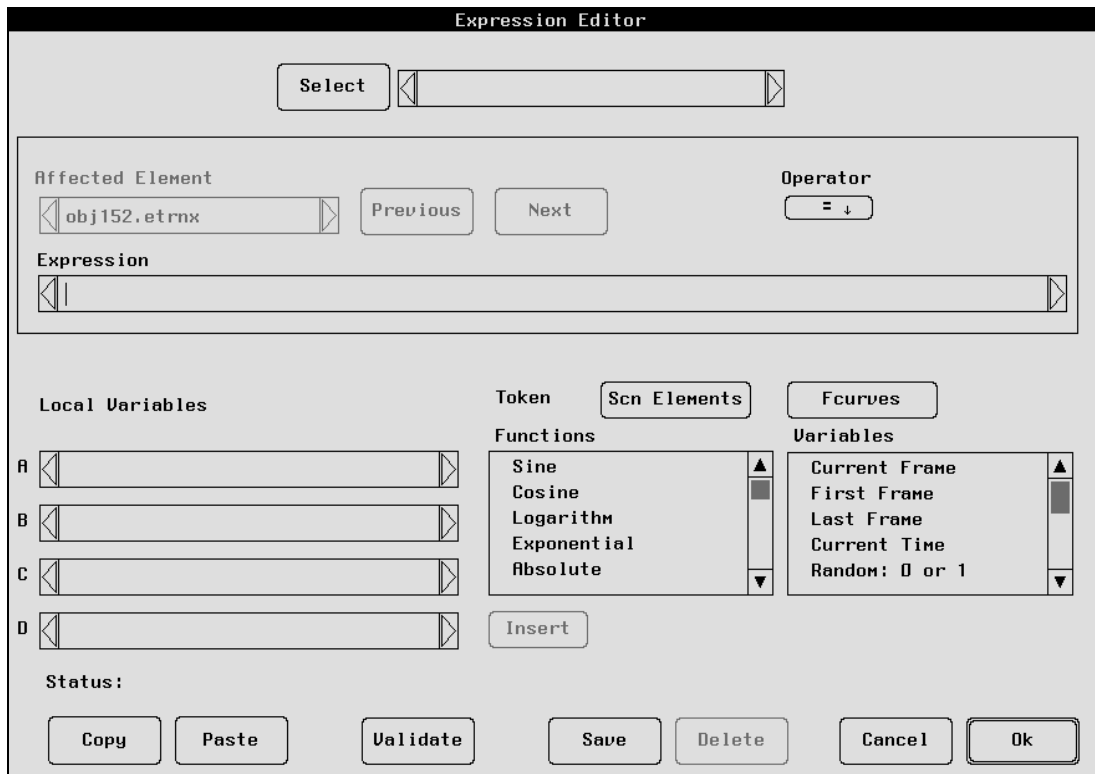
Editing Function Curves Using Expressions

In addition to the Fcurve window, you can also modify an existing function curve with mathematical expressions. For example, you could add jitters to an existing scaling curve by adding an expression to it, or create an undulating wave using a simple sine function.

Using expressions in conjunction with the Fcurve window brings up the same Expression Editor as when using the **Expressions → Edit** command in the Motion module; however, with function curves the expression is not retained as a permanent link between elements: it is merely used as a tool to draw and edit curves. You can, of course, save any expression to a file to use it again.

1. Select an animated element. This can be an object, a light, a camera, etc. – anything with a function curve. This is called the “affected element.”
2. Choose **FcrvSelect** and the appropriate command relating to the type of function curve you want to modify. The Fcurve window appears.
3. Choose the **FcrvEdit → Expression** command.

The Expression Editor dialogue box appears.



For a description of all parameters in this dialogue box, see *Parameters* on page 647 of the *Reference Guide*.

There are many possibilities for using the parameters, but here's a basic procedure for editing a function curve. See *Expressions* on page 197 for more information about using expressions.

4. The element and its function curve you selected in the scene appears in the **Affected Element** text box. This cannot be changed.
5. Now create your expression. If there is a predefined expression you want to use, click the **Select** button and choose that expression from the browser that appears.

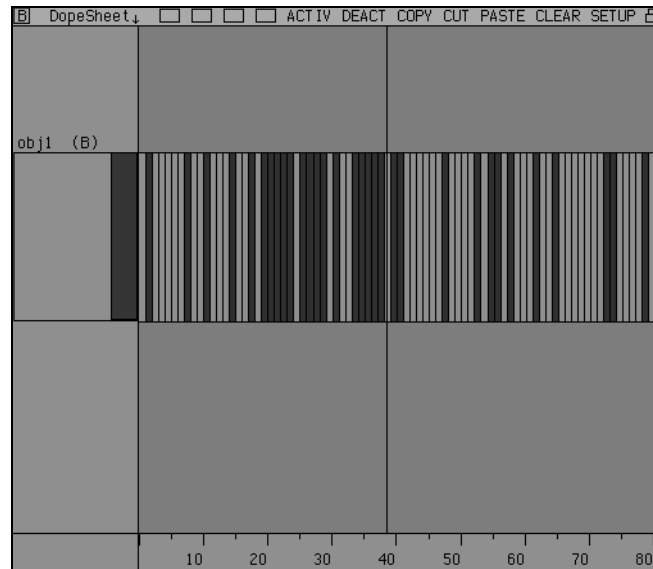
To create an expression from scratch, follow these steps:

- If you want to create an expression that doesn't relate to any other element in the scene, skip steps 6 and 7.

- If you want to reference the values or animation of other elements, select the element and then do steps 6 and 7.
- 6. Click in the **Expression** text box and then click the **Scn Element** button. The Select Element browser appears in which you can choose any element from the scene.
- 7. Add the scene element's function curve that is used to edit the affected element's function curve. Click at the end of the element's name (after the period) and then click the **Fcurve** button. Select one of the possible function curves from the Function Curves dialogue box that appears.
- 8. At this point, you could add an operator and some numeric values to this expression, such as + 3. You could also use the predefined math **Functions** such as cosine, logarithm, or conditional, or **Variables** such as current frame, first frame, or a random value.
- 9. If a segment of the expression needs to be repeated, you can define it in the **Local Variables** text boxes. This lets you keep the main expression simpler. To refer to a local variable, use the corresponding letter (A, B, C, or D) in the main expression.
- 10. Click **Validate** to make sure that you have a valid expression.
- 11. If you want to save the expression to a file, click **Save** and specify a name for it in the Save Expression dialogue box that appears.
- 12. Click **Ok** to apply this expression to your affected element's function curve.

Editing with the DopeSheet Window

The DopeSheet window provides you with another way of viewing and editing the animation data for selected objects in an animated scene. You can use the DopeSheet to modify your sequences by editing keyframes with standard editing operations such as moving, copying, and cutting. In addition, certain relations between animated objects, such as constraints, can be activated or deactivated from the DopeSheet. Essentially, the DopeSheet is an alternative to the Fcurve window as a way of manipulating your animation data.



The DopeSheet displays the animation data in horizontal tracks. A track consists of a series of rectangular boxes, each representing one frame. An object such as a camera, model, or light, or a named selection must be selected before its corresponding tracks can be displayed in the DopeSheet window. If an object is selected, a track for that object is displayed.

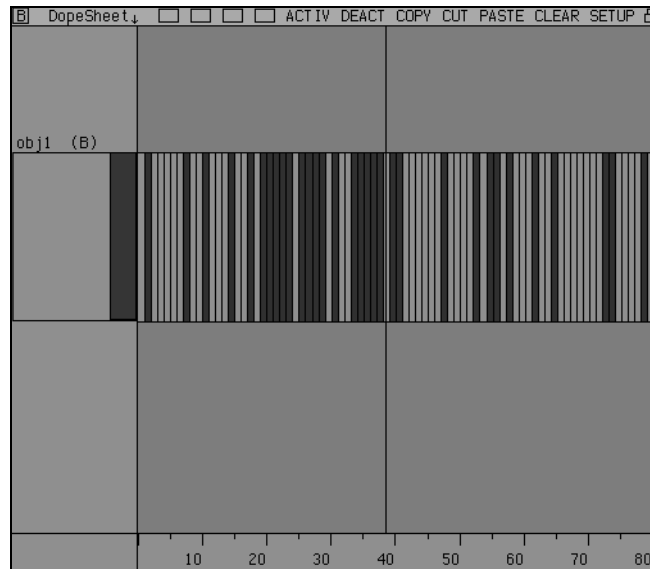
Tracks

The DopeSheet window displays a series of animation and relation tracks. A track consists of a series of narrow rectangular boxes, each representing one frame. User-defined keyframes are coloured blue and linked keyframes are coloured light green. The green box to the right of the track name indicates that a track is active. If the box is

red, it means that the track is inactive. A dotted green box means that some tracks beneath are active, while others are not.

Track Levels

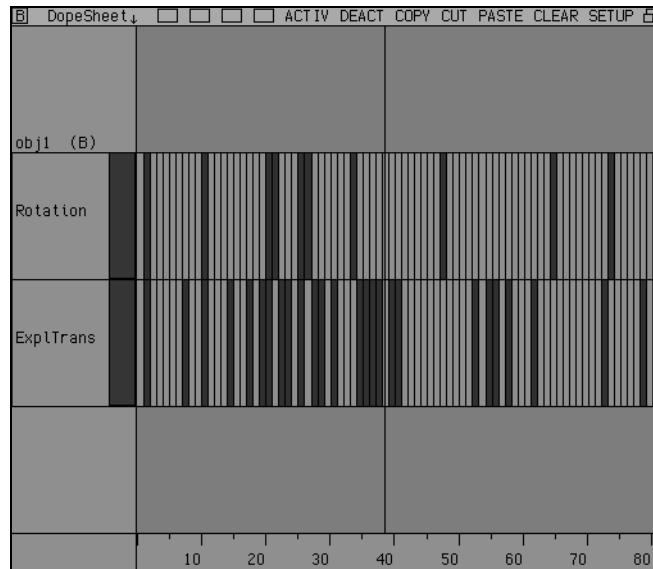
When you open the DopeSheet window, the initial track displayed for each selected object or named selection is an *object track*. An object track represents all animation and relation tracks for that object. The name of the object, such as “obj1,” is positioned just above the track on the left side of the window. A letter “B” beside an object name means the object is branch-selected. Clicking the name opens an Info dialogue box, which allows you to edit the object.



The initially displayed object track can be “expanded” to access various levels of related groups of tracks beneath it.

To access the next level of a track, click in the large rectangle to the left of a track, beneath the object name. You can “collapse” a track

and return to the level above by middle-clicking the same area.



Navigating in the DopeSheet Window

The standard methods for moving around in any window, such as **Zoom**, **Rect Zoom**, **Frame All**, and **Reset** (all found in the **Camera** menu), and the **Preferences → Zoom on Cursor** command, can also be applied to the DopeSheet window. In addition, there are other zoom operations especially for the DopeSheet window:

- To zoom in on the horizontal axis of the DopeSheet, hold down the **z** and **x** Supra keys simultaneously. Use the middle mouse button to zoom in and the right mouse button to zoom out.
- To zoom in on the vertical axis of the DopeSheet, hold down the **z** and **s** Supra keys simultaneously. Use the middle mouse button to zoom in and the right mouse button to zoom out.
- To drag the camera on the horizontal axis of the DopeSheet, hold down the **z** and **x** Supra keys simultaneously. Use the left mouse button to drag the camera left or right.
- To drag the camera on the vertical axis of the DopeSheet, hold down the **z** and **s** Supra keys simultaneously. Use the left mouse button to drag the camera up or down.

Activating and Deactivating a Track

An active track is identified by a green box located to the left of the animation track, while a red box represents a deactivated track. If you do not want a particular animation or relation to be updated during playback of your scene, you can either deactivate the entire track or just a selected segment of that track. To activate an entire track:

- Click the red box to the left of the track.

The box turns green to indicate that the track is activated.

Note A dotted green box indicates that some of the tracks beneath the currently displayed level are deactivated.

To deactivate an entire track:

- Click the green box to the left of the track.

The box turns red indicating that the track is deactivated.

Deactivating a track at one level affects the tracks at all levels beneath it. For example, if an animation attribute track for rotation is deactivated, all rotation parameters (rotx, roty, rotz), are also deactivated.

Note It is not currently possible to deactivate cluster relations in the DopeSheet; you must use the **Shape → Cluster List** command's dialogue box.

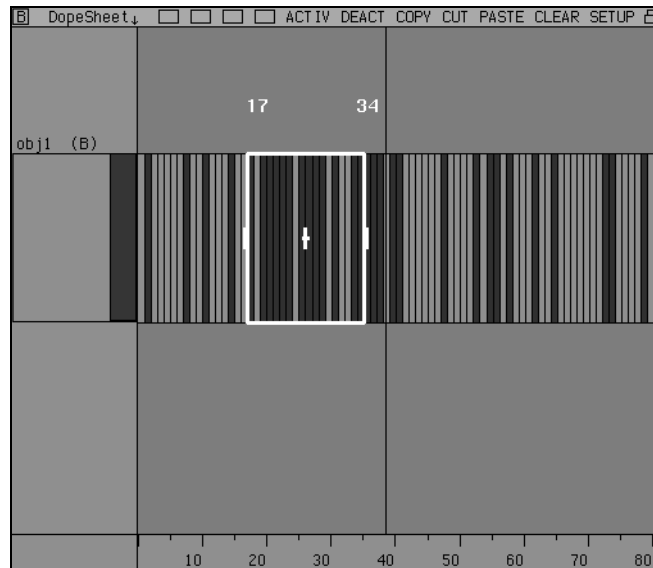
Selecting Segments

In the DopeSheet window, you can edit your animation and relations by activating and deactivating entire tracks or segments of tracks, and by moving, scaling, copying, and cutting segments of tracks.

Before any of these editing operations can be performed, you must first select segments within an editing box:

1. Move the mouse cursor over an animation track and hold down the left mouse button as you drag the mouse to the left or right.

2. Release the mouse button when the desired segment is selected.



The selected segment is outlined in a white editing box with the first and last frame numbers of the segment shown above it. A message displayed in the status bar reads “Create editing box.”

Note

Only one segment at a time can be selected; however, you can edit multiple objects at once using Named Selections.

3. Right-click anywhere on a track to deselect a segment.

The editing box is deleted and the status bar message reads “Remove editing box.”

To modify the size of an editing box, use the left mouse button to drag one of the handles on either side of the editing box to the left or right (if you use the middle mouse button, you will modify the contents of the track). The status message reads “Resize editing box.”

To move an editing box to another area of the same track or to another displayed track, click the cross in the middle of the editing box and drag the box to the desired area. The status message reads “Move editing box.”

Playing Back a Selected Segment

You can play all or part of your animation at any time while working in the DopeSheet window by clicking the playback box arrows.

To play part of the animation only, select the desired segment, and then click the playback box arrows. Only the frames within the editing box and for the object represented by the track are played back. As the animation plays, a black vertical indicator moves through the DopeSheet. You can also do this using the F, K, and L options in the playback box as you would normally while playing back an animation.

Activating and Deactivating Segments

To activate a segment of a track, select the segment, and then click **ACTIV** in the window title bar.

You can deactivate a selected segment of a track so that it will not be updated during playback. This is especially useful if you have multiple constraints on one object and would like to deactivate some of those constraints for part of your animation.

To deactivate a selected segment of a track, select it, and then click **DEACT** in the window title bar. The deactivated segment is dimmed.

Scaling Segments

When you scale segment using the middle mouse button, the location of the keys within the editing box adjust to accommodate the change in size of the selected segment. Scaling up the selected segment slows down the animation by increasing the number of frames between keyframes, while scaling down the selected segment speeds up its animation by decreasing the number of frames between keyframes.

Note

You can set the DopeSheet to automatically snap keyframes to frame boundaries upon editing your animation. For example, if you try to scale a segment of channel animation (one keyframe per frame), snapping keys on frame boundaries may result in overlapping keyframes. In this case, you would not use the snap option. See page 68 of the *Reference Guide* for more information.

There are two ways to scale a segment:

- If you do not want to affect the keys outside the editing box as you

scale the segment, hold down the Shift key and drag one of the handles on either side of the editing box with the middle mouse button. The editing box turns red, and the status bar message reads “Scale keys and drop.” This means that the keys outside the segment do not move at all.

- To scale the selected segment so that the keys outside the editing box are also moved, drag one of the handles on either side of the editing box with the middle mouse button. The editing box turns red as you scale the segment, and the status message reads “Scale keys and push.”

This method causes the keyframes outside that segment to move to accommodate the modification. If a segment is scaled up, the keys outside the editing box are pushed in the same direction in which the editing box is being enlarged. For example, if you drag the right side of the editing box to the right, the keys inside the box are scaled up and the outside keys to the right of the box are pushed farther to the right. As a result, the animation relating to the segment slows down. If you scale an editing box down, the opposite effect occurs: the outside keys are “pulled” together, and the animation relating to the segment speeds up.

Moving Segments

If you move a selected segment using the middle mouse button to the left of its original position, the keyframes within it are pushed back in time. If you move a selected segment to the right, the keyframes within it are pushed forward in time.

There are two ways to move a selected segment: the first method does not affect the keys outside the editing box, while the second method does.

- If you do not want to affect the keys outside the editing box as you move the segment, hold down the Shift key and drag the cross in the middle of the editing box with the middle mouse button. The editing box outline turns red as you move the segment and the status bar message reads “Move keys and drop.”
- To move the selected segment so that the outside keys are also moved, simply drag the cross in the middle of the editing box using the middle mouse button. The outside keys are “pushed” together in the same direction as the editing box is being moved. The editing box outline turns red and the status bar message reads “Move keys and push.”

Copying, Cutting, and Pasting Segments

Copying

The COPY option in the DopeSheet window title bar is used with the PASTE option. It copies a selected segment of a track to a paste buffer, allowing you to reinsert it in another area of a track. You can paste a segment into another selected segment that is a different size. The pasted segment is then scaled up or down to fit into the new segment.

To copy a selected segment to another area of a track:

1. Select the segment and click COPY in the window title bar. The segment is copied to a paste buffer.
2. Click the frame at the desired insertion point, or select the segment into which you would like the copied segment inserted, and click PASTE.

The segment is copied from the paste buffer and reinserted into a track. When only the point of insertion is indicated, the frames to the right of the inserted segment move further to the right (forward in time) to accommodate that segment. When a segment has been selected, the original segment is scaled up or down to fit in it and *replaces* the original contents of the editing box.

Cutting

The CUT option is used to delete segments from the animation completely. Used with the PASTE option, you can remove a segment from one area and paste it to another. You can paste a segment into another selected segment that is a different size from the original segment. The pasted segment is scaled up or down to fit into the new segment and *replaces* the original contents of the editing box.

Once a segment is cut from the track, the remaining frames to the right move left, or back in time, to take that segment's place.

To cut a segment and paste it in another area:

1. Select the segment and then click CUT in the window title bar. The segment is removed to a paste buffer.
2. Click the frame at the desired insertion point, or select the segment into which you would like to paste the cut segment, then click PASTE.

The segment is copied from the paste buffer and reinserted into a

track. When only the point of insertion is indicated, the frames to the right of the pasted segment move further to the right (forward in time) to accommodate that segment. When a segment has been selected, the pasted segment is scaled up or down to fit it, and *replaces* the original contents of the editing box.

Pasting

The PASTE option is used with the CUT and COPY options. It copies a segment from the paste buffer and inserts it into a selected area of a track, replacing the original contents of the editing box.

Because the segment remains in the paste buffer until it is replaced by another segment, you can paste the same segment as many times as desired by continuing to specify insertion points and clicking PASTE.

Although you can copy/cut and paste segments from a lower level track to a higher level track (such as from the animation parameters level to the animation track level (*scaly* to *Scale*)), the cut/copy and paste operations have certain limitations:

You can copy/cut and paste segments from one object's track to another's *only* if the track level is the same or higher, *and* the animation attribute or relation are the same. For example, you can copy/cut and paste from one object's animation track corresponding to another object (Rotation to Rotation), or from one object's attribute parameter track to another object's animation track (*roty* of the first object to Rotation of the second object).

An exception to this occurs at the lowest level tracks (attribution parameters and target relations level). At this level, you *can* copy from one attribute parameter type to another. For example, you can copy one object's *scaly* to another object's *rotx*.

Deleting Keyframes

The CLEAR option deletes selected keyframes and the white editing box from an animation track.

Unlike the CUT and COPY options, the CLEAR option does not copy keyframes to the paste buffer, and the frames to the right of the cleared keyframes do not move to replace them.

To delete a keyframe and an editing box, select the segment or keyframe and click CLEAR. You can delete only the white editing box segment by right-clicking any animation track.

Note

The operations previously described are performed on branches when you are in Branch mode; if you are in Multi-node mode, the operations affect only the node.

Operations performed by the options LINK, UNLINK, COPY, CUT, PASTE, and CLEAR, as well as the scaling and moving of segments can be reversed using the **History** → **Undo** or **Redo** commands. For more information on the **History** menu cell, see page 849 of the *Reference Guide*.

Linking Keyframes

The LINK option in the window title bar is visible only in the landscape (horizontal half-view) or full screen view. It allows you to link selected keyframes of the same type of relation (such as Rotation with Rotation, Scaling with Scaling, etc.) of a branch object in a hierarchy. Then when the linked keyframes of one node from that branch are edited, the other branch's linked keyframes are also edited. You must be in branch selection mode to use the LINK option, but you can see the effect of that link when you edit any mode of that branch which has linked keyframes.

Note

All objects must be in a hierarchy, and the linked keyframes must be on the same frame.

To link keyframes:

1. Select the objects which have keyframes you want to link so that they are displayed in the DopeSheet. Make sure they are in a hierarchy.
2. Click the sizing icon of the DopeSheet window to display a full-screen view of the DopeSheet, or middle-click it to display a landscape view.

The LINK option is now visible in the window title bar.

3. Select a segment of an object's track which includes the keyframes you want to link to those of another object.
4. Click LINK.

The selected keyframes and the keyframes of the other objects within the hierarchy that are on the same frame are linked.

5. Edit the keyframes as desired: all linked keyframes are also edited.

Note

If you are working in Branch mode, the selected keyframes (and the other keyframes that are linked to them) turn light green.

Unlinking Keyframes

The UNLINK option unlinks keyframes. Like the LINK option, it is visible only in the full-screen or landscape window views.

1. Select a linked keyframe.
1. Click the sizing icon of the DopeSheet window to display a full-screen view of the DopeSheet, or middle-click it to display a landscape view
2. Click UNLINK.

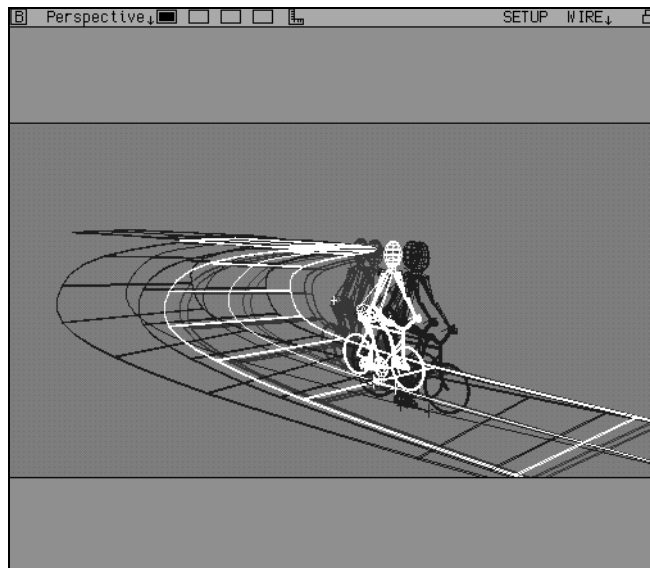
The selected keyframes and all linked keyframes are unlinked.

Note

If you are working in Branch mode, the selected keyframes (and the other keyframes that were linked to them) return to a blue colour. If you are working in Multi-node mode, only the selected keyframe returns to blue.

Using Ghost Frames

The Ghost view mode is an animation display tool that allows you to display a series of frozen snapshots of your animated objects at frames or keyframes behind and ahead of the current frame. Ghost frames act as guides to help you edit keyframes, somewhat like viewing adjacent cells in a traditional animation.



To activate the ghost view, select the Ghost view mode in any window's mode menu in the title bar (Shade, Wireframe, etc.). By default, the current ghost frame is coloured white, the keyframes following it are blue, and keyframes before are red. You can also select any frame to display as static frames. These frames remain in the view regardless of the position of the frame indicator and are used to provide a reference point. You can then manipulate your objects and save their new values.

Refer to page 43 of the *Reference Guide* for a detailed description of the Ghost view mode.

Using the Spreadsheet Window

You can use the Spreadsheet window to help you do large-scale editing of elements in your scene. This allows you to edit multiple elements from within the Spreadsheet window. Rather than selecting each element individually, you can run a query and edit many at once. For example, you can use the Spreadsheet to hide all the lights in your scene at once or change the volume of many identical polygon objects at the same time.

The Spreadsheet is also very useful for selecting and deselecting elements in your scene. The Schematic window may be used for this purpose, but with a large and complex scene, you may not be able to read the text boxes.

For more information about using the Spreadsheet to edit your animation, see page 84 of the *Working with SOFTIMAGE 3D User's Guide*.

C H A P T E R F I V E

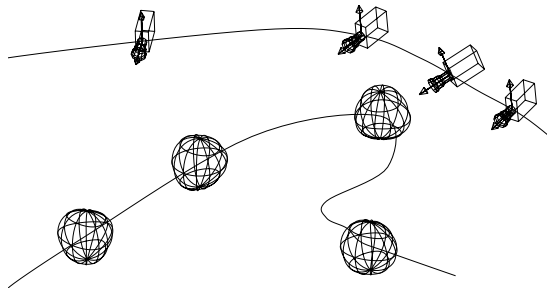
Animating on a Path

Introduction

A path is any curve along which an object can be translated. The position of the object on the path at a specific frame is calculated according to the path's length and the percentage that the object has travelled. One function curve for each translation is created. For example, in an animation, the first frame (the start of the function curve) is considered to be 0% of the path's length and the last frame (the end of the function curve) is considered to be 100%. The function curve is used to control and animate the percentage of the path the object has travelled.

Path animation defines the object's translation as a percentage of the path travelled, compared to explicit translation, in which the object's translation is defined by its explicit x, y, and z coordinates in space.

You can also use a trajectory as a path. For more information about trajectories, see the next chapter called *Plotting Animation*.



Assigning an Object to a Path

There are two main ways to create path animation: you can either define the path first, then place the object on the path, or you can animate the object, and SOFTIMAGE 3D automatically creates a path based on your object's keyframed position(s).

The most common way to create path animation is by drawing the path, then attaching the object to the path. To do this:

1. Create or select an object, such as a cube.
2. Draw a curve using one of the **Draw → Curve** commands in the Model module.
3. Select the cube and choose the **Path → Pick Path** command in the Motion module.
4. Play back your animation. The cube follows the path you have created.

You can also use the mouse in conjunction with the **Translate** menu cells to move the object and save keys with the **SaveKey → Object → Translation** or **SaveKey → Object → Keypath Translation** commands in the Motion module.

For example, to keyframe a cube's translation and then have a path generated, do the following:

1. Create or select an object, such as a cube.
2. Set the end frame of the time line to frame 30 and move the time line pointer to frame 30.
3. Save the first keyframe by choosing the **SaveKey → Object → Translation** command.
4. Move the time line back to frame 1.
5. Translate the cube to wherever you want to put it.
6. Save a second keyframe by middle-clicking on the **SaveKey** menu cell.
7. Play back the animation.

When you associate an object with the path on which it is travelling, the path adopts the wireframe color of that object. You can set this option in the Schematic window by activating the **DEFCOL** option in the title bar. For more information about the Schematic window

options, see page 57 of the *Working with SOFTIMAGE 3D User's Guide*.

Using Custom Effects to Create Paths

You can also use the **Path → Parabolic** command in the Motion module which allows you to animate an object along a parabolic path. You can specify the height of the arc, the start and end frames for the animation, and the source and destination points for the motion. (The source and destination are specified as the SOFTIMAGE 3D centre of user-selected models.) The object then moves along a characteristic gravitational arc, slowing down as it approaches the top and speeding up on descent.

To animate one curve along another, use the **Path → Roll** command in the Motion module. The curve can be unrolled, such as a carpet going down stairs, or rolled like a tire along the road.

For more information on using these effects, see their command descriptions in the *Reference Guide*.

Displaying and Hiding the Path

The **Path → Show All/Hide All** commands allow you to display or hide the paths for all objects in a scene.

You can also use the **Display → Hide** commands or the Spreadsheet to select all the paths and hide them.

Substituting a Path

If you have associated an object on a path for path animation, you can select a new path for a translated object without replacing the translation function curve. This is useful for completely changing your animation.

1. Select an object that is attached to a path.
2. Choose the **Path → Substitute** command in the Motion module.
3. Pick the new path (curve).

The object is assigned to the new path and timing values are preserved.

Changing the Translation's Timing

The **Path → Timing** command in the Motion module allows you to modify the timing for translation of an object on a path. It uses the same dialogue box as the **Path → Pick Path** command. This is useful to match the animation of one object to a musical score, for example, or to another animated object.

It allows you to control the overall length of a path animation (in frames). This is useful because path animation function curves are represented as percentages, and the path animation is relative to the path timing.

1. Select an object.
2. Set the end frame of the time line to the frame you want.
3. Draw a curve using one of the **Draw → Curve** commands.
4. Select the object and choose the **Path → Pick Path** command.
5. Play back your animation. The object follows the path you have created.

To change the timing of the path animation:

6. Choose the **Path → Timing** command.
7. In the dialogue box, change the end frame value to a new frame.
8. Click Ok to confirm or Cancel to exit.

The object now moves along the path until the new frame value.

Converting to and from Paths

You can use certain **Path** commands to change the type of animation of an object. For example, you may have an object that is animated on a path, and you wish to copy its animation to another object. You can convert the path to explicit translation, and then copy the animation of the object. You may also want more control over the animation of your object: convert the path animation (which only expresses the object's movement as a percentage) to explicit translation and you can modify the x, y, and z animation values.

Converting Explicit Translation to a Path

You can convert an explicit translation animation into a path animation by selecting a explicit animation function curve and choosing the **Path → Convert to Path Translation** command.

Converting a Path to Explicit Translation

You can convert a path animation into an explicit translation animation by selecting a path animation function curve and choosing the **Path → Convert to Explicit Translation** command.

When you convert a path to explicit translation, SOFTIMAGE 3D creates a raw function curve (a keyframe at each frame) for the animation. Refer to page 48 for more information about raw function curves.

Converting from Tagged Points to a Path

The **Path → Tag2Path** effect allows you to define curve paths from tagged points on a model. Animate a model, then tag one or more of its points. Tag2Path converts the motion of those points (between the start and end frames you set) into a hierarchy of curves, one for each tagged point. This is very useful for creating motion paths that are linked to particular points of a model, or for converting animated patch or NURBS surface deformations, dynamic simulations, or flock animations into curve paths. For more information on using this command, see the command description in the *Reference Guide*.

C H A P T E R S I X

Plotting an Animation

Introduction

You can plot trajectories for explicit translation, rotation, shape, velocity, and acceleration. This allows you precise control over a moving object.

A trajectory is a curve on which an object can be translated. Each key corresponds to a frame. The first key on the trajectory is called the “start frame.” Trajectory keys define both timing and position (unlike paths); there are no function curves associated with trajectories. The position of an object on a trajectory is calculated according to the key that corresponds to the appropriate frame number on the trajectory. For example, if the start frame of an animation is 2, the fourth key is at frame 6.

You can plot function curves to monitor the animation that is used to control an external device, such as a camera, to ensure that it maintains the proper speed, or so that there are no sudden changes in its velocity. You can also set a maximum allowed velocity or acceleration for the external device by plotting its function curve.

In some cases, you can plot a function curve to create effects such as with Quick Stretch and Expressions. You do not necessarily need to plot a function curve for a Quick Stretch effect, but plotting the function curve allows you to tweak the animation.

Recording an Object's Values

The commands in the **Plot** menu in the Motion module allows you to record the resulting position or orientation of one or many selected objects at every frame. Five types of plots can be generated:

- Trajectories
- Explicit translation function curves
- Rotation function curves
- Shape function curves
- Velocity and acceleration function curves

The **Plot** menu cell is activated at the current frame, which is useful when you want to continue generating values from that specific point. If you want to generate values for the entire animation sequence, be sure to return to the first frame of the sequence. Also, each **Plot** command opens a confirmation dialogue box in which you can specify the start and end frames.

Plotting is useful to generate one type of animation from another type, such as generating the trajectory of a constrained object so that the constraint can be relaxed.

Steps to record an object's position or orientation:

1. Move to the frame at which you want to activate the command.
2. Select the object.
3. Choose the **Plot → Explicit Trans. Fcvs, Rotation Fcvs, or Shape** command. A confirmation dialogue box appears.
4. Select the start and end frame.
5. Click **Ok** to proceed or **Cancel** to cancel the operation.
6. Open an **Fcurve** window and choose the **FcrvSelect → Object → Explicit Translation → All** command to view the three function curves that have been generated.

Plotting Trajectories

A trajectory is a curve on which an object can be translated. Each key corresponds to a frame. The first key on the trajectory is called the “start frame,” which is edited in the Trajectory Timing dialogue box. Trajectory keys define both timing and position (unlike paths); there are no function curves associated with trajectories. The position of an object on a trajectory is calculated according to the key that corresponds to the appropriate frame number on the trajectory. For example, if the start frame of an animation is 2, the fourth key will be at frame 6.

Trajectories apply only to models, but not to cameras and lights. However, you can constrain a camera or a light to a null, then create a trajectory for that null.

Creating a Trajectory

The **Plot → Trajectory** command in the Motion module allows you to record the position of an animated object at every frame by generating a trajectory. On the trajectory, each key corresponds to a frame and represents both position and timing.

1. Animate an object for which you want to create a trajectory.
2. Move to the frame at which you want to activate the command.
3. Select an animated object.
4. Choose the **Plot → Trajectory** command. A confirmation dialogue box appears.
5. Click Ok to proceed or Cancel to cancel the operation.

Assigning an Object to a Trajectory

The **Path → Pick Trajectory** command allows you to link a selected object to a trajectory and specify at which frame you want the object’s animation to start. Trajectories apply only to models, not to cameras and lights.

1. Create a path using any type of curve.
2. Select an object.
3. Choose the **Path → Pick Trajectory** command.
4. Pick a curve following the mouse instructions in the status bar at the bottom of the display.

The curve flashes green and the Trajectory Timing dialogue box is displayed.

5. Set the **Start frame** parameter.

6. Click Ok to confirm the setting and exit the dialogue box.

The object is linked to the trajectory.

7. Using the playback box, play the animation to view your results.

You can edit the start frame of the animation by choosing the **Path → Timing** command to display the Trajectory Timing dialogue box again. Modify the **Start Frame** text box, then click Ok to confirm the changes and exit the dialogue box. When you play back the sequence, the new timing applies.

Converting Explicit Translation into a Trajectory

The **Path → Conv. to Trajectory** command allows you to create a trajectory (one point per frame) for an object that has explicit translation function curves. Trajectories apply only to models, not to cameras and lights.

1. Select an object animated with explicit translation.

2. Choose the **Path → Conv. to Trajectory** command.

3. The explicit path of the selected object is converted to a trajectory.



The **Path → Conv. to Expl. Trans.** command converts the trajectory back to an explicit translation function curve.

Plotting Explicit Translation

The **Plot → Explicit Trans. Fcvs** command in the Motion module allows you to record each explicit x, y, and z position of an object at every frame. A function curve for each axis is created, marking the x, y, and z-axes positions of the object at each frame.

Plotting Function Curves Using Expressions

When you create an expression for an object that contains variables that represent an expression of another object that does not have a function curve, you must use the **Plot → Explicit Trans. Fcvs** command to generate a function curve. This applies when you use the **at_frame** or the **at_time** functions in the Expressions Editor. These functions get values for a frame other than the current frame, and need function curves available to find the proper values at the proper time or frame. If you do not have function curves, you need to plot the animation to create function curves.

The following example uses the oscillation function available in the Expressions Editor to create translation in y for two objects by plotting the function curves of the first object after creating their expressions.

1. Create a sphere and duplicate it (**Get → Primitive → Sphere** and **Duplicate → Immediate** commands). Translate the duplicated sphere so that it is beside the original sphere.
2. Select the first sphere, choose **Info → Selection**, and name it “Harry.” Follow the same procedure for the second sphere and name it “Sally.”
3. Select Harry and choose the **Expressions → Edit** command in the Motion module.
4. In the **Affected Element** text box, type `Harry.etrny`. This means that the expression will affect Harry’s y-position.
5. In the **Expression** text box, type:

```
oscil( 1, 200, 50, 50, SplineInterp, 3, 3,  
SplineInterp)
```

This creates an oscillating effect on the y-position of your sphere using spline interpolation (from frame 1 to 200).

6. Click **Validate**. A message appears stating that the expression is valid. If it is not, then check the syntax in the **Expression** text box. Click **Ok** to apply the expression to Harry.

7. Select Sally and choose the **Expressions → Edit** command.

8. In the **Affected Element** text box, type `Sally.etrny`.

9. In the **Expressions** text box, type:

```
at_frame( Fc - 4, Harry.etrny)
```

This expression states that Sally will follow Harry's explicit translation in y four frames behind.

10. Click **Validate**. If the expression is valid, click **Ok** to apply it to Sally.

You must now plot Harry's explicit translation to make Sally move. If you play back your scene right now, only Harry will move.

11. Choose the **Plot → Explicit Trans. Fcvs** command, and set it to plot from frame 1 to frame 200. Click **Ok**.

12. Select both Harry and Sally (**Select → All**) and play back your animation.

<i>Note</i>	If you change any values in Harry's expression, you must plot the explicit translation curves again, otherwise the animation will not update correctly.

Plotting Rotation

The **Plot → Rotation Fcvs** command allows you to record the x, y, and z-axes orientation of an object at every frame. A function curve for each axis is created, marking the x, y, and z-axes' positions of the object at each frame.

1. Move to the frame at which you want to activate the command.
2. Select the object.
3. Choose the **Plot → Rotation Fcvs** command. A confirmation dialogue box appears:
4. Select the start and end frame.
5. Click Ok to proceed or Cancel to cancel the operation.
6. Open an Fcurve window and choose the **FcrvSelect → Object → Rotation → All** command to view the three function curves.

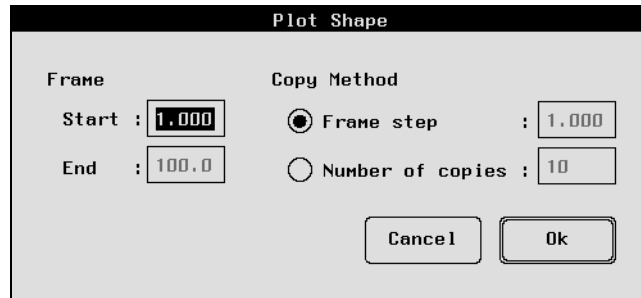
This command is particularly useful for recording the resulting rotation function curves of chain joints for which the corresponding chain effector has been animated, or attached to an animation path. This essentially allows you to convert inverse kinematics into direct kinematics animation.

Make sure to relax or remove the animation of the effector. If not, the playback results in a different animation since both the direct and inverse kinematics is applied to the chain (rotations on the joints define the preference angle for inverse kinematics).

Plotting Shape

The **Plot → Shape** command records shapes of an object previously animated by methods such as lattice or cluster animation, curve or surface deformation, or Quick Stretch. These shapes are generated onto the actual geometry of the object, resulting in shape animation. You can then delete the previous form of animation. For example, you can animate a curve used to create a revolving object, plot its shape, and then delete the animated curve. The animation is retained because it has been copied onto the geometry of the object.

1. Select the animated object.
2. Choose the **Plot → Shape** command. The Plot Shape dialogue box appears.



3. Enter the **Start** and **End** frames of the animation.
4. Choose the **Copy Method**:
 - **Frame step** records the object's shape at the selected frame interval. For example, if you type 2 in the text box, the object's shape is recorded at every second frame.
 - **Number of copies** records the number of shapes you specify at equal intervals during the animation.
5. Click **Ok** to accept the settings and close the dialogue box. The shapes are generated and copied onto the object.

This command is quite useful when you want to save the shapes created using complex combinations of deformation on an object.

Note You cannot use **Plot → Shape** animation for an object that already has shape animation, even if other deformations are applied.

Generating Velocity and Acceleration Function Curves

You can use the **Plot → Velocities & Accel** command to generate linear and angular velocity and acceleration function curves recording the motion from the selected objects that have been animated. These function curves are the same ones used by Quick Stretch to create flex, stretch, and yield deformations on a moving object. You can use them to create a Quick Stretch effect, to fine-tune the movement of an object, or use them within an expression.

You can also use expressions to constrain a parameter to the linear and angular velocity and acceleration function curves.

For more information about animating objects using Quick Stretch commands, see *Quick Stretch* on page 109.

1. Select the object animated with Quick Stretch.
2. Choose the **Plot → Velocities & Accel** command. A confirmation dialogue box appears.
3. Choose the start and end frames of the animation, and click Ok. The linear and angular velocity and acceleration function curves are plotted.
4. Choose the **FcrvSelect → Object → Velocities** command to view the six linear and angular velocity function curves in the Fcurve window, or choose the **FcrvSelect → Object → Accelerations** command to view the six linear and angular acceleration function curves.

Velocity and Acceleration Function Curves in Expressions

You can refer to an object's velocity or acceleration when creating expressions. However, these function curves are not listed in the Function Curve scroll box in the Expressions Editor dialogue box. The following expression statement is an example of using the velocity of an object to create an expression:

```
cube.rotY = cube.velY * 10.
```

Velocity is not calculated in real time so that the velocity of the object can be computed while viewing it in your scene. Because of this, you need to apply Quick Stretch to it. When you apply Quick Stretch, it forces the interactive update of the velocity function curves of the object.

To avoid deforming your object, deactivate all the Quick Stretch effects.

C H A P T E R S E V E N

Deformations

Introduction

When you deform an object, you are modifying its geometry. Deformation is a powerful modeling tool, but you can also animate the various types of deformation, such as shape and cluster animation, lattice deformation, curve and surface deformation, Quick Stretch, and wave effects.

With all deformation tools, the original geometry of an object is not affected if you don't want it to be, so the effects can be dissociated without changing the original model.

You can use shape animation to “morph” objects together, providing that they have the same number of points. Cluster animation is also useful for creating detailed movements, since you have fine control of details in your animation. You can create a shape library of any number of shapes that you want to use. By substituting one key shape in the sequence, you can completely alter the animation layers of deformation.

You can use lattice animation to directly change an object's overall deformation. To use lattices in another way, you can deform an object via its parent. For example, use a cylinder (parent) to deform a sphere (child) to create interesting animation effects as the sphere travels through the cylinder. This is useful to create effects such as a car travelling through a keyhole, or a bug being sucked through a straw.

Curve and surface deformation uses another object to alter an object's geometry. You can use curve and surface deformation to create effects such as a ribbon winding and twisting around a pole. Curve and surface deformations applied to a parent object are inherited by any children objects, so you can animate a whole hierarchy of objects at once.

For example, let's say you want to make a car dance. You also want it to sweat at the same time so that the drops of sweat drip down the surface of the car. To make the car dance, you can deform an entire car hierarchy to a grid and animate its movements by animating the grid. You can then deform selected parts of the hierarchy (the drops of sweat falling down the car) to panels of the car itself.

Quick Stretch is used to deform an object using acceleration and velocity. This lets you achieve cartoon-like effects because the object's geometry is deformed according to its own movement. For example,

you can create a flexing effect by applying Quick Stretch to an object that you have translated. The object appears to bend or flex in the direction of the motion, as if it was due to the resistance of air (or water). You can also use Quick Stretch to obtain a sense of anticipation for the objects to which it is applied.

You can create many different types of waves to deform your objects. Depending on the settings you choose for the wave, you can create shock waves, tidal waves, a drop of water falling on a pond, and other types of natural disturbances that affect the shape of the objects in your scene.

To deform an object's surface by using a texture map, see *Displacement Mapping* in the *Defining Materials and Textures User's Guide*. This, too, can be animated.

Shape and Cluster Animation

Shape animation lets you animate the entire geometrical structure (topology) of an object. You tag and move points on an object to deform it, then save a “key” shape for this. For each variation in the shape you make, you can save a key shape for it to create a deformation key shape library.

Cluster animation is another form of shape animation. You can create clusters (groups) of points on an object and then animate them.

For example, you can animate a face using both shape and cluster animation. Larger facial parts such as the head, eyebrows, and mouth could be shape animated, while you could use cluster animation to alter a part of a larger object, such as changing the position of the cheekbones in response to frowning or smiling.

Note

When using shape interpolation, the objects you are deforming must have the same number of points. To create a metamorphosis between two dissimilar objects, use the **Deformation → ShrinkWrap** effect to create the same number of points on each object. See the *Reference Guide* for more information about this effect, as well as the **Effects → AnimatedShrinkWrap** command in the Motion module.

Key Shapes or Clusters

With shape animation, you can animate the structure by simply tagging and moving points that define the object’s shape, then saving the changes as typical keyframes using the **SaveKey → Object → Shape** command. To assign the different shapes to objects with the same number of points at specific keyframes, use the **Shape → Select Key Shape** command: SOFTIMAGE 3D interpolates between the shapes. See page 93 for more information about defining key shapes.

For clusters, first define a group of points as a cluster using the **Shape → Set Cluster** or **Tag → Set Cluster** command, then use the **Shape → Save Cluster Key** command to keyframe the changes.

Interpolation

When you create shape or cluster animation, SOFTIMAGE 3D interpolates between the key shapes or clusters. The function curves generated from shape or cluster animation fall into two categories, called *average* interpolations and *additive* interpolations.

Creating an average interpolation effect means that the resulting shape or cluster animation is an average value between the key shapes. Average interpolation is the default.

The additive interpolation means that the two shapes are combined. Additive shape or cluster animation is especially useful for facial animation when you want to create extreme facial expressions. For example, adding a surprised expression to a scared expression would yield an expression of open-mouthed terror, whereas averaging the same two expressions would yield a mildly surprised and apprehensive expression.

Types of Interpolation

There are three basic types of average and additive function curve interpolations: *Linear*, *Cardinal*, and *Weighted*.

- **Linear** interpolation means that the physical points of the deformed object will travel in a linear direction.
- **Cardinal** interpolation calculates keyframes on a curved trajectory.
- **Weighted** interpolation creates one function curve for each key shape or cluster. The term *weighted* comes from the fact that you can adjust the animation to emphasize either of the key shapes.

Note

Negative values can be used with this interpolation. This allows you to create an inverted shape when using additive shape interpolation; that is, points move in a negative direction. If the default average shape of interpolation is specified, negative values are assigned a weight of zero.

In shape animation, the advantage of weighted interpolation is that it allows you to interpolate between any shape, not just adjacent shapes. For example, assume you modelled five mouth shapes, and you want to shape animate them. In Linear interpolation mode, you must animate the shapes one at a time in the order you want the deformation to occur. With Weighted interpolation, you can jump from shape 2 to shape 5, and back to shape 3, with a different weight at each shape keyframe. The weighted shape animation, therefore,

lets you define a series of keyshapes, and then combine them regardless of the order in which the key shapes were created.

Defining Key Shapes

The **SaveKey → Object → Shape** command allows you to define keyframes for the shape at specific frames. This method lets you deform the same object and save key shapes for each deformation. This is a simple way of creating a shape animation. The example, *Creating Talking Lips* on page 94, lets you do shape animation in a slightly different fashion.

1. Select an object.
2. Modify the shape of the object by tagging points and moving them.
3. Choose the **SaveKey → Object → Shape** command.
4. Move to another frame and change the object's geometry again.
5. Middle-click **SaveKey** to save another key shape.
6. Repeat the previous steps for every shape variation to be defined.

This creates a function curve which can then be edited to change the animation. This also adds key shapes in the Shape List for that object, which you can see by choosing **Shape → Shape List**.

Assigning Key Shapes

Once you have some key shapes objects defined in your scene, you can start creating your shape animation by defining different key shapes for an object at different frames.

To assign the key shapes to an object:

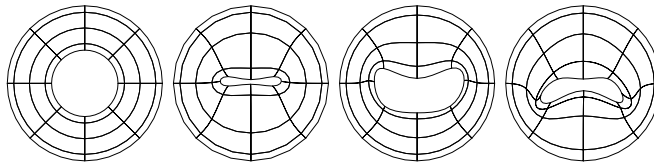
1. Select the object you wish to receive a key shape.
2. Choose the **Shape → Select Key Shape** command.
3. Pick the key shape you want your selected object to assume.
4. Move to another frame on the time line and repeat the process by selecting another key shape.

When you're done, play the animation to see the shape interpolate between the key shapes you set.

Creating Talking Lips

This is an example of how to duplicate objects, deform them, and then assign them to an object to create a shape animation.

1. Choose **Get** → **Primitive** → **Torus** and make it a B-Spline patch.
2. Rotate the torus 90 degrees on the x-axis.
3. Duplicate the torus three times using **Duplicate** → **Repetition** with a translation of 20 units along the x-axis.
4. Change to TAG manipulation mode (press F9). Use the Supra key **t** to tag points and modify each torus into a different mouth shape: an open mouth, a closed mouth, a smiling mouth, and a sad mouth.



5. Change back to Object (OBJ) mode and select the open mouth. Make sure that the time line is at frame 1 and save the first key shape by choosing **SaveKey** → **Object** → **Shape**.
6. Move the time line to frame 20 and save the second key shape by choosing **Shape** → **Select Key Shape** and picking one of the three mouths. A key shape is automatically saved.
7. In the same fashion, save a key shape using the other two mouths at frames 40 and 60.
8. Go back to frame 1 and move the time line pointer to frame 80 using the right mouse button – that way the screen does not get refreshed so the shape stays the same. Save one last keyframe by choosing **SaveKey** → **Object** → **Shape**.
9. Play back the animation.



Exchange the second key shape with the last one and see how the animation changes (see *Replacing a Key Shape* on page 95).

Using a Library of Key Shapes

When you define the key shapes to use to change the animation of an object, these shapes are automatically placed in a shape library. The shapes are named according to the order in which you created them, for example, shape1, shape2, etc., regardless of what the name of the object may be.

The **Shape → Shape List** command allows you to select a shape by name from the Shape List dialogue box. The shape you select is displayed and becomes the active shape. Regardless of the active shape, you can define the shape animation as additive or average, replace a key shape, or delete the active shape from the shape animation.

Replacing a Key Shape

You can also change your entire animation by replacing one key shape with another. For example, if you exchange keyshape1 with keyshape2, your animation reflects the change because the interpolation between shapes will be different.

1. To replace a key shape, choose the **Shape → Select Key Shape** command and select a different key shape.
2. A dialogue box appears asking you whether or not you want to create a new key shape, or replace the existing key shape. Click **Replace**.

To delete a key shape, select it from the list and click **Delete**. The interpolation between the previous and next key shapes reflects this change.

Changing the Interpolation

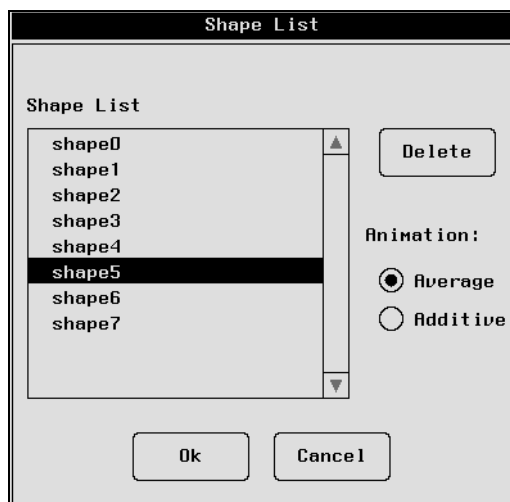
As mentioned earlier, there are two types of interpolation: called **average** interpolations and **additive** interpolations.

Creating an average interpolation effect means that the resulting shape animation is an average value between the key shapes. The additive interpolation means that the two shapes are overlapped, creating a combination of the two keyshapes.

You can set the type of interpolation to be either average or additive between key shapes in the **Shape → Shape List** dialogue box.

1. Select the shape you wish to animate.

2. Choose the **Shape → Shape List** command. The Shape List dialogue box appears.



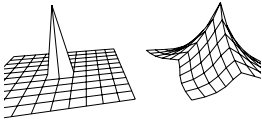
3. Select the type of interpolation you want to use between key shapes (additive or average).

To change the interpolation of the function curve created by shape animation:

1. Select an object animated with shape animation.
2. Choose the desired **Shape → Shape Interp** command: **Cardinal**, **Linear**, or **Weighted**. These commands are toggles.

The only way to view the effects of additive effects in shape animation is to use **Shape → Shape Interp → Weighted**. The reason for this is that additive effects require the key shapes to be overlapped, since the resulting shape is a combination of the percentage “weight” between key shapes.

Resetting an Object's Values



The **Shape → Restore** custom effect in the Motion module “restores” an object’s parameters, such as volume, face angles, edge lengths, or surface area, to their original settings with a minimal distortion of vertex positions. Restore can be used, for example, to aid in the animation of a cloth by automatically restoring the edge lengths of a polygon mesh to their original values at each frame. For more information on using it, see its command description in the *Reference Guide*.

Creating and Editing Clusters

If you are creating very large models, you should consider using cluster animation instead of shape animation. A cluster stores only an index plus a vector for each control vertex, unlike a shape which stores complete geometry information. When you create a cluster for this purpose, make sure to tag all the points of the object.

To create a cluster:

1. Tag points on an object using the **Tag** commands or the **t Supra** key.
2. Choose the **Shape → Set Cluster** command in the Motion module, or the **Tag → Set Cluster** command in the Model module. This command appears in both modules for convenience sake, depending on the type of operation you’re doing (modelling or animating). The Set Cluster dialogue box appears.
3. Enter the cluster’s name.
4. Select the **Create Center** option to create a null centre for the cluster (instead of the object’s centre). This allows you to treat the cluster like an object.
5. Click **New** to add a new cluster name to the cluster list.
6. Click **Ok** to accept the operation and exit the dialogue box.

To edit a cluster, follow the same procedure as above, but click **Replace** to update the points assigned to the current active cluster.

Creating Clusters on a Sphere

In this example, the sphere represents a head, and clusters are created to represent cheekbones on a face, which will be animated separately from the animation of the head.

1. Choose **Get → Primitive → Sphere**. Accept the default values.
2. in the appropriate place on the sphere (you know where cheeks are!), set a cheekbone cluster by choosing **Shape → Set Cluster**. Give the cluster a name.
3. Select the **Cluster Centre** option to create a cluster centre. This allows you to control the animation of the cluster by moving its center (null).
4. Click on **New** to exit the Set Cluster dialogue box and create the cluster.
5. Set another cluster for the other cheekbone following the same procedure.
6. Choose **Shape → Cluster List**. There should be two clusters listed in the scroll box. You can now animate the cheekbones using these two clusters.

Saving Cluster Keys

Once you have created one or more clusters, you can create a cluster keys for animating.

The **Shape → Save Cluster Key** command allows you to define cluster animation on an object using cluster keys at specific frame numbers.

1. Select an object with an active cluster.
2. Select a frame using the time line pointer.
3. Transform the cluster as desired, such as moving or scaling it.
4. Choose the **Shape → Save Cluster Key** command (or **SaveKey → Object → Cluster Key**).
5. Repeat this procedure for every cluster variation to be introduced at every new frame.

A function curve is created for the cluster key, which can then be edited to change your animation.

Viewing or Deleting Cluster Keys

You can view the next or previous cluster key, or delete the current cluster key.

1. Select an object with an active cluster.
2. Choose the **Shape → Cluster Keys → Next Key/ Previous Key** subcommand to view them.

3. Choose the **Shape → Remove Key** subcommand to delete a cluster key.

Removing All Cluster Animation

To delete the animation for a cluster, select it and choose the **Shape → Remove All Cluster Anim** command in the Motion module.

Selecting or Deleting a Cluster

You can select a cluster by name from the Cluster List dialogue box. The points in the cluster are tagged and become the active cluster. You can animate the object by moving the tagged cluster. You can also delete the cluster from the cluster list.

1. Select an object with clusters.
2. Choose the **Shape → Cluster List** command in the Motion module. The Cluster List dialogue box appears.
3. From the Cluster List, choose the cluster name of the cluster you want to activate, or select **Activate All Clusters Animation**. These become the tagged clusters on the object.
4. You and also click **Delete** to delete the active cluster or **Animation: Delete** to delete the cluster's animation.
5. Click **Ok** to accept the selection or any changes or **Cancel** to exit the dialogue box without making any changes.

Changing the Interpolation Between Clusters

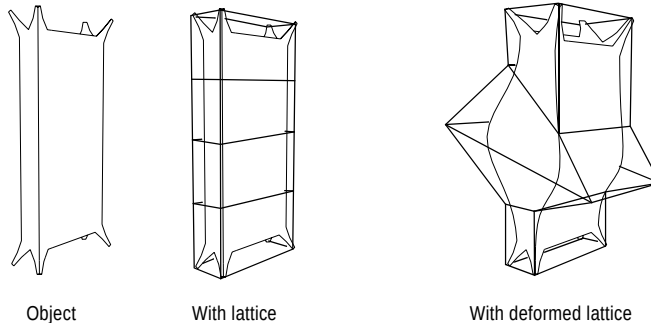
As mentioned earlier, there are two types of interpolation: called **average** interpolations and **additive** interpolations.

Creating an average interpolation effect means that the resulting animation is an average value between the clusters. Unlike shape animation, *all* cluster interpolation types (Linear, Cardinal, or Weighted) can be additive or averaged. In other words, you can add cluster animation from different clusters if some points overlap, even those sets of clusters that are *already* animated with Linear, Cardinal, or Weighted interpolation.

1. Select an object with an active cluster.
2. Choose the desired **Shape → Cluster Interp** command: **Cardinal**, **Linear**, or **Weighted**. These commands are toggles.

Animating with Lattices

Lattices allow you to deform an object or a group of objects linked in a hierarchy. A lattice is a bounding box with a variable resolution. Every subdivision on the lattice is associated with a number of points on the object. By moving the lattice points, you can deform the object in the same way.



Although lattices are used mainly for animation effects, they can also be used as a modelling tool (see *Deforming with Lattices* on page 294 in the *Modelling User's Guide* for more information about types of lattices).

Creating a Lattice

1. Select an object or a branch of a hierarchy.
2. Choose the **Lattice → Node → Create** command in the Model module if an object is selected, or the **Lattice → Branch → Create** command in the Model module if a hierarchy is selected. The Create Lattice dialogue box is displayed.
3. For each axis (x, y, and z), you can set the **Subdivisions** (resolution). Low resolution allows a more global, “crude” deformation effect. High resolution allows more localized deformations of the latticed object.

You cannot change the resolution of a lattice after it is created, so make sure the resolution is correct when you create it.

4. Select the **Interpolation** type to be either Linear or Curve. Curve interpolation lets you achieve a smoother deformation.
5. When you create the lattice, it is highlighted in yellow and surrounds the object you selected. In the Schematic window, NL and BL indicate node and branch lattices.

Deforming a Lattice

After you have created a lattice, you can move the points on it as you would for any other object.

1. Create a node or branch lattice.
2. Tag points on the lattice using **t Supra** key.
3. In TAG mode, transform the tagged lattice points using the **ScaleXYZ**, **RotXYZ**, or **TransXYZ** menu cells. You can also choose **Edit → Move Point** to move individual points on lattices.

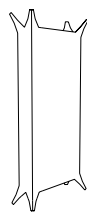
The object deforms accordingly to the deformation of the lattice points. Now you can animate these deformations.

Animating a Lattice

There are two basic ways of animating with a lattice: the lattice directly deforms the object it surrounds, or it deforms the parent of the child object you want to deform. The first method is the simplest, but the second method allows you to do quite interesting effects.

To animate a lattice in the simplest manner:

1. Select the object you want to deform
2. Create a lattice using the **Lattice → Node → Create** command in the Model module.
3. Deform the lattice's points.
4. To animate the effect, use the **SaveKey → Object → Node Lattice** command.
5. Go to the next frame, deform the lattice, save a keyframe, and continue in the manner until the animation is done.



Frame 1
Original object



Frame 10
Save keyframe with
lattice deformation.



Frame 70
Save keyframe



Frame 100
Save keyframe

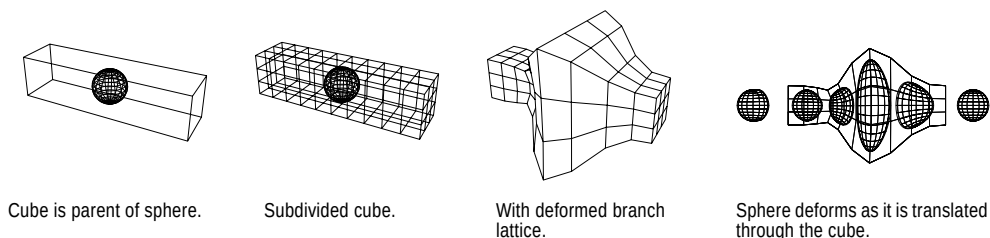
Using a Lattice to Deform a Child Object

The following example uses a lattice to deform a parent object to change the animation of the child object. The important thing about using this method is that you must create a larger parent “container” object in which the child object moves. You then deform the “container” with the lattice, and save keyframes for the *explicit translation* of the child object, not the lattice. As you deform the container parent object, the child deforms accordingly.

This example shows how to deform a sphere going through a tube:

1. Get a sphere with a radius of 1. Get a cube and scale it on the x-axis to be fairly long like a tube.
2. Make the cube the parent of the sphere (choose the **Parent** menu cell in the Model module and select the sphere as the child).
3. Choose **Lattice → Branch → Create** in the Model module. Use the default values for x, y, and z and click Ok. The yellow lattice appears on the cube.
4. Change the shape of the lattice by tagging various points (t Supra key) in TAG mode and moving the points.
5. Select the sphere and animate its position along the x-axis by saving keyframes at frames 1 and 100 with the **SaveKey → Object → Explicit Translation → X** command.

The sphere conforms to the shape of the cube as it travels along the x-axis.



You can also animate the lattice on the cube by changing its shape and using the **SaveKey → Object → Branch Lattice** command and then editing the animation in the Fcurve window.

Animating Local Deformations

To animate local deformations on a lattice, you can use the **Effects → LocalLattice** command in the Motion module. The **Effects → BranchLattice** command is a version of LocalLattice that you can apply to a hierarchy. These effects only work if you have created the lattice with the **Lattice → MakeLattice** command in the Model module. See page 936 of the *Reference Guide* for more information.

You can position the lattice anywhere on the object being deformed, and apply the effect only to the area within the lattice bounding box. The deformation it produces, as well as the position of the lattice on the object being deformed, can be animated.

The position of the lattice is determined by its scale, rotation, and translation values, and like the SOFTIMAGE 3D lattice, the deformation produced is defined by editing its points.

With LocalLattice, the lattice is a separate object to which you can apply other effects, such as path and shape animation, wave, or deformations.

For more information on using these effects, see their command descriptions in the *Reference Guide*.

Moving Between Key Shapes

You can use the **Lattice → Previous** and **Next** commands in the Motion module to move between lattice keyframes. This is similar to using the **K** option in the playback box to move between keyframes.

Removing Lattices and Lattice Animation

You can remove a lattice from the object or branch of a hierarchy, and return the object to its original shape. If the lattice is animated, the animation is also removed.

1. Select an object or branch of a hierarchy with a lattice.
2. Choose the **Lattice → Node → Remove** command in the Model module if an object is selected, or the **Lattice → Branch → Remove** command (also in Model) if a hierarchy is selected.

The lattice is removed from the object or branch of a hierarchy.

To remove a lattice keyframe, choose the **Lattice → Remove Key** command in the Motion module.

Deactivating a Lattice

You can temporarily deactivate the lattice and so that it and its effects on the object or branch of a hierarchy are no longer visible.

1. Select an object or branch of a hierarchy with a lattice.
2. Choose the **Lattice → Node → Deactivate** command in the Model module if an object is selected or the **Lattice → Branch → Deactivate** command (also in Model) if a hierarchy is selected.
3. To reactivate a lattice and its effects on the object or branch can be choose either the **Lattice → Node/Branch → Create** command (in Model) or the **Lattice → Node/Branch → Info** command (also in Model).
4. Select the **Activate** option in the Lattice Info dialogue box.

Resetting a Lattice Position

You can temporarily return the lattice to its original shape. This command is used most frequently with animated lattice shapes.

1. Select an object or branch of a hierarchy with a lattice.
2. Choose the **Lattice → Node → Reset** command in the Model module if an object is selected or the **Lattice → Branch → Reset** command (also in Model) if a hierarchy is selected.
3. If you click the time line, the lattice is refreshed and takes the current frame position.
4. If you like, you can save a keyframe for this new shape to change the lattice animation.

Animating Curve and Surface Deformation

You can associate objects in such a way that one is deformed according to the shape of the other one. The deforming tool can be any type of curve, or a patch or NURBS surface object. You can deform a single object, a whole hierarchy of objects, or selected objects in a hierarchy.

Curve or surface deformation can also be used as an animation effect. By saving keyframes of the transformations (scaling, rotation, and translation), the object's deformation along the other object can be animated. For example, using curve and surface deformation, you can create animation effects such as a rain drop sliding down the surface of a statue.

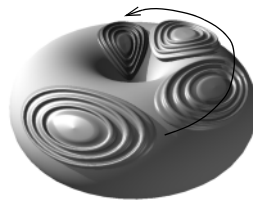
A model with a hierarchy provides more possibilities for deformation animation. You can apply the deformation on the parent, in which case the deformation is transmitted to its children as if the model was a single piece. You can also animate the children individually: any deformation you apply explicitly to a child is added to the deformation inherited from the parent. For this type of animation, use the object's transformation values when saving a key.

You can also fit a child to a deforming object. This would be useful if you wanted an object to deform and remain in the same place. In this case, you would animate the child with deformation transformations and compensate with object transformations on the parent (a null). All animations created with transformations, shapes, or lattices can be combined to produce complex animations.

For more information on the basics about deforming curves and surfaces, see *Deforming by Curves and Surfaces* on page 298 of the *Modelling User's Guide*.



Object animated along a curve by translating in y.



Object animated along a surface (NURBS torus) by translating in x and z.

Creating a Curve or Surface Deformation

The object is deformed according to the shape of the selected curve or surface and is controlled by deformation transformation. For more information on the proper orientation and positioning of objects, see *Setting Up for Deformation* on page 298 of the *Modelling User's Guide*.

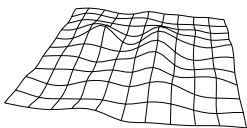
This process applies to both curves and surfaces: simply choose the appropriate **Deformation** command.

1. Create an object to deform, as well as the curve or surface by which you want it deformed.
2. Select the object you want to deform.
3. Choose the **Deformation** → **By Curve/Surface** → **Node/Branch** → **Create** command in the Model module.
4. Pick the deforming curve or surface object.

Deforming along a Surface

You can animate your deformations by translating, rotating, or scaling the objects along their deformed surface.

The following example outlines the steps for animating a deformed object. This shows you how you can make a grid conform to another bumpy grid using surface deformation.



1. Create an object which you will use as your deforming object (in this case, a grid). Choose **Get** → **Primitive** → **Grid** and choose Patch with B-Spline for both x and z directions. Click Ok.
2. Choose the **Effect** → **Randomize** command and use the default values for x, y, and z to bump up the surface. Click Ok to accept the changes.
3. Tag some of the points (use the t Supra key) and using TAG mode, deform the surface some more (**Edit** → **Move Point** and translate it).
4. Create an object you want to deform. Choose **Get** → **Primitive** → **Grid** using the same settings as before.
5. Give this object a material to distinguish it from the deforming object (choose the **Material** menu cell and create a colour such as purple). Click Ok to accept the values in the Material dialogue box.

6. Choose **Deformation → by Surface → Node → Create** in the Model module and pick the grid that you deformed (the first one you created). This creates a deforming relation between the two grids.
7. At frame 1, choose **SaveKey → Object → Node Surface Deformation → All** in the Motion module to save a keyframe for all deformations.
8. Go to frame 20, change the rotation and translation of the purple grid you want to deform so that it starts to move across the deforming grid. Save another keyframe here (middle-click on **SaveKey**). Do this step for frames 40, 60, 75, and 100.
9. Hide the deforming object (the first grid you created) by choosing **Display → Hide → Toggle & Desel. Hidden**.
10. Play back the animation and watch the grid float through the air!

Deforming Along a Curve

In the following example, you are deforming a cylinder along a curve.

1. Create a cylinder and give it some definition so that you can see it deform along a curve. To do this, choose **Get → Primitive → Cylinder** and set the **Height** to 10, and the **Step** for Longitude and Latitude to 15.
2. In the Front window, choose **Draw → Curve → Cardinal** and draw a vertical curvy path for the cylinder to follow (you may want to zoom out so that the curve can be long enough for the cylinder to deform along).
3. Select the cylinder.
4. Choose **Deformation → by Curve → Node → Create** in the Model module and pick the curve (without pressing the space bar). The cylinder should now be at the beginning of the curve.
5. Translate the cylinder on the y-axis. You will notice that the curve has in effect become the y-axis for the cylinder.
6. To animate the deformation of your cylinder, go back to the beginning of the curve and save the first keyframe by choosing **SaveKey → Object → Node Curve Deformation → Translation**.

7. Move the time line to frame 100, translate the cylinder to the end of the curve, and save a second keyframe by middle-clicking on the **SaveKey** menu cell.

8. Play back the animation.

The curve (path) can always be edited by moving its control points.

Removing a Deformation

You can remove the object from the associated deformation curve or surface and returns it to its original shape.

1. Select the object you have deformed.
2. Choose the **Deformation → By Curve/By Surface → Node/Branch → Remove** command.

Deactivating a Deformation

You can temporarily suspend deformation of the object. This option is used when you need direct control over the object's transformation or when you want to manipulate the object's centre or points.

Deformation is reactivated by choosing the **Deformation → By Curve/By Surface → Node/Branch → Create** command.

1. Select the object you have deformed.
2. Choose the **Deformation → By Curve/By Surface → Node/Branch → Deactivate** command.

Resetting a Deformation

You can reset all deformation transformations to the original values set when the deformation was created.

1. Select the object you have deformed.
2. Choose the **Deformation → By Curve/By Surface → Node/Branch → Reset** command.

Quick Stretch

Quick Stretch can be used to deform any model according to its own movements. It is a very quick and easy way to achieve cartoon-like deformation effects.

The following pages explain the principles of Quick Stretch animation, summarize the various options you can apply and illustrate some of the results you can achieve. Some tricks are included for achieving the effects of anticipation and of twisting.

Types of Motion

Quick Stretch uses the following aspects of an object's motion to deform the object.

- Linear velocity
- Linear acceleration
- Rotational velocity
- Rotational acceleration

A moving object has speed or *velocity*. If it is changing speed, then it also has *acceleration* or deceleration. For example, a car moving at a steady 100 km/h has a velocity of 100, but an acceleration of 0.

Velocity and acceleration can each be further divided into two types: *linear* and *rotational*. For example, a car that moves along a straight line has only a linear motion, whereas a ball which spins on the spot has only rotational motion. Regardless of the type of motion, the faster the object is moving, the more it is deformed.

For an example using acceleration to feign anticipation, see *Quick Stretch Tricks* on page 116.

Overview of Quick Stretch

To use Quick Stretch, follow these steps:

1. Create an object to which you want to apply Quick Stretch. Make sure that it has fairly good definition (subdivisions) so that the effects are noticeable.
2. Choose the **QStretch** → **Setup** command in the Actor module to set the type of deformation you want to use (Flex, Stretch, or Yield). See page 110 for more information about the types of deformation you can use.

3. Move the object around the screen to view the results of applying Quick Stretch.

Setting Up

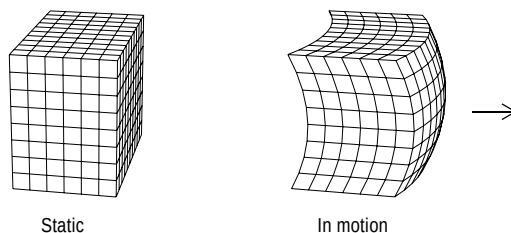
Whichever type of motion you give to an object, it can be deformed in one or more ways: it can flex, stretch, yield, or any combination of the three.

Each of these deformation “modes” can be weighted independently of the others, but their effects are additive.

Each effect is designed to be tweaked separately as much as possible, so that when you use them all together, the resulting deformation is easier to predict.

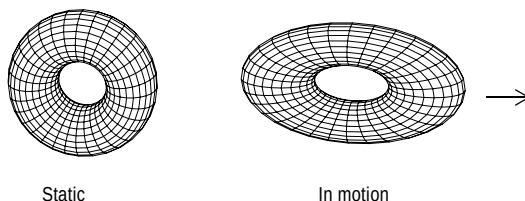
Flexing

A flexible object when moving rapidly in one direction appears to bend or flex in the direction of the motion, due to the resistance of the air (or water).



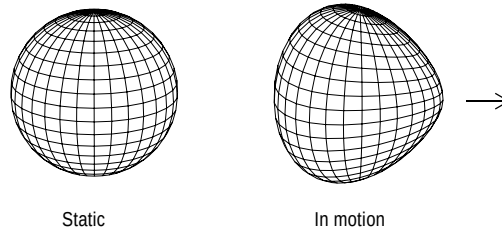
Stretching

In the case of a cartoon “squash and stretch” effect, the object usually elongates in the direction of the motion, and becomes thinner in the other directions.



Yielding

Depending on the mass assigned, a moving object might appear to bulge due to the internal displacement of its mass.

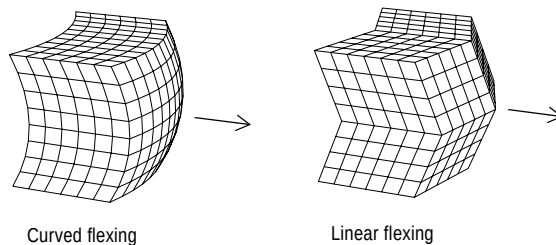


Scaling and Locking Deformation

You can apply a scaling factor and set a maximum deformation for each deformation mode.

For more information on the parameters called Sensitivity and Maximum, refer to **QStretch** ➔ **Setup** on page 1179 of the *Reference Guide*.

For each deformation mode, further options help you decide the shape of the deformed object. For example, say you have decided to deform an object by flexing, you can also decide whether the flexing will give a curved shape or linear shape, reflecting the type of material of which the object is “made.”



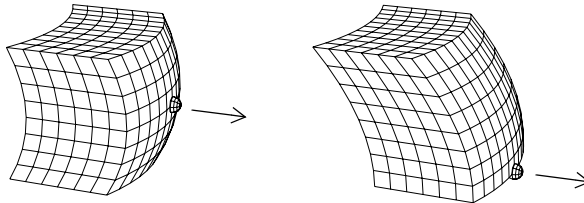
Origin of Deformation

An object in Quick Stretch is moved by some kind of force. It might be pushed or pulled, it might have an “engine” that makes it move on its own, or it might fall from the effect of gravity. In all cases, the forces that make the object move are applied somewhere on the object, and this affects how a flexible object deforms.

For example, if a character hits a box, you would expect it to deform from the point where it was hit. If a building faces such a strong wind

that it should bend a little, the deformation should originate from the ground up, since you expect the building to be attached to the ground.

In Quick Stretch, the origin of the deformation is the centre of the object, regardless of whether the centre is located inside or outside the object's control vertices: two control icons are displayed at the centre and cannot be disassociated from it.



Centres: Similar cubes flex differently, according to the location of their respective centres (shown at base of each arrow).

Applying and Editing Parameters

To apply Quick Stretch, you must select an object as a node or branch. Make sure the objects have enough definition (subdivisions) to give an interesting result. For example, a default cube should be subdivided a reasonable amount to show the curvature of the deformations.

You can edit the Quick Stretch effects by:

- Modifying the Quick Stretch Setup; that is, modify the settings and verify the effects as before (direct manipulation, playback).
- Editing the object's animation.
- Plotting velocities/accelerations and editing these function curves. This allows you to modify the actual motion vectors of the object.
- Using the Quick Stretch control icons to change or animate the amplitude or direction of the velocity motion vectors.

With **QStretch → Setup**, you set various parameters related to the types of motion and deformation already described. These can be combined and weighted independently for a variety of effects.

1. Select the object to which you have applied Quick Stretch (or still wish to apply).
2. Choose the **QStretch → Setup** command. The Quick Stretch Setup dialogue box appears.

3. Select the **Current Component** to the appropriate Velocity or Acceleration option.
4. Select the Flex, Stretch, or Yield Modes, or any combination of them.
5. Choose the appropriate Profile or Volume option for each one.
6. Change their other parameters, such as the object's sensitivity to each of these settings and maximum deformation levels for each.
7. Click Ok to accept the changes. Move your object around the screen to view the changes in Quick Stretch effects.

Fine-tuning the Results

If you want to do some fine-tuning, you can edit the parameters in the Setup dialogue box some more. Alternatively, you can change the linear or rotational velocity components with two 3D control icons that can be animated; see *Using Quick Stretch Control Icons* on page 114 and the **QStretch** → **Setup** command on page 1179 of the *Reference Guide*.

- In Quick Stretch Setup dialogue box, you could change:
 - The deformation modes applied.
 - The values for each Sensitivity parameter and Maximum values.
- Using the control icons, you could change:
 - The amplitude or direction of the linear/rotational velocity vectors.

Furthermore, if you have plotted velocities and accelerations using the **Plot** commands in the Motion module, you can tweak these curves in the Fcurve window. For more information, see *Plotting the Animation* on page 115, as well as *Editing Acceleration Curves* on page 117.

Viewing the Results

As a deformation tool, Quick Stretch can be used as both a modelling and animation tool. That is, you can use Quick Stretch deformation to deform a static object or an animated object.

You can view the results of Quick Stretch deformation interactively or by plotting the motion and later editing it. The following methods are discussed in detail here:

- Direct manipulation of the model to which it has been applied.

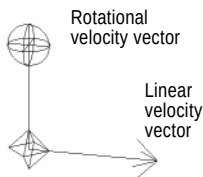
- Using QStretch control icons.
- During animation playback.
- Using function curves. Using the **Plot → Velocities & Accel.** command in the **Motion** module, you can record the motion vectors at each frame and plot them into function curves. Once they are plotted, Quick Stretch uses them and they can be edited.

Direct Manipulation

If you want to quickly see what sort of deformation effects you can achieve, just apply Quick Stretch to an object and then move it around to see the effects interactively.

This kind of interactive deformation is not possible if you have explicitly plotted velocities and acceleration.

Using Quick Stretch Control Icons



Once you apply Quick Stretch to an object, two control icons are created and attached to your object. These icons represent the linear and rotational motion vectors.

The arrow tip icon modifies the linear velocity vector of the object; the spherical tip controls the rotational velocity. There are no control icons for the acceleration components.

The y-scaling of these controls determines the amplitude of the velocity; their orientation controls the direction of the velocity. They are constrained to the speed of the object. As you move a model interactively, observe how these control icons match the direction of the current velocity vectors.

Note If you select one of the control icons instead of the object, and then choose **QStretch → Setup**, the Quick Stretch Setup dialogue box is displayed with the corresponding effect (linear or rotational velocity).

- **Static Objects:** If an object is static, you can still use these control icons to deform the associated object. They control the amount and direction of the deformation. Since these controls are similar to null objects, you can change the orientation and the scaling as you would any object.
- **Animated Objects:** If the object is animated, you can still keyframe, for instance, the amplitude of the velocity through the y-scaling of the control icon. However, this might be more useful when the object is not animated.

Playing Back Your Animation

If the position or orientation of an object is animated (for example, an object animated on a path), the Quick Stretch deformation effects are computed at each frame according to the motion of the object.

Since motion vectors are computed on the fly during playback of an animated sequence, the deformation that occurs is different if you play the animation backward!

Similarly, if you jump from one frame to another one, a huge deformation may occur if the difference in position implies a very large velocity or acceleration. In this case, the geometry returns to normal at the next refresh.

To prevent these effects, once you are satisfied with the results, you should record the motion vectors, as described in the next section.

Plotting the Animation

By default, motion vectors are computed on the fly for each object affected by Quick Stretch. Once you are satisfied with the motion, record the motion vectors using the **Plot → Velocities & Accel.** command in the Motion module.

This generates 12 function curves for the linear and rotational velocities and accelerations; these function curves can be displayed in the Fcurve window using either the **FcrvSelect → Object → Velocities** command or the **FcrvSelect → Object → Acceleration** command in the Motion module.

1. Select the “Quick Stretched” object for which you wish to edit the velocity function curve.
2. Choose the **FcrvSelect → Object → Velocities** command. The Fcurve window opens and the velocity function curves for that object appear. Edit this like any other function curve (see *Editing Function Curves* on page 30).

The Fcurve window and the appropriate function curve(s) appear. The velocity function curves are raw function curves, meaning they have one key point per frame. You can convert the function curve to a spline to reduce the number of key points by choosing the **FcrvEdit → Fit Spline to Fcurve** command.

You can edit these function curves to change the motion vectors and tweak a deformation. You can then plot these vectors again at any

time; of course, this will override the editing you may have made on them. Quick Stretch uses any available function curves instead of the values computed on the fly.

Before rendering, previewing, or saving a scene, a warning is displayed if there are Quick Stretched objects without plotted velocity/acceleration curves.

Note

If you want to render your scene in an order other than the full sequence, you must first plot the explicit motion curves; otherwise, the deformation will be incorrect (see *Playing Back Your Animation* on page 115).

Removing Quick Stretch from an Object

You can use the **QStretch → Relax** command to remove the Quick Stretch effect from an object, as well as the control icons associated with it.

Quick Stretch Tricks

You can use the acceleration components of a moving object to obtain a sense of anticipation for Quick Stretched objects.

If you have an object that goes from A to B in a straight line, its speed increases and then decreases until it comes back to rest.

In this case, if only the velocity component is used for the deformation, the object always deforms in one direction, since the velocity is always positive. It will look as if a wind (increasing, then decreasing) is applied to the object to make it deform.

To make it look more as if the motion originates from the object, traditional animation uses what is called *anticipation*. Something similar to this can be obtained by using the acceleration component since the object decelerates in the second part of its motion, thereby adding a deformation in the opposite direction of the velocity.

By setting the amplitudes properly, you can make it appear that the object is braking, and as if a part of the mass of the object is trying to continue its motion despite the deceleration.

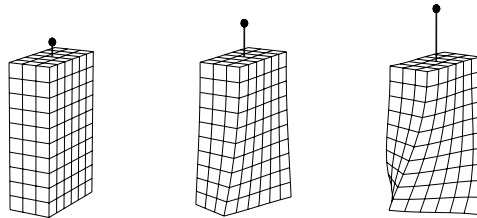
Editing Acceleration Curves

The Acceleration component may include discontinuities or sudden large values when the velocity comes to a stop.

To remove these discontinuities, you may have to plot the motion vectors and edit the acceleration curves as described in *Plotting Motion Vectors* on page 115.

Twisting an Object

1. Select an object. For this example, choose **Get** → **Primitive** → **Cube**. Apply the **Effect** → **Subdivision** command. This subdivides the cube so that the results of Quick Stretch are more obvious.
2. Apply Quick Stretch by choosing the **QStretch** → **Setup** command and setting the **Rotational Velocity** current component. Click Ok.
3. Select the rotational velocity control icon (the sphere tip) and scale it in y.
4. When you rotate the object, it should twist along the axis of the control icon, starting at the centre of the object.
5. In the Quick Stretch Setup dialogue box, adjust the **Sensitivity** values and **Maximum** deformation values for the rotational velocity effects.



Putting it All Together

Here's a short example to show you how to use a few options together.

1. Choose **Get** → **Primitive** → **Cylinder** and set the Longitude and Latitude step to 20. Click Ok.
2. Choose the **QStretch** → **Setup** command. In the Quick Stretch dialogue box, select the **Flx** (Flex) and **Str** (Stretch) options for the **Velocity** current component in the Modes: overview area. Click Ok.

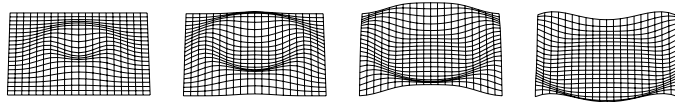
Note

The Flex option bends the cylinder in the direction of the motion and Stretch elongates the cylinder in the direction of the motion.

3. Move the cylinder around to view the Quick Stretch effects.
4. To exaggerate the stretching effect, choose the **QStretch** → **Setup** command again and set the sliders for **Stretch** to the maximum value.
5. To animate the cylinder over 100 frames, set keyframes for rotation and translation. Make sure that the cylinder is moving quickly, otherwise it might be difficult to see the results of Quick Stretch.
6. Play back your animation.

Making Waves

You can create shock waves, water waves, or other types of natural disturbances by using the **Effects → Wave** command. You can set the parameters of waves such as the wave type, the velocity and acceleration of the wave, the displacement of the wave, and the falloff rates. You can also use the wave effect to animate objects passing through a deformation, or to animate the surface of a collision after you have created a dynamic simulation.



Grid deformed by a circular wave. Profile of wave was defined by the curve at right.

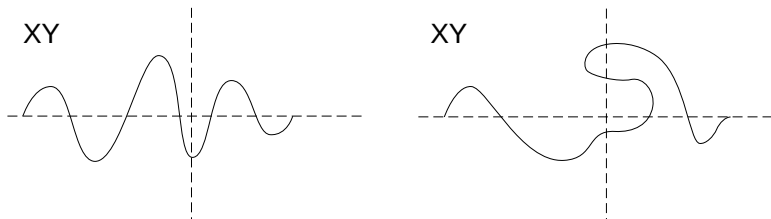
See the *SOFTIMAGE 3D Tutorial Workbook* for examples of wave animation.

Creating a Wave

The **Effects → Wave → Create** command allows you to create a wave, which can then be attached to an object to produce a deformation.

1. Create a curve representing the profile of the wave.

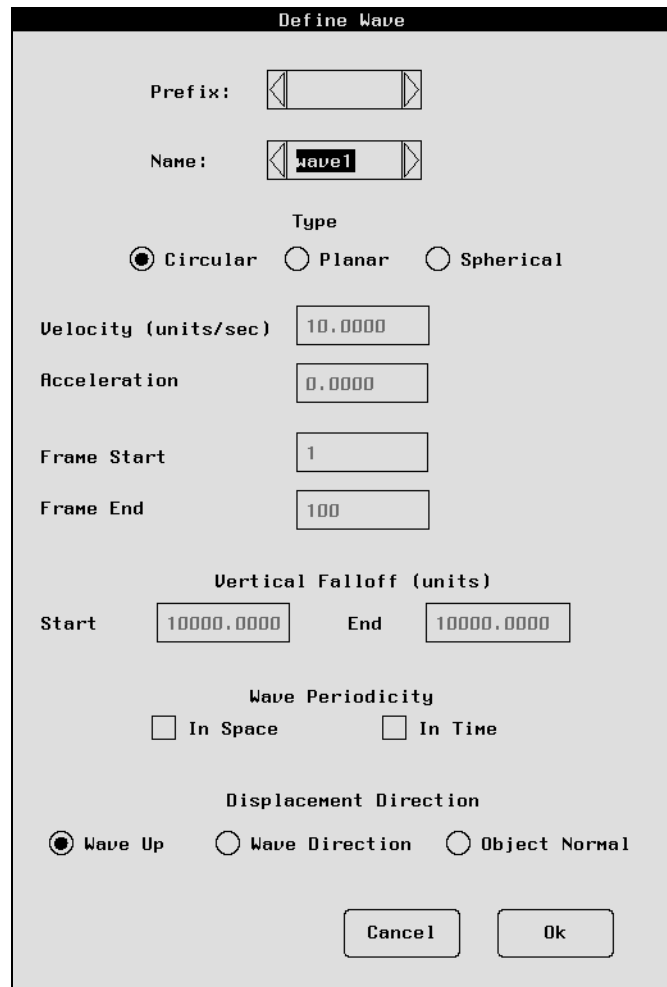
The curve should be drawn in the xy plane, and should not double back on itself (see illustration). If you create a wave with an invalid profile curve, a warning message is displayed in the status bar. The magnitude of the disturbance is governed by the height (y values) of the profile curve. The portion of the curve with $x > 0$ affects objects as soon as the wave starts to act. The portion of the curve with $x < 0$ affects objects as the wave propagates. This part of the disturbance appears to be coming out of the origin of the wave over time.



Valid Profile Curve

Invalid Profile Curve

2. Choose the **Effects** → **Wave** → **Create** command in the Motion module. The Define Wave dialogue box appears.



The image shows a 'Define Wave' dialog box with the following fields and options:

- Prefix:** A text field with a left arrow and a right arrow.
- Name:** A text field containing 'wave1' with a left arrow and a right arrow.
- Type:** Three radio buttons: ☒ Circular, ☐ Planar, and ☐ Spherical.
- Velocity (units/sec):** A text field containing '10.0000'.
- Acceleration:** A text field containing '0.0000'.
- Frame Start:** A text field containing '1'.
- Frame End:** A text field containing '100'.
- Vertical Falloff (units):** Two text fields: **Start** containing '10000.0000' and **End** containing '10000.0000'.
- Wave Periodicity:** Two checkboxes: ☐ In Space and ☐ In Time.
- Displacement Direction:** Three radio buttons: ☒ Wave Up, ☐ Wave Direction, and ☐ Object Normal.
- Buttons:** 'Cancel' and 'Ok' buttons at the bottom right.

3. Select the type of wave desired (circular, planar, or spherical) and set the parameters as desired. Depending on the type of wave you select, you can create effects such as: water droplets (circular), tidal waves (planar) or a wave as a result of an explosion (spherical).

You can also define other parameters such as: the velocity of the wave, its acceleration, the fall-off rate and the start and end frames.

For a complete description of all parameters, see the **Effects → Wave → Create** command description on page 520 in the *Reference Guide*.

4. Click **Ok** to accept the settings or **Cancel** to exit the dialogue box.
5. A wave icon is created which lets you select it to animate, edit the wave, assign a new profile, etc.

Attaching a Wave to an Object

The **Effects → Wave → Attach → Node/Branch** command allows you to attach a wave to a node or branch object. During playback, when you attach a wave to a node object, the object's points are modified by the wave. If you attach the wave to a branch hierarchy, all the objects in the hierarchy are modified by the wave.

1. Select the wave icon.
2. Choose the **Effects → Wave → Attach → Node/Branch** command.
3. Pick the node object.

The object flashes showing that the wave has been attached.

Note

When a wave is attached to a node, it becomes a child of the node. If the object is moved or animated, the wave (and its effect) follows the object. Normally, this is what you want. However, if it is desired that the wave remain stationary while an object moves through it, this can be accomplished by attaching the wave as a branch wave to a null parent of the object. Refer to the **Effects → Wave → Attach → Branch** command.

The wave's effect is computed in the space defined by the top node of the branch. The wave moves with this node, and the effect on the children of the node is computed accordingly.

Detaching the Wave

After attaching a wave, you can also detach it from an object to remove the wave's effect. The wave icon remains on the screen and may be attached to the same object again or a different object.

1. Select the wave icon you want to detach.
2. Choose the **Effects → Wave → Detach** command.

The wave is detached from the object and the object is restored to its original geometry.

Note

The **Show → Points** command has no effect on objects that have a wave associated with them. These points cannot be displayed. If shape animation or point modification is desired, all waves should be detached from the model. This also applies when changing the step resolution of patch or NURBS object, although the **Show → Surface** command should work in either case.

Animating the Wave

You can animate your wave by keyframing the scaling, rotation, and translation of its icon using the **SaveKey → Wave** commands. For more information about keyframing, see *Keyframing* on page 8.

To make a wave get increasingly larger, follow these steps:

1. Create a wave following the steps on page 119.
2. Position the time line pointer at frame 1.
3. Select the wave icon, and scale it to twice its original size. Choose the **SaveKey → Wave → Scaling → All** command.
4. Place the time line pointer at 50 and scale the wave again. Middle-click the **SaveKey** menu cell to save another keyframe of this type.

Continue to scale the wave as much as you dare (of course, this has an effect on the other objects in the scene).

Editing the Wave

The **Effects → Wave → Edit** command allows you to edit the existing parameters of a wave. You can also edit the wave's function curves using the **FcrvSelect → Wave → Decay** and **FcrvSelect → Wave → Spread** commands, as described on page 668 of the *Reference Guide*.

1. Select the wave icon to be edited.
2. Choose the **Effects → Wave → Edit** command.

The Define Wave dialog box is displayed as is shown for the **Effects → Wave → Create** command.

3. Select the type of wave desired and set the parameters as desired. Refer to the **Effects → Wave → Create** command on page 520 of the *Reference Guide* for a description of the wave parameters.
4. Click Ok to accept the settings or Cancel to exit the dialog box.

All parameters except the Wave Type may be edited. If the wave's start and stop frames are changed, then the decay and spread function curves are resized and offset to match the new values. Similarly, if the Wave Periodicity In Time option is selected, the extrapolation type of the decay and spread function curves is changed to cyclic (constant).

Changing the Profile Curve

The **Effects → Wave → Change Profile** command allows you to redefine the profile curve of an existing wave. This, of course, changes the shape and nature of the wave's animation.

1. Create and select a new profile curve either by editing an existing curve, or by creating a new curve.
2. Choose the **Effects → Wave → Change Profile** command.
3. Pick the wave icon.

The wave icon flashes, showing that the profile curve has been changed.

Freezing a Wave

The **Effects → Wave → Freeze** command allows you to freeze the effect of the wave on an object. By freezing the wave, you can use it as a powerful modelling tool for deforming an object.

1. Select the wave icon you want to freeze.
2. Choose the **Effects → Wave → Freeze** command.

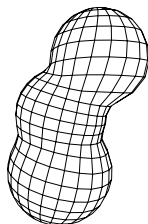
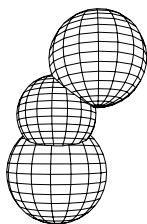
SOFTIMAGE 3D freezes the object's points in their current deformation and detaches the wave from the object. The wave icon remains on the screen and can be reattached to the same object or a different object.

Freezing is only implemented for node waves, not for branch waves. The **Effects → Wave → Freeze** command does not freeze the object's transformation, just its current shape (geometry). If there are several waves attached to an object, or if there are other effects such as curve or lattice deformations on the object, these effects are also frozen. This can lead to a doubling of the effects remaining on the object. Removing, or detaching the other waves returns the object to its normal "un-doubled" state, since if more than one wave is still attached to the frozen model, they amplify their effects on this new object.

Deforming Using Custom Effects

There are a number of custom effects in SOFTIMAGE 3D that let you deform various objects using different methods. For more information on all of these effects, see their command descriptions in the *Reference Guide*.

Conforming an Object to a Surface

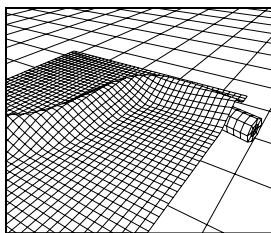


The **Effects** → **AnimatedShrinkWrap** command in the Motion module allows you to animate how the shape of an object (curve, polygon mesh, patch, or NURBS surface) conforms to the surface of a polygon mesh or patch object. You can control the direction of the shrinking, as well as the amount of shrinkage. You can apply the effect to the entire model, or only to tagged points.

This effect is typically used to:

- Create a metamorphosis between two dissimilar objects by shrink-wrapping another object (such as a large patch or NURBS sphere or a subdivided cube) to the two objects and then applying the shape animation on the results.
- Project a curve path onto a surface, such as an uneven terrain or a model of a character. This lets you easily create paths that are guaranteed to lie on the surface.
- Create seamless models. A large patch or NURBS surface can be shrunk to a model composed of several objects, such as a head modelled from several scaled and deformed spheres. This creates a simpler model consisting of a single patch or NURBS surface without seams.

Deforming by Points

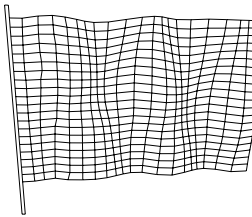
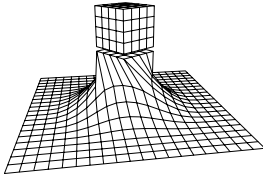
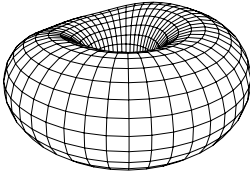


The **Effects** → **Magnet** command in the Motion module allows you to deform the points of one model by using another model as a magnet. You can control which points of the object are moved, how much they are moved, and in which direction. All elements, including the shape and location of the magnet and the magnitude and extent of the force, can be animated. By freezing the magnetic effect, you can use Magnet as a modelling tool.

These are typical effects that can be created using Magnet:

- An object moving under a rug forcing the rug to deform according to its shape.

- A footprint can be easily impressed into a grid or other surface. The edges are automatically softened by the magnetic effect on nearby points.
- A muscle that is animated with an inverse kinematic chain may be made to bulge realistically by adding magnets to the chain's envelope.



The **Effects** → **AnimatedPush** command in the Motion module allows you to animate the movement of points on a model in the direction of the model's surface normal. You can specify how far to move the points and the software calculates the rest using the vertex normals in the case of polygon meshes, and computing the surface normals in the case of patches, NURBS surfaces, and curves.

The **Effects** → **Glue** command in the Motion module allows you to “glue” tagged points of one object to the tagged points of another “surface” object. You can apply the Glue effect to polygon meshes, patches, NURBS surfaces, or curves. Either of the models can be animated, and the glued portions will stick together. Falloff parameters enable the effect to be applied smoothly to nearby points.

Locking Points to a Location

The **Effects** → **Lockdown** command in the Motion module allows you to lock points to their original locations regardless of any additional effects applied to the model. This applies to both SOFTIMAGE custom effects and to standard SOFTIMAGE 3D deformation effects such as **Wave** or **Lattice**. At each frame in an animation, the tagged points are returned to their original locations. Dropoff and damping parameters allow the effect to be applied smoothly to neighbouring points.

These are some effects that can be easily produced with Lockdown:

- A flag created by applying a wave to a grid, then held to a flag pole by applying Lockdown to the points of the grid next to the pole.
- A part of a character undergoing deformation; for example, a nose can be held to the rest of the face by applying Lockdown to the points of the nose nearest the face.
- A vibrating drum top that is held to the frame by applying Lockdown to the outer ring of points of the top. Similarly, a vibrating guitar string can be held to the guitar by applying Lockdown to the end points of the string.

C H A P T E R E I G H T

Kinematics

Introduction

The human skeleton is made up of torso, limbs, and articulation points, each of which can move independently of each other to some extent, and also together. Similarly, SOFTIMAGE 3D skeleton hierarchies can include any number of articulated chains which can act independently or in unison.

You can animate articulated chains using conventional keyframing of translations, rotations, and scaling, or by constraining it to another animated object that you translate, rotate, or scale. You could also animate it by applying dynamics simulations or by using animation data from a motion capture session.

How parts of the skeleton move relative to each other is determined by the way your skeleton hierarchy is built, whether and how objects are constrained to each other, and if you have created any expressions to govern their motion.

Inverse Kinematics versus Forward Kinematics

You can use either inverse or forward kinematics to animate your skeleton.

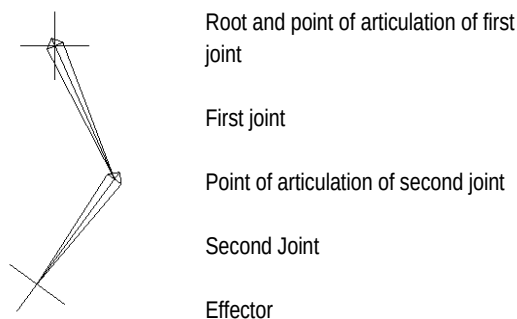
Inverse kinematics provides a quick and easy way to create animations: you define the goal of the action by positioning the end effector of an articulated chain (see next page) and SOFTIMAGE 3D calculates to how to position the rest of the chain to reach that goal.

When you use forward kinematics, you move the skeleton into position, specifying the angle of each joint and then keyframing these positions. Each movement is carefully planned to create the resulting animation. Using forward kinematics, you have complete control over the orientation of each joint, allowing you to create many types of movements that may not be possible to animate with inverse kinematics alone.

For more information on creating skeletons and modelling with them, see *Skeletons* on page 225 in the *Modelling User's Guide*. This chapter discusses the animation of skeletons.

Anatomy of a Skeleton

A skeleton is a hierarchy of articulated chains. In turn, an articulated chain is a hierarchy made up of a chain root, a given number of joints, and an end effector.



Chain Root

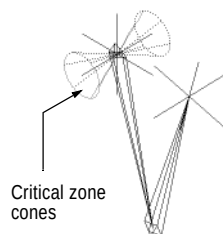
The chain root is the first point of a chain. Its position determines the orientation of the chain, and often contains data associated with the whole chain. It is displayed as a null model that can be selected.

In skeletons with multiple chains, a chain root can be the child of another chain or object.

Critical Zone

There is a zone around the chain root in which the general orientation of a chain will flip if the effector is translated into the zone. It is called the *critical zone* and it is represented by a double-cone shaped icon located at the point of articulation of the first joint of a chain. The chain will not flip in the plane perpendicular to the preferred axis of rotation.

To visualize it, choose the **Skeleton → Show Critical Zone** command in the Actor module.



Joint

A joint is a link in the articulated chain. It is composed of a point of articulation and a stretched octahedron. Joints are connected end to end with the local x-axis pointing towards the next joint.

Joints can be scaled (to change the length of an actor's leg, for example), and the first joint has three translational degrees of freedom (x, y, and z) since the chain root can be moved anywhere in space. The default node name for a joint is `jnt1_1` or `jntn_n` as you can see in the Schematic window or by using the **Info** ➔ **Selection** command.

Effector

The effector is the last point of a chain. Inverse kinematics is invoked when you move the effector. In the display windows, the effector is displayed as a null model that can be selected. In skeletons with multiple chains, the effector of one chain can be defined as the parent another chain.

2D Chains and 3D Chains

You can create either a 2D chain or a 3D chain. Both 2D and 3D chains can be rotated about three local axes (like a ball joint) at the point of articulation when using forward kinematics and dynamic simulations. However, when you invoke inverse kinematics, 2D and 3D chains behave differently.

In 2D chains, only the chain root and first joint rotate around three axes, like a ball joint. All other joints are constrained to stay on a single plane which is defined by the x and y-axes of the first joint of the chain. These joints in the chain can rotate about the local z-axis *only*, like a hinge joint rather than a ball joint. As a result, all joints of a 2D chain other than the first one, can only rotate around their local z-axis.

You can change the plane for these joints when you are animating your skeleton – see *Controlling the Orientation of a Chain* on page 137.

All joints of a 3D chains work like ball joints. They can be rotated about three axes and can move around in any plane. As a result, 3D chains are more flexible than 2D chains. You can use 3D chains to create models that are not restricted to a plane like 2D chains. For example, you can use 3D chains to create a swaying cat's tail, or seaweed, which makes use of the random movement.

Using Single Joint Chains

You can also use single joint chains by constraining them to other objects in your scene. This allows you to benefit from the control the chain provides, such as constraining only an effector to point in a particular direction. For example, you could use single joint chains for creating eye movement. To move an eye in the direction of the object upon which it is focused, constrain the single joint chain to the eye's pupil and then constrain the chain to the object at which the eye is looking.

You can also use single joint chains to represent objects that cannot be scaled, such as the camera. For example, in a very large scene, you may not be able to view your camera when you zoom out. To work around this problem, constrain a single joint chain to the camera, with the effector pointing in the direction of the camera interest. You can scale the chain to see where your camera is located in the scene, and in which direction it is pointing.

Creating Chains

Articulated chains are created by choosing the **Skeleton → Draw → 2D Chain** or **3D Chain** command in the Actor module and then entering a series of points in the window. The first point entered defines the chain root, and the last point entered defines the effector.

For more information about modelling skeletons, see *Drawing Articulated Chains* on page 231 of the *Modeling User's Guide*.

Animating with Inverse Kinematics

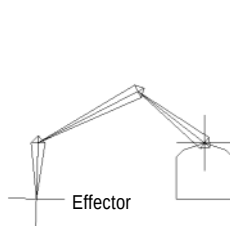
Inverse kinematics provides a quick and easy way to create animations: you define the goal of the action by positioning the end effector of an articulated chain and SOFTIMAGE 3D calculates to how to position the rest of the chain to reach that goal.

There are two basic ways of keyframing the translation: by explicit translation or by path animation. When you animate the explicit translation of chains, three function curves (x, y, and z) are generated. Using path animation allows you to translate chains by linking them to a path or trajectory.

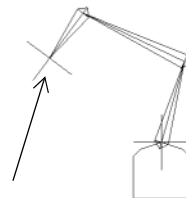
Explicit Translation

This is the most basic way of animating chains with inverse kinematics:

1. Create a chain by choosing the **Skeleton → Draw 2D Chain** or **3D Chain** command in the Actor module.
2. Select the end effector and choose **SaveKey → Object → Explicit Translation → All** in the Motion module to record a keyframe.
3. Invoke inverse kinematics by moving the effector to another position and choose the **SaveKey → Object → Explicit Translation → All** command again.



Frame 1
Save position of effector.



Frame 10
Move effector and save new position.



To translate the entire skeleton, select the chain root as a tree and move it to its new position.

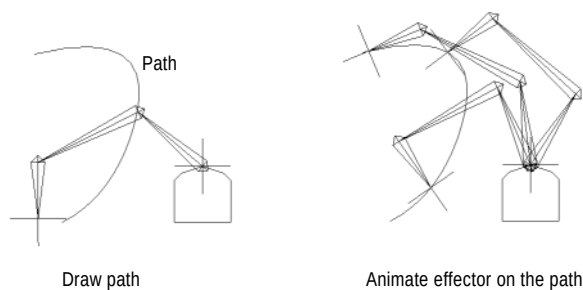
Simple instructions that show you how to create a walking actor using this technique are described in the Lesson 8, *Walk* in the *Quick Start* guide that came with your SOFTIMAGE 3D software.

Path Translation

You can also translate chains by linking them to a path or a trajectory following the steps described below:

1. Create a chain by choosing the **Skeleton** → **Draw 2D Chain** or **3D Chain** command in the Actor module.
2. Create a curve of any type using one of the **Draw** → **Curve** commands in the Model module.
3. Select the joint of the chain to be linked to the path.
4. Choose **Path** → **Pick Path** in the Motion module and pick the curve as the path.

The selected joint is linked to the path. When the sequence is played back, the joint translates along the path just like any other object. Since the joint is part of a chain, however, the principles of inverse kinematics are still applied.



For information on path animation, see *Animating on a Path* on page 67.

Constraining Joints and Effectors

There are a number of ways you can constrain joints and effectors. Two that are especially useful with articulated chains are the **Up Vector** and/or **Preferred Axis** constraints. You can use these constraints to control the general orientation of a chain (or its resolution plane). See *Controlling the Orientation of a Chain* on page 137 for more information or refer to the constraints found in the **Skeleton** command, starting on page 1306 of the *Reference Guide*.

You can also limit the movements of joints by using the **Rotation Limit** constraint (damping, however, does not apply when invoking inverse kinematics). Joints can also be constrained using the **Position**, **Scaling**, or **Position Limit** constraints, but not in direction or tangency. For more information, see *Constraints* on page 167.

Constraining the Effector to a Null

When you keyframe using path translation, the chain is constrained to the location where it was created. When you keyframe using the following method, you can move the chain around.

When you animate effectors using explicit translation, the effectors “stick” to the global coordinates that you keyframe. If you want to move your entire skeleton in space, you must re-key the effectors.

You can avoid re-keying by creating a null, constraining an effector to that null and then making the null the child of the chain root. When you keyframe the explicit translation of the nulls, the nulls keep their relative position to their parent (root). This lets you move your entire hierarchy and its animation.

1. Create a chain by choosing the **Skeleton → Draw 2D Chain** or **3D Chain** command.
2. Create a null (**Get → Primitive → Null**).
3. Choose the **Parent** menu cell and make the null a child of the chain root.
4. Select the effector.
5. Constrain it to the null by choosing **Constraint → Position** in the **Actor** (or **Motion**) module and picking the null as the constraining object.

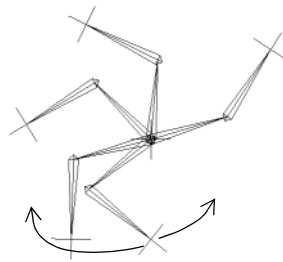
6. Select the null and choose **SaveKey → Object → Explicit Translation → All** in the Motion module to record a keyframe.
7. Move the null to another position and choose the **SaveKey → Object → Explicit Translation** command again.

This lets you move the entire hierarchy and its animation in space.

Controlling the Orientation of a Chain

In SOFTIMAGE 3D, 2D and 3D chains have a preferred axis of rotation around which the chain tries to rotate when you animate the translation of the effector. The default axis of rotation is the z-axis of the chain's root.

You can predict the behavior of an articulated chain because of this preferred axis of rotation. For example, if you draw a 2D chain in the Front window, when you translate its effector around its root (in the Front window) the chain moves as illustrated below:



With 2D chains, the preferred axis of a chain is perpendicular to the plane in which SOFTIMAGE 3D tries to keep the chain when moving the effector in space. This plane is referred to as the general orientation or resolution plane of a chain. The general orientation of a 3D chain is also influenced by its preferred rotation axis. However, the joints of a 3D chain do not necessarily lie on the same plane as the joints of a 2D chain.

The resolution plane of a 2D chain is the plane in which the entire chain lies. This plane corresponds to the local xy plane of the first joint (and all subsequent joints) of a 2D chain. The resolution plane is defined by the position of the chain root and the position of its effector. For more information about changing the resolution plane of a chain, see page 139.

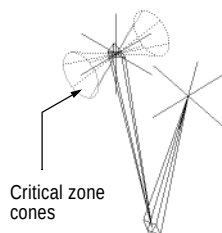
You can control the orientation of a chain to help you avoid problems with the chain flipping. This problem occurs when the effector of a 2D or 3D chain is translated near the preferred axis of rotation (z). As a result, the general orientation of the chain changes rapidly and dramatically (flips) when it crosses the preferred axis.

For more information about avoiding problems with flipping, read the following sections.

Showing the Critical Zones

You can use the **Skeleton → Show Critical Zone** command to avoid placing the effector of a chain within the region in which the flipping will occur. The chain will not flip in the plane perpendicular to the preferred axis of rotation.

The critical zone icon is a double-cone shaped icon located at the chain root. Each cone indicates the region in which the general orientation of a chain changes if the effector is translated into this area.

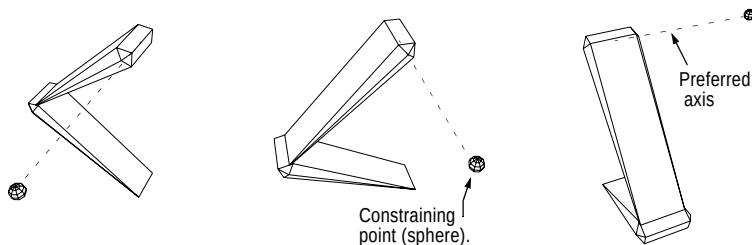


The critical zone is blue when the default preferred axis of rotation is used, or when a preferred axis constraint is applied to the chain. The critical zone is green when an up vector constraint is applied to the chain.

Constraining the Preferred Axis of a Chain

There are some constraints that allow you to control the preferred axis of the chain and its resolution plane.

When you use the preferred axis constraint, you can change the preferred axis of rotation to be x or y instead of the default (z). When you change the preferred axis of rotation of a chain, SOFTIMAGE 3D tries to rotate the chain around the new axis when you animate the movement of the effector.



Preferred axis constraint
Chain has a preferred axis constraint to the sphere.

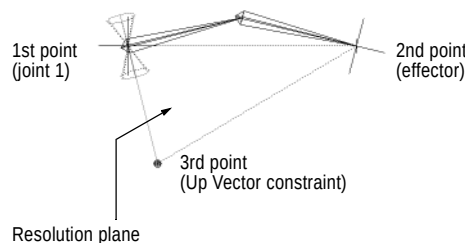
You can choose the **Skeleton → Automatic Preferred Cns** command to automatically create this constraint when you draw an articulated chain, or you can apply the constraint after you have created a chain by choosing the **Skeleton → Preferred Axis Cns** command.

For example, if you want to change the preferred axis of a chain from the global z-axis to move around the chain's local y-axis, you would do the following:

1. Draw a 2D chain of two joints in the Front window (use the **Skeleton → Draw → 2D Chain** command).
2. Choose **Get → Primitive → Null** to create a null and position it in the direction of the local y-axis of the chain's root.
3. Select the first joint of the chain and choose the **Skeleton → Preferred Axis Cns** command.
4. Select the null as the reference to constrain the 2D chain.

Constraining the Up Direction of the Chain

You can use the **Skeleton → Up Vector** command to constrain the orientation of a chain (you can also use the **Constraint → Up Vector** command to achieve the same results). This constraint provides full and precise control of the general orientation of your chain and helps you control where flipping problems will occur. The up vector constraint specifies the third point of the resolution plane of the chain, while the first two points are the positions of the first joint and the effector. The Up Vector constraint has priority over the Preferred Axis constraint when both are applied to the first joint of a chain.

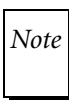
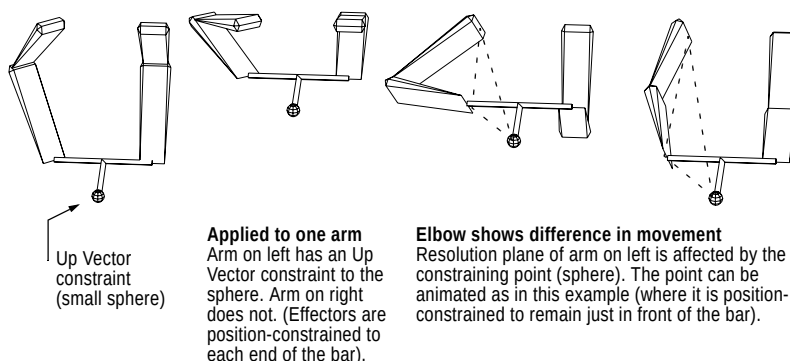


For example, if you animate a breaststroke swimmer, when its arms are extended, its elbows are positioned at the same height as its shoulders. You can apply an Up Vector constraint to its arm, located to the side of its shoulder, at the same height.

1. Draw a 2D chain of two joints in the Top window (use the **Skeleton → Draw → 2D Chain** command).
2. Choose **Get → Primitive → Null** to create a null and position it next to the shoulder.
3. Select the first joint of the chain and choose the **Skeleton → Up Vector Cns** command.
4. Select the null as the reference to constrain the 2D chain.

When the swimmer's hands are near to the chest, the elbows must be lowered, so you must also lower the Up Vector constraint.

You can also use the **Skeleton → Automatic Up Vector Cns** command to constrain the up direction of a chain. If this command is selected, a null is automatically created when you draw a chain, the first joint of the chain is constrained to the null, and the null becomes the child of the chain root.



In the Schematic window, you will see the Up Vector constraint as a “resplane” null.

Changing the Preferred Angle of a Chain's Joint

You can also change the preferred angle of a joint in an articulated chain. This affects the movement of the chain when you animate it.

When you draw an articulated chain, the preferred angles of a joint are the local rotation angles. These angles influence the behavior of the entire chain because when you animate the movement of the effector, SOFTIMAGE 3D tries to preserve the preferred angles as much as possible.

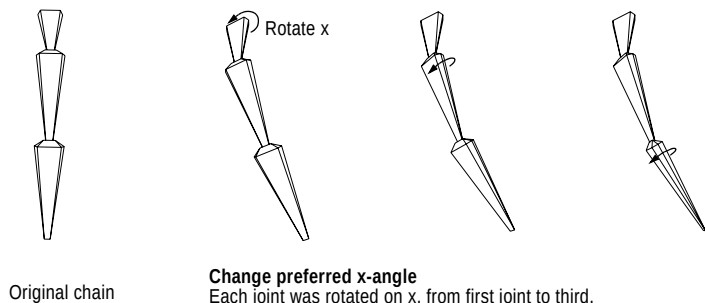
When using inverse kinematics, only the preferred z-angle of a joint is used (though the preferred x-angle has some meaning as described below). Each joint is constrained to remain on a single plane, and can rotate only around its local z-axis when the effector is moved.

When using 3D inverse kinematic chains, the preferred x, y, and z angles of the joints are used, since 3D joints work like ball joints.

For a 2D chain, the preferred x-angle of each joint has a special meaning. This does not apply to 3D chains because all angles (x, y, z) are already used to specify the orientation of the joint itself.

The preferred x-angle of the first joint of a 2D chain allows you to incline the resolution plane of the entire chain around its root-effector direction. When you change the preferred x-angle of the first joint of your chain, the chain is oriented (using the preferred axis or up vector constraint if you have applied one), then the inverse kinematics is resolved to reach the goal (translate the effector), and finally the resolution plane is inclined based on the preferred x-angle you have set. You can also add a rotation offset to the default resolution plane or to the resolution plane specified by the preferred rotation constraint or up vector constraint.

Once you have changed the preferred angle of the first joint, the rotations are cumulative when you change the preferred angle of the second and subsequent joints. This allows you to gradually twist an envelope, and you can animate this effect by rotating subsequent joints of a 2D chain.



For example, if you are keyframing the motion of an arm and you want to rotate the arm from the shoulder to the wrist, then you can change the preferred angle of the first joint. You can either change these angles by rotating the joint using the **Rotation** menu cells, or by entering new values in the Spreadsheet:

1. Open the Spreadsheet window and click **QUERIES** in the title bar.
2. Select the **Chains** query from the menu. Chain attributes appear in the Spreadsheet.
3. Middle-click on the preferred angle cell (protx, proty, or protz) corresponding to the joint you wish to change, and enter a new value. Press Enter to accept this value.

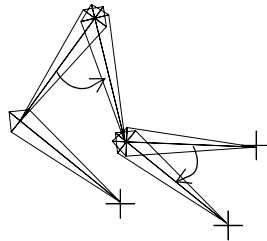
You can also change the preferred angles of joints using the **Skeleton** → **Move Joint** or **Skeleton** → **Move Branch** commands on page 1301 and page 1303 of the *Reference Guide*.

You can animate the preferred angles of a joint by saving rotation keyframes (**SaveKey** → **Object** → **Rotation** commands). When the animation is played back, the keyed rotation values replace the preferred angles of the joint before the inverse kinematics of the chain is resolved. This allows you to change the behavior of a chain as the animation progresses.

Animating with Forward Kinematics

When you animate articulated chains with forward kinematics, you don't animate the chain with inverse kinematics: instead, you animate the rotations of all joints. This affects the end effector's position, not the reverse.

1. Create a chain by choosing the **Skeleton → Draw 2D Chain** or **3D Chain** command.
2. Select the joints you want keyframe, and choose **SaveKey → Object → Rotation → All** to record a rotation keyframe.
3. To animate your object, move the time line pointer and continue to create rotation keys for the joint using **SaveKey → Object → Rotation → All** command.



Although you are not using inverse kinematics to its advantage, there may be other advantages of using this method:

- The exact orientation of the joint is keyed. This prevents any “surprises” that can occur when 2D chains flatten on their 2D plane.
- Three function curves are generated for each joint orientation, which gives you precise control over your animation.

C H A P T E R N I N E

Dynamics

Introduction

Dynamic simulations provide a quick and easy way to create realistic motion by simulating physical forces and assigning physical properties to objects. Using Dynamics, SOFTIMAGE 3D performs complex calculations to set transformation values to simulate the physical forces which act upon real objects. The advantage to dynamic simulations is that it allows elements in a scene to appear to react to their environment, creating complex animation effects with minimal input.

For example, using dynamics you can easily create an articulated chain swaying in the wind, a task that would be very time-consuming to perform using keyframing. Also, since dynamics simulate nature, you could attach your chain to a moving object, and the chain would react correctly to the object's movements, such as a kite's tail reacting to the movements of the kite as it is blown by the wind.

Dynamics is the ideal tool for creating secondary motion that enhances your animation. You can add subtle, realistic effects to your scene quickly and easily. For example, if you create a hopping rabbit, you could easily make its ears flop in response to its movements by assigning dynamics to the ears.

You can also set parameters related specifically to physical dynamics. Physical properties such as elasticity and roughness can be keyframed. You can set obstacles and use collision detection to create realistic impacts between objects and obstacles in your scene. To help reduce computation costs, you can define your objects and obstacles as bounding planes, boxes, or spheres.

When activated, dynamics automatically generates raw data keys, creating rotation and position function curves for your object. You can combine dynamics simulations with traditional keyframing. In other words, dynamic animation can use keyframed values as starting values – the simulated values are appended to the keyframed animation data. For example, you can keyframe a model rocket flying up from a launching pad, then combine that animation with the simulation of the rocket being pulled upon by gravity, and falling back to earth.

For an example using dynamics, see *The Bouncing Ball* on page 163.

Animating with Dynamics

Dynamic simulation is used to add realistic motion to your animations, and can easily be combined with standard keyframe animation and inverse kinematics.

When dynamics is used, a *transition curve* is generated. The transition curve shows when an object is animated using standard animation methods (such as keyframing) and when dynamics is applied; it ensures seamless motion continuity when switching from standard methods to dynamics. The transition curve shows on and off states only and resembles a square wave. Also, the simulated rotation and translation values generated when dynamics are incorporated into the object's function curves are raw data keys (a key for every frame) that can be edited.

Overview of Creating a Dynamic Simulation

To use dynamics, follow these basic steps:

1. You must first assign density to an object before you can obtain a dynamic simulation. Choose the **Dynamics → Physical Properties** command in the Actor module. A gravity icon is automatically created.
2. You can then assign other physical properties such as elasticity and roughness to modify the behavior of your dynamic simulation. This is also done using the **Dynamics → Physical Properties** command. For more information, see *Assigning Physical Properties* on page 150.

You can also set parameters using the **Dynamics → Option Setup** command (page 153). This allows you to define how your chain or object will respond to dynamic forces and to collisions.

3. After physical properties are added to your object, it can be influenced by *physical forces* such as gravity, force, wind, fans, or nails using the **Control → Get Special Control** command. For more information, see *Setting Up Dynamics Properties* on page 153.

4. A *dynamic simulation* is achieved by manipulating physical forces that produce motion in objects over time. To create the simulation, choose the **Simulate ➔ Start/End** command. You then choose the **Simulate ➔ Simulate (automatic)** command to play the simulation.

To specify when a simulation should start and finish, you must select the start and end frame. For more information, see *Creating Dynamic Simulations* on page 158.

5. If you like, create a *collision* with an obstacle as part of the dynamic simulation using the **Dynamics ➔ Option Setup** and **Dynamics ➔ Physical Properties** commands. For more information, see *Creating Obstacles for Collisions* on page 159.

Assigning Physical Properties

Assigning Density to an Object

To simulate an object's motion using dynamics, you must first assign density to the object (objects include articulated chains) using the **Dynamics → Physical Properties** command in the Actor module (see page 394 of the *Reference Guide*).

Assigning density lets an object's mass be computed by SOFTIMAGE 3D. *Mass* is obtained by multiplying the density you specify by the volume of the object. Depending on the mass you have specified for your object, it responds differently to physical forces.

1. Select an object.
2. Choose the **Dynamics → Physical Properties** command and use the Density slider to set the density of the object. The density can be any positive value above zero.
3. Click Ok All to accept the settings for all objects to which this is applied, such as in a hierarchy.

To try a dynamics tutorial, see *The Bouncing Ball* on page 163.

Assigning Other Physical Properties

After assigning density, you can also assign other physical properties such as elasticity, or static and kinetic roughness, which also uses the **Dynamics → Physical Properties** command.

- *Elasticity* is the amount of energy lost from one object when it collides with another object. For example, when a ball hits the floor, elasticity influences how high the ball bounces.
- *Static roughness* is the friction factor that determines how a static object slides on the surface of another. A rolling ball “adheres” to the ground more with high static roughness, and slides more with low static roughness.
- *Kinetic roughness* is the friction factor that determines the energy that is lost by an object as it moves on the surface of another. To use the rolling ball example again, the ball travels far with low kinetic roughness and a short distance with high kinetic roughness.

Physical properties can be assigned to any number of nodes in a model hierarchy. However, properties must be assigned to the root node, and no gaps are permitted between the root and the last node

with properties. For example: a hierarchy ($a \rightarrow b \rightarrow c$, where a is the root) has properties in a , and can have properties in $a + b$ and $a + b + c$, but not in $a + c$. Note that nulls and curves are not considered as part of the hierarchy.

Assigning Friction to Joints

Physical properties assigned to an articulated chain affect all of its joints. You can, however, assign different values to the physical properties of each joint. After assigning basic physical properties, you can further modify the effects of a dynamic simulation by changing the joint properties.

The **Dynamics → Joint Properties** command in the Actor module allows you to assign friction to each joint in an articulated chain on all three axes.

Joint properties are also present in the root of a hierarchy of models where an automatically generated virtual joint exists between the root model's centre and the overall centre of mass.

Note

To use dynamics, your chain or object must have physical properties assigned to it. Select the chain root or hierarchy root (for models) of an object.

1. Select the chain or joint with the middle mouse button.
2. Choose **Dynamics → Physical Properties** to set the physical properties of the object and click **Ok All**.
3. Choose the **Dynamics → Joint Properties** command. The Joint Physical Properties dialogue box is displayed.
4. Set the **Friction** parameters as desired for each axis. A higher value increases the joint's stiffness and a lower value "greases" the joint. If you set the **Friction** to 100, it locks the joint in place so that it cannot be moved.
5. Click **Ok** or **Ok All** to assign the new values to one or all joints in the chain.

Changing the joint's properties can be used only with dynamic simulations: you cannot change the friction of joints when you are only using inverse kinematics or keyframing.

Removing Physical Properties

You can remove all physical properties, including option setup and joint properties, that are assigned to an object. Select the object and choose the **Dynamics ➤ Remove Properties** command in the Actor module. The associated transition curve, if present, is also deleted; however, you are prompted to keep or delete the simulated values.

Setting Up Dynamics Properties

The **Dynamics → Option Setup** command in the Actor module displays a dialogue box that contains general parameters that allow you to define how your chain or object will respond to dynamic forces and to collisions.

You can set the root constraints, dynamic constraints, and dynamic geometry of an object. For more information about modifying the parameters of an object for a dynamic simulation, see page 389 of the *Reference Guide*.

1. Create an object and assign its density by choosing the **Dynamics → Physical Properties** command.
2. Choose the **Dynamics → Option Setup** command. The Option Simulation Setup dialogue box is displayed.
3. Set the options such as the Root constraints (position and orientation), Dynamics constraints (such as gravity, collisions, and wind), and Dynamic geometry.

If you are using a chain, the Root constraint options affect how the chain's root responds to the dynamic simulation.

4. Click **Ok All** to accept the physical properties for all selected elements, or click **Ok** to accept physical properties for one element at a time.

Adding Physical Forces

Once you have assigned physical properties to any chain or object, SOFTIMAGE 3D automatically invokes the default gravity. This gravity applies to any other objects with physical properties that are added to your scene. The default gravity is shown as a vector icon with a G, which is like other physical forces you can define. Choose the **Show → Controls** command if the dynamics icons are not visible in your scene.

These other forces you can use to modify your simulation are accessed from the **Control** menu in the Actor module and include: user-defined gravity vectors, force vectors, wind vectors, fans, and nails.

Physical forces are models that are displayed as icons. Like other models, their values can be modified, they can be animated, and they can be included in hierarchies to allow you to achieve the specific effects you want.

You can add the following physical forces:

- **Gravity** controls the acceleration at which an object falls or moves in the direction of the gravity vector.
- **Force** applies force to a joint (for example, a constant push or an instantaneous burst from an explosion).
- **Wind** simulates the effects of natural wind.
- **Fan** applies wind locally.
- **Nail** constrains a joint to a position. It is used like a regular positional constraint, but works like a magnet.

The **Control → Flash Controls** command allows you to view which controls are assigned to a specific object.

You can remove physical controls that are assigned to a specific object by choosing the **Control → Relax** command.

Assigning a Nail Control to a Joint

In this example, you can create a nail to constraint a chain to a certain point.

1. Draw a 2D articulated chain (**Skeleton → Draw 2D Chain**).

2. Apply physical properties to the entire hierarchy by selecting the chain, choosing **Dynamics** → **Physical Properties**, and clicking **Ok All**.
3. You can choose **Dynamics** → **Joint Properties** to set the friction of each joint.
4. Choose **Control** → **Get Special Control** → **Nail**. A nail icon appears.
5. Select a joint (with left mouse), choose **Control** → **Select**, and pick the nail icon. The nail icon flashes red.

The joint is now constrained to the nail icon.

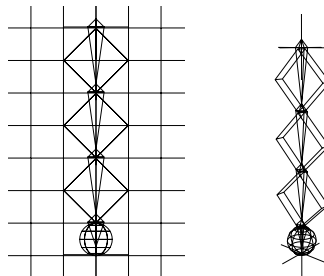
You can adjust the attraction factor of the nail. A very small amount of scaling in x (0.05) reduces the attraction factor just enough to make the joint react to it. If you use an x-scale of 0.2, it has enough attraction to completely pull the joint toward it.

See the **Control** menu cell commands starting on page 223 of the *Reference Guide* for more information about controls.

Creating a Wind Chime

This example incorporates skeletons, special controls, and simulation. To create a wind chime using chains and primitive objects:

1. Choose **Skeleton** → **Draw 2D Chain** in the Actor module and create a chain of four joints in the Front window (see the illustration).
2. Create a cube. Scale, translate, and rotate the cube to position it over the first joint of the chain as shown in the illustration.
3. Duplicate the cube twice and move each copy over a joint.
4. Create a sphere and move it over the fourth (bottom) joint.



5. Make every joint the parent of their respective object (cube or sphere).
 6. Select the chain with the middle mouse button and apply a default gravity factor by choosing **Dynamics → Physical Properties**. Accept the default values by clicking the Ok All button.
 7. Choose **Control → Get Special Control → Fan** and a fan icon appears in the centre of the windows.
 8. Move the fan to be in front of and to the left of the wind chime. Rotate the fan to face the wind chime.
 9. Use the **ScaleXYZ** menu cells to define the strength of the wind and the area and distance it affects.
 - x scaling defines the length of the decay vector.
 - y scaling defines the amount of air current generated by the fan.
 - z scaling defines the radius of influence of the cone.
 10. Select the entire wind chime hierarchy using the right mouse button. Choose **Control → Select** and pick the fan to assign it as a special control.
 11. Move the time line pointer to frame 1.
 12. Choose **Simulate → Save Start/End** to initialize the simulation. Choose **Simulate → Simulate (automatic)** to create the animation.
 13. Play back the animation.
- To get a more dynamic animation, manipulate the scaling values along each axis.
14. Animate the fan by rotating it on the y-axis and setting keyframes. Make the animation of the fan a cycle. When a simulation is created, a normal function curve for the rotation is automatically created and can be edited at any time.



To create a cycle motion of the fan, create three keyframes of rotation on the y-axis. The first and last keyframe should have the same values. Select the fan's y-rotation function curve in the Fcurve window and choose **FcrvEdit → Extrapolate mode → Cycle**. The function curve should now be cyclical for the entire length of the animation.

15. Before creating a new simulation, you must first delete the original one. To do so, choose **Simulate → Delete Transition Curve** and then **Simulate → Delete Simulation**.
16. Choose **Simulate → Save Start/End** to initialize the simulation again and then **Simulate → Simulate (automatic)** to create the new animation.
17. Play back the animation.

Creating Dynamic Simulations

Once basic physical properties have been assigned to an object, you can create a simulation. A dynamic simulation is achieved by manipulating physical forces that produce motion in objects over time.

1. To define when you want your simulation to start and end, you must select a start and end frame. Choose the **Simulate → Start/End Frame** command to define the length of your simulation.
2. Select the **Simulate → Simulate (automatic)** command to start the simulation.
3. Play back the simulation from the first frame.

For more information, see the **Simulate** commands starting on page 1287 of the *Reference Guide*.

During a simulation, time is sampled at a higher rate than frames which provides precise results. The result values, however, are recorded at each frame, which lets you play back your scene with the actual simulated values for each frame.

Because dynamics simulation imitates physical laws by performing intensive calculations, there are certain limitations. Infinite and very small values, such as an object with hardly any mass, should not be used because the calculations required may introduce numerical errors into the simulation.

Each SOFTIMAGE 3D grid unit is assigned a default value of one (cubic) centimeter. The default gravity is set to the gravity of earth, so your object falls the same distance in the same time on your screen as it would on earth.

Creating Obstacles for Collisions

You can also incorporate collisions in your dynamic simulations by using the **Collision → Select Obstacle** commands (Bsphere, Bplane, or Bbox) in the Actor module. For example, if you want to animate a ball bouncing down a flight of stairs, select the ball, and then select the stairs as an obstacle. Your ball then detects the stairs as an obstacle and responds to the impact appropriately.

You can modify your collision simulations and reduce computation time by using collision commands and by modifying elasticity and roughness parameters in the **Dynamics → Option Setup** and **Dynamics → Physical Properties** command dialogue boxes.

SOFTIMAGE 3D allows you to define models that collide with the obstacle objects as bounding boxes, spheres, or their actual shape. If you select an alternate shape, SOFTIMAGE 3D uses the simpler geometry of the bounding plane, sphere, or box to perform the simulation calculations. Since a dynamics simulation continuously calculates the position of an object while checking for collisions, using a bounding box shape saves time while maintaining a high degree of precision. For example, if you want to simulate marbles being poured onto a table, you could define the marbles as bounding spheres and the table as a bounding box.

Although obstacles are at the receiving end of a collision, the obstacle itself is not affected by a collision. For example, if you drop marbles onto a table, the table's geometry does not change, or a ball that hits a cube does not change the position of the cube.

To create a collision:

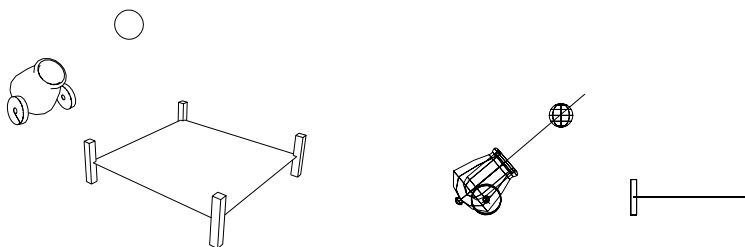
1. Create an object and assign physical properties to it. Animate it as you like.
2. Select the object and choose the **Dynamics → Option Setup** command and activate Collision detection for that object.
3. Choose one of the **Collisions → Select () Obstacle** commands to tell the object which obstacle to consider. Pick this obstacle object: it flashes red.
4. Choose the **Simulate → Simulate (automatic)** command. The animation plays back.

The **Collisions → Flash Obstacles** command allows you to view which obstacles are assigned to a specific object.

To remove the constraint between an object and its obstacle, choose the **Collisions** → **Relax** command.

Creating a Dynamic Simulation with a Collision

You can create a cannon and trampoline effect by using dynamic simulations.



1. Create a cannon (cylinder), trampoline (grid), floor (a larger grid), and cannon ball (sphere) using primitive objects.
2. Position the cannon away from the trampoline on the x-axis as shown in the illustration.
3. With the sphere at the bottom of the cannon, set a keyframe at frame 0 for the cannon ball using **SaveKey** → **Object** → **Translation** in the Motion module.
4. Move the time line pointer to frame 10, and translate the sphere about 20 units from the mouth of the cannon along the trajectory. Set a second keyframe.
5. Choose **FcrvSelect** → **Object** → **Translation** to see the function curve of this motion in the Fcurve window.
6. To remove the deceleration at the top of the function curve, click **EDITKEY** in the window title bar. Click and drag the last key point's handle down to increase the cannon ball's acceleration over time.
7. Choose **Dynamics** → **Physical Properties**. Select **Bounding Sphere** as the **Collision** type and click **Ok All** to accept these properties.
8. Move the time line pointer to frame 7 and save a starting frame for the simulation by choosing **Simulate** → **Save Start/End**.

Note

The simulation and the keyframed motion are overlapped by a couple of frames to ensure continuity of motion.

9. Create the first simulation by choosing **Simulate → Simulate (automatic)**. Play back the animation.
10. To adjust the trajectory of the cannon ball, open a Schematic window and select the gravity vector. In the Front window, scale the gravity vector down to increase the cannon ball's trajectory or scale it up to decrease the cannon ball's trajectory. Run the simulation again. Repeat this process until the sphere hits the trampoline.

Note

Each time you want to create a new simulation, you must first delete the previous simulation by choosing **Simulate → Delete Simulation**.

11. Assign the trampoline as an obstacle for the cannon ball. To do this, select the cannon ball, choose **Collisions → Select Bbox Obstacle** and pick the trampoline as the obstacle.
12. Assign the floor as an obstacle, but this time make it a Bplane.
13. Create the simulation again.

Note

A Bbox obstacle's influence is limited to the actual size of the object to which it is assigned – in this case, the trampoline. A Bplane obstacle's influence, on the other hand, is not limited to the object's size but continues in the specified plane to infinity.

14. Select the sphere and choose **FcrvSelect → Dynamics → Transition Curve** in the Motion module. As you can see, this function curve varies between 0 and 1, with 0 representing keyframed animation and 1 the simulated animation, respectively.

Creating Real Time Collisions

The **Channel → Shock** and **Channel → Shock3D** custom effect commands allow you to simulate real-time dynamic collision between models. Shock supports the collision between the bounding spheres of single objects on a plane using both the **Custom** and **Channel** menu cell commands to exchange data with SOFTIMAGE 3D.

Shock3D supports the collision between the bounding sphere and/or bounding box of single objects in 3D space using both the **Custom** and **Channel** menu cell commands to exchange data with SOFTIMAGE 3D.

For more information on using these effects, see their command descriptions in the *Reference Guide*.

The Bouncing Ball

This is a basic tutorial that shows you how to use all elements of dynamics together for the classic bouncing ball scenario.

Throw a Ball into the Air and Let it Fall

Set the frame range on the time line from 1 to 60. Make sure you are currently at frame 1.

1. Get a grid of any size using the **Get → Primitive → Grid** command.
2. Get a sphere with a radius of 2.0 using the **Get → Primitive → Sphere** command, and set the translation values as follows:
 - TransX -18.0
 - TransY 9.0

You may want to zoom back to view the entire scene.

3. Choose the **SaveKey → Object → Translation** command in the Motion module to set the first keyframe.
4. Go to frame 9 and translate the sphere to its release position:
 - TransX -10.0
 - TransY 19.0
5. Choose the **SaveKey → Object → Translation** command (or simply middle-click **SaveKey** to repeat its last used command).
6. Play back the motion.

The ball follows the path from frame 1 to 9.

7. Choose the **Dynamics → Physical Properties** command to assign physical properties to the sphere and click **Ok All** to accept the default values.

If you deselect the Grid option in the window's Layout dialogue box and orbit the camera, you can see the centre of the mass icon (a small cross and a small box) in the centre of the sphere. The default gravity icon is also displayed.

8. Set the current frame to 8 and make sure that the ball is selected.
9. Choose the **Simulate → Save Start/End** command to set the start frame for the simulation.

The sphere flashes to show that the start frame has been accepted, and a transition curve is created. Note that the simulation overlaps (by one frame) the simulation start and end of the keyframed motion to the continuity of motion.

10. Choose the **Simulate** → **Simulate (automatic)** command to start the simulation. Click the right mouse button to end or interrupt the simulation.

The ball starts with the velocity obtained from the keyframed motion and falls under the effect of the default gravity. It falls through the floor because it has not been set to detect collisions with obstacles.

Incorporate a Collision

11. Select the ball and choose the **Dynamics** → **Option Setup**.
12. In the Option Simulation Setup dialogue box that is displayed, select the **Collision Detection** and **Bounding Sphere** options to activate collision detection for the ball.
13. Choose the **Collisions** → **Select Bplane Obstacle** command to tell the ball which obstacle to consider. Pick the grid with the left mouse button. The grid flashes red.
14. Choose the **Simulate** → **Simulate (automatic)** command.

The animation plays back. The ball collides with the grid (floor) and bounces.

Modify Physical Properties to Vary the Motion

15. Select the ball and choose the **Dynamics** → **Physical Properties** command. Reset the **Elasticity** parameter to 1.
16. Choose the **Simulate** → **Simulate (automatic)** command.



You do not have to reposition the time line pointer because the simulation start is determined by the transition curve values. Your ball should now be very bouncy!

17. Choose the **Dynamics** → **Physical Properties** command.
 - Change the **Elasticity** parameter to 0.2.
 - Set the **Roughness (static)** parameter to 0.
 - Set the **Roughness (kinetic)** parameter to 0.

18. Choose the **Simulate** ➔ **Simulate (automatic)** command again.

Your ball barely bounces and skids along the floor without losing any speed.

19. Choose the **Dynamics** ➔ **Physical Properties** command again and change the **Roughness (kinetic)** parameter to 1.

20. Choose the **Simulate** ➔ **Simulate (automatic)** command.

The ball still slides, but slows down and quickly comes to a stop. Try setting the **Roughness (static)** value to 1 and see how it controls the way the ball rolls.

21. You can also improve your starting motion by modifying the acceleration. Choose the **FcrvSelect** ➔ **Object** ➔ **Translation** command in the Motion module. The Fcurve window opens.

22. Select EDITKEY from the Fcurve window title bar. Click on the second key's slope handle and move it down so that the velocity increases with time.

23. Choose the **Simulate** ➔ **Simulate (automatic)** command.

The simulation's initial speed is accelerated, and the ball travels further.

By decreasing the gravity amplitude (changing its y-scaling), you can also modify the motion so that your ball bounces like a beach ball.

C H A P T E R T E N

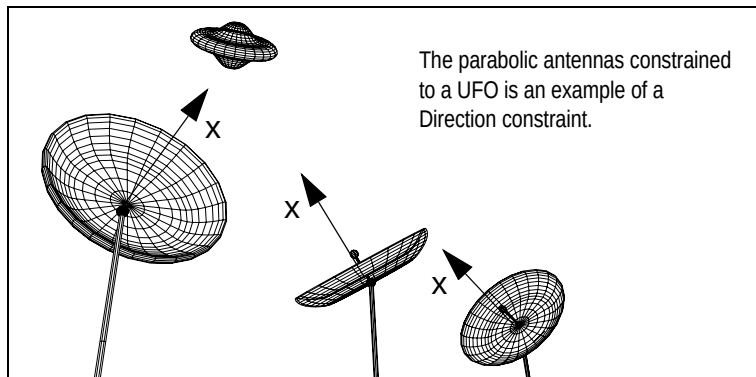
Constraints

Introduction

Constraints allow you to constrain objects, lights, and cameras to another object. This is very useful for creating complex behaviour by having an object react automatically to another's animation. You can even use the sun light to constrain another object's behaviour! You can also constrain one object to many objects, in which case the average position is computed.

For example, the **Constraint** → **Direction** command constrains the x-axis of one or more selected objects to another object, without changing the location of those objects. The x-axis of each constrained object always remains facing the centre of the constraining object, no matter where the constraining object is.

Suppose you want to animate antennas on satellites to follow a UFO. To do this, constrain each antenna to the UFO and then animate only the UFO. The x-axes of the antennas will always point toward the UFO.



If you would have used keyframing to animate the same scene, you would have to generate function curves for each of the three antennas as well as the UFO. You would also probably have to edit the orientation of each antenna to make sure it appeared to be tracking the UFO realistically. With the Direction constraint, the only object you have to keyframe is the UFO.

There are many different types of predefined constraints available for your use. These are defined as commands in the **Constraint** menu in the Motion and Actor modules. In addition to using these commands, you can create custom constraints using mathematical expressions. For more information, see *Creating a Custom Constraint* on page 203.

Using Multiple Constraints

It is possible to have more than one constraint applied to a particular object at the same time. When more than one constraint is active at the same time on an object, the average value of the constraints is used.

Building complex constraint relationships between elements can eventually create a cycle of dependencies. If this happens, a message appear in the status line telling you to relax (remove) the latest constraint you set. As a simple example, constraining **a** to **b** (such as with an Orientation constraint) and then trying to constrain **b** to **a** will create a cycle of dependencies.

Constraining Object Centres

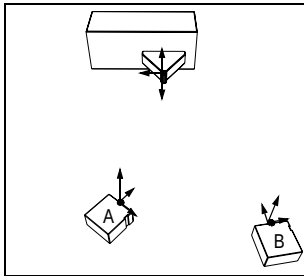
The **Constraint → Position** command constrains the centre of one or more selected objects to the centre of another object or to a null.

Note

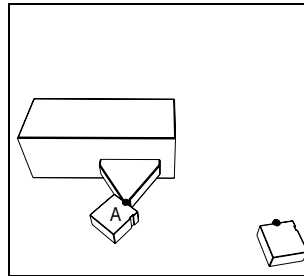
A **Position** constraint is the only constraint that can be applied to lights.

1. Select the objects to be constrained.
2. Choose the **Constraint → Position** command in the Actor or Motion module.
3. Pick the object that is to act as a constraint.

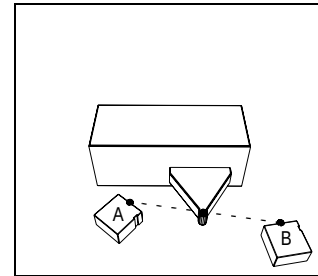
The constraining object flashes red. The constrained objects reposition their centres to the centre of the constraining object. When a transformation is performed on the constraining object, the constrained objects' centres remain aligned.



Position constraint (before)
Brick at top is object affected.
Centres are shown.



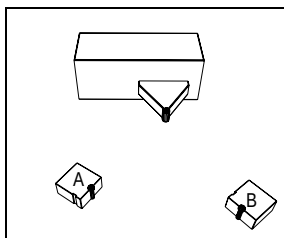
Constrain brick to block A
Brick is translated so that its centre is at same location as centre of brick A.



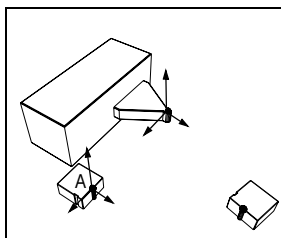
Constrain brick to both blocks A and B
Brick is translated to a point between A and B (averaged).

Constraining the Rotation

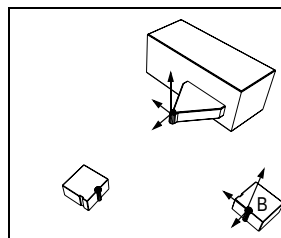
The **Constraint → Orientation** command constrains the rotation values of one or more selected objects to another object without changing the location of the constrained objects. A rotation performed on the constraining object is also performed on the constrained objects.



Orientation constraint (before)



Constrain brick to block A
Brick is rotated so that its centre is oriented to that of block A.



Constrain brick to block B
Brick is rotated to block B's orientation.

You can also constraint one object to many, which results in the average orientation being used.

For example, you can make a group of cheerleaders' heads all move in sync by animating one head and then constraining the orientation of all the others to this head.

1. Select the object or objects to be constrained.
2. Choose the **Constraint → Orientation** command.
3. Pick the object that is to act as a constraint.

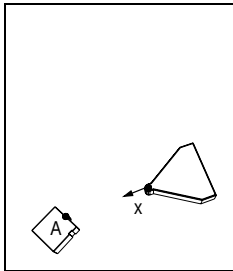
The constraining object flashes red.

4. Select the constraining object and rotate it as desired.

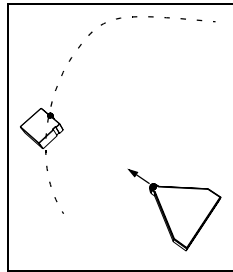
The constrained objects also rotate.

Constraining by Direction

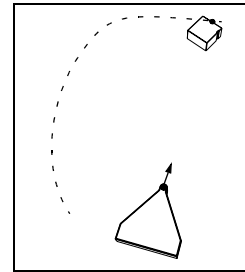
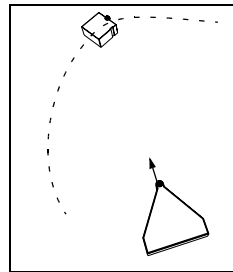
The **Constraint → Direction** command constrains the x-axis of one or more selected objects to another object, without changing the location of the objects. The x-axis of each constrained object always remains facing the centre of the constraining object.



Direction constraint
Constrain triangle to the block (A)



Move the block
Triangle is rotated so that its x-axis points to the block.



1. Select the objects to be constrained.
2. Choose the **Constraint → Direction** command in the Actor or Motion module.
3. Pick the object that is to act as a constraint.

The constraining object flashes red. The constrained objects rotate so that their x-axes face the constraining object's centre.

By default, a dotted line appears between the constraining and constrained objects to show the direction of the constraint and to make repositioning simpler. The dotted line can be hidden with the **Constraint → Show Direction** command.

4. Translate either type of object as desired.

If the constraining object is translated, the constrained object responds by rotating so that its x-axis remains facing the constraining object. If the constrained objects are translated, they also rotate so that their x-axes always face the constraining object.

Note

The rotation of the constrained object around its x-axis can be modified and keyframed by changing its x-axis rotation value (known as the “roll” angle of the object). The **Rotation** menu cells, however, continue to display the rotation values of the object in global or local coordinates.



The **Constraint → Up Vector** command can be further used on the constrained object to control the “roll” or up direction of the object.

Showing the Direction

The **Constraint → Show direction** command works in conjunction with the **Constraint → Direction** command. When a direction constraint is applied to an object and this command is activated, a dotted line automatically appears to show direction between constrained objects and the corresponding constraining object.

By default, the **Constraint → Show direction** command is activated. When it is, the dotted line is displayed.

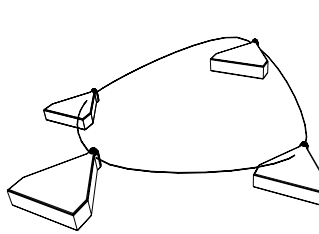
When both **Direction** and **Orientation Limits** constraints have been applied to an object, colour feedback is provided. The dotted line that automatically appears when a **Direction** constraint has been activated will change according to the following:

- A *blue* line is the default colour, which means that no orientation limit has been set.
- A *pink* line means that both orientation limits and damping have been set, and the object has been moved to where damping is activated.
- A *red* line means that both orientation limits and damping have been set, and the object has reached its maximum limit.

Using the Tangent

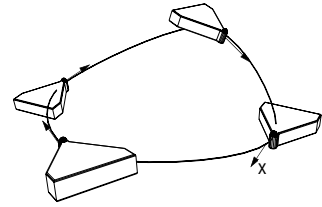
The **Constraint → Tangency** command constrains the x-axis of an object to be tangent to a curve (path); that is, the x-axis points in the direction of the curve's slope at all times.

This constraint is particularly useful for having an object follow a path's direction, such as a roller coaster car following the tracks.



No constraint

Triangle is animated along path, but does not rotate to follow the direction of the curve.



Tangency constraint

Triangle's x-axis is oriented to the tangent of the curve at any given point.

There are three ways in which you can use this constraint:

- If the object is not on a path, but is constrained to be tangent to the closest part of a curve.
- If an object is on a path and is constrained to be tangent to the curve of that path.
- The object is on a path and is constrained to be tangent to another curve. This could be useful to have a “control” curve for the tangency which is not on the actual path. The object on the path is oriented according to the tangency of the point on the constraining curve. This corresponds to the current distance that the object is on the path. For example, if an object is at 50% of the path, it will be tangent to the point at 50% of the constraining curve.

For more information on path animation, see *Animating on a Path* on page 67.

Note

You can also use this constraint in conjunction with the **Constraint → Up Vector** command to define the rotation of an object around its x-axis. Otherwise, the roll around the x-axis is determined by the x-rotation of the object.

This procedure describes the most typical way to use a tangency constraint (constraining an object to be tangent to its path):

1. Create a curve to be used as the path.
2. Select the object you want to constrain to the path.
3. Choose the **Path → Pick Path** command and pick the curve. The object becomes “attached” to the path.
4. Make sure that the object is still selected.
5. Choose the **Constraint → Tangency** command in the Actor or Motion module.
6. Pick the curve (path) to which you want to constrain the object.
The constraining object flashes red.
7. Right-click to end the mode.

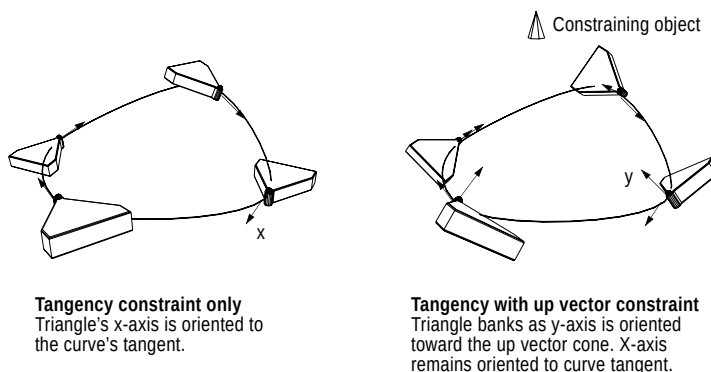
The constrained object's x-axis is repositioned in the direction of the slope of the curve. As the object moves along the path, its x-axis follows the curve's slope.

Using the Up Direction

The **Constraint → Up Vector** command allows you to control the up direction (or y-axis) of a camera or an object by using a secondary constraint object or null.

You can also apply an up vector constraint to articulated chains using the **Skeleton → Automatic Up Vector Cns** or **Skeleton → Up Vector Cns** commands.

In the case of models, the constraint has an effect only if the model has a **Direction** constraint, a **Two Point** constraint, or a **Tangency** constraint. These three constraints makes the x-axis point to a constraining object and the up vector constraint makes the y-axis point to another constraining object.



There are two reasons to control the up vector of the camera:

- To prevent the camera from flipping when it reaches a vertical axis with the interest. For example, you can produce a proper rollercoaster-like action with the camera using an up vector constraint.
- To control the banking of the camera using a separate control path.

To use the up vector constraint in its simplest form, follow these steps. This uses a camera as the example object:

1. Select the camera.
2. Choose **Constraint → Up Vector** in the Actor or Motion module and select the object to which you want to constrain the camera's up vector.

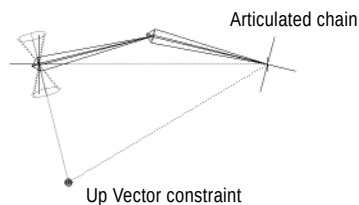
Using an independent object is not always the most effective way to control the up vector. If the up vector constraint coincides with the interest, a flip occurs on the current vertical axis. This problem can be solved by creating the following nulls and constraints. This procedure makes the up vector follow the “point of view” of an animated object exactly, allowing you to animate the camera the same way as you would animate any object:

1. Select an object.
2. Create three nulls:
 - Place one null in front of the object.
 - Place one null in the centre of the object as its “point of view.”
 - Place one null on the y-axis directly above the “point of view” null.
3. Make these nulls children of the object so that they follow the object's movements.
4. Select the camera and constrain it positionally (**Constraint → Position**) to the null in the centre of the object (the “point of view” null).
5. Select the camera interest and constrain it positionally (**Constraint → Position**) to the null in front of the object.
6. Select the camera, choose **Constraint → Up Vector**, and pick the null directly above the object's centre on the y-axis.
7. Animate the object as you like.

Now when the object moves, the camera's up vector is oriented correctly because it is constrained to the null above the object.

Using an Up Vector Constraint with Articulated Chains

You will also find the up vector constraint as **Skeleton → Up Vector Cns** command in the Actor module. The up vector constraint controls the general orientation of an articulated chain. More precisely, it specifies the y-axis of the resolution plane. The x-axis of this plane is specified by the position of the effector relative to the root of the chain.

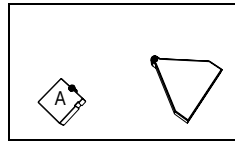


Note The general orientation of the chain changes rapidly when you move the effector into the region near the Up Vector constraint direction.

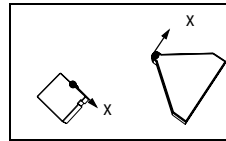
For more information, see *Constraining Joints and Effectors* on page 135.

Constraining the Scaling

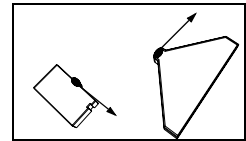
The **Constraint → Scaling** command constrains the scaling values of one or more selected objects so that they are the same as the scaling values of a constraining object. This command is useful for scaling all objects in a hierarchy at once in accordance to the scaling changes applied to one object.



Scaling constraint
Constrain triangle to the block



Scale the block in x
Triangle is scaled by the same factor.



You can also constrain one object to many, which results in the average scaling being used.

1. Select the objects to be constrained.
2. Choose the **Constraint → Scaling** command in the Actor or Motion module.
3. Pick the object that is to act as a constraint.

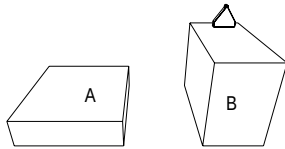
The constraining object flashes red.

4. Select the constraining object and scale as desired.

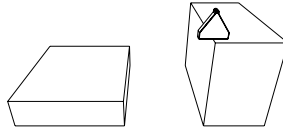
The objects constrained to it are also scaled, using the same scaling values as the constraining object.

Constraining to a Plane

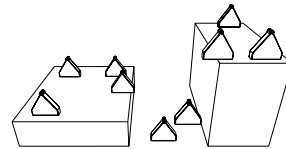
The **Constraint → Bounding plane** command makes sure that one or more constrained objects remain at the surface of the bounding plane object. This is especially useful for constraining an object's location to the floor's surface, a staircase's surface, etc.



Bounding plane constraint
Triangle is constrained to both blocks A and B.



While in vicinity of B
Triangle's y translation is adjusted so that triangle remains on top of block B.



Other positions
Triangle moves freely only when not in vicinity of either block.

The **Constraint → Bounding plane** command can be applied to all types of objects, including meta-clay elements, skeletons, faces, deformed objects, and curves. However, it can only be applied to the root of a hierarchy.

1. Select the objects to be constrained.
2. Choose the **Constraint → Bounding plane** command.
3. Pick the object that is to act as a constraint.

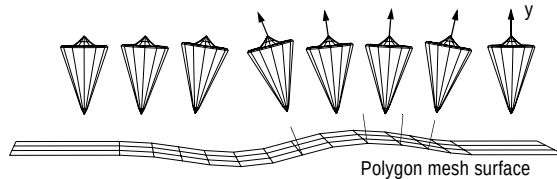
The constraining object flashes red.

4. Select either a constrained object or the constraining object, and perform a transformation as desired.

The constrained objects never moves away from the constraining object's bounding plane.

Normals to a Surface

The **Constraint → Normal to Surface** command constrains the y-axis of an object to the direction of the constraining object's surface normal to which it is closest. This constraint works only with polygon mesh objects as the surface object.



Normal to surface constraint

Cone is rotated so that its y-axis is aligned with the normal of the nearest surface polygon.

This constraint is useful for keeping an object perpendicular to a surface, usually when the surface is deformed, such as a ship sailing over a section of choppy water.

Note

You can use this constraint in conjunction with the **Object to Cluster** constraint to constrain the object to the same polygon or set of points.

1. Select the object to be constrained.

Tip

You may want to choose the **Show → Normal** command to see the object's normals. If the normals are pointing in the wrong direction, choose the **Effect → Inverse** command in the Model module.

2. Choose the **Constraint → Normal to Surface** command.
3. Pick the constraining object. This must be a polygon mesh object. The constraining object flashes red.
4. Right-click to end the mode.

The constrained object's y-axis is repositioned in the direction of the surface normals of the constraining object. As you move the constrained object, its y-axis readjusts its direction to match that of the normals to which it is closest.

Constraining between Clusters and Objects

Objects to Clusters

The **Constraint → Object to Cluster** command constrains an object's position to the geometrical (virtual) centre of a cluster of points. If the cluster contains many points, this constraint creates a “nest” for the object in the cluster. Of course, the cluster can also be a single point, which means that the object is constrained to the position of a single control vertex.

This constraint could be used for gluing features onto a face, such as constraining an eye to a cluster on a face. If you bulge the cluster out, the eye moves with it.

1. Create a cluster of points on the constraining object (tag the points and choose **Tag → Set Cluster**).
2. If the object has more than one cluster, it is the active (current) cluster that will be used as the constraint. Make sure you have selected the cluster to be current in the dialogue box that appears when you choose **Tag → Cluster List**.
3. Select the object to be constrained.
4. Choose the **Constraint → Object to Cluster** command in the Actor or Motion module.
5. Pick the object with the cluster that is to act as a constraint.

The constraining object flashes red.

6. Right-click to end the mode.

The constrained object is repositioned to the virtual centre of the active cluster's vertices. When a transformation is performed on the cluster, the object moves with it.

Clusters to Objects

The **Constraint → Cluster to Object** command performs the opposite of the **Constraint → Object to Cluster** constraint: it constrains all the points of a cluster to the centre of the constraining object or null. The cluster is gathered to one point that is repositioned at the object's centre.

This constraint could be used to create a trampoline by having each trampoline post being one object and each corner of the trampoline being constrained to a different post. Then if a character jumps on the trampoline, its corners remain constrained to the posts.

This constraint is compatible with object deformations. For example, a wave could also be applied to the same constrained object. The constraint operation is “added” to the object after the other deformation operations have been processed.

1. Create a cluster of points on the object that will be constrained (tag the points and choose **Tag → Set Cluster**).
2. If the object has more than one cluster, it is the active (current) cluster that is used as the constraint. Make sure you have selected the correct cluster in the dialogue box that appears when you choose **Tag → Cluster List**.
3. Select the object with clusters that is to be constrained.
4. Choose the **Constraint → Cluster to Object** command in the Actor or Motion module.
5. Pick the object that is to act as a constraint.

The constraining object flashes red.

6. Right-click to end the mode.

All the points in the cluster are gathered to one point and repositioned to the centre of the constraining object. When a transformation is performed on the object, the cluster follows the object.

Clusters to Cluster Centres

The **Constraint → Cluster Centre** command constrains an object’s cluster to a cluster centre. This is a way of creating a centre for a cluster, in case you didn’t create one when you initially set the cluster. The cluster centre can be a null or any other object.



When you create a cluster, you can create a centre for it automatically by selecting the **Cluster Centre** option when you choose the **Tag → Set Cluster** command.

This command is useful when doing facial animation, or any other type of animation that requires shape deformation. For more

information about using shape deformation to animate objects, see *Deformations* on page 87 of the *Animating User's Guide*.

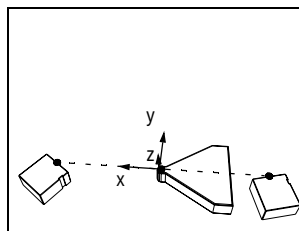
1. Create a null or any object to be the cluster centre.
2. Select the model with the cluster that you want to constrain to the centre.
3. Choose the **Constraint → Cluster Centre** command in the Actor or Motion module and pick the cluster centre object as the reference. The cluster on the model is now constrained to the cluster centre.

Unlike the **Constraint → Cluster to Object** command, the clusters are not “gathered” to a point and moved to the cluster centre with this command. You only notice the effect of the constraint when you transform the cluster centre.

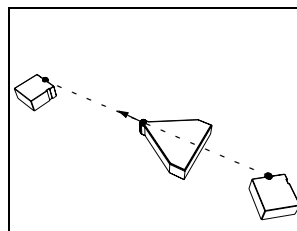
Constraining between Points

Between Two Points

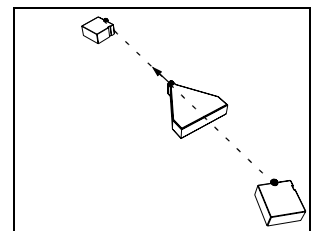
The **Constraint → Two Points** command positions an object to be constrained centrally between two other objects or nulls. The constrained object's x-axis is oriented along the vector that goes from the first selected constraint to the second one.



Two point constraint
Triangle's centre is positioned at midpoint between two blocks.

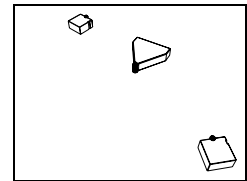
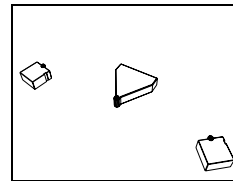


Move one block
Triangle remains at midpoint.



Note the orientation
Its x-axis is oriented along a line that would join the blocks.

Compare with position constraint
At right, a position constraint does not affect the orientation.



This constraint is useful for keeping an object between two points pointing along their direction, such as a muscle between two arm joints.

It also provides an easy way to convert two positions into an orientation for motion capture. When you have motion curve data without orientation information, you can infer rotations from position data by using this constraint.

Note

You can also use this constraint in conjunction with the **Constraint → Up Vector** command to define the rotation of an object around its x-axis. Otherwise, the roll around the x-axis is determined by the x-rotation of the object.

1. Select the object to be constrained.
2. Choose the **Constraint → Two Points** command.
3. Pick two constraining objects.

The order in which you select your constraining objects is important to get the correct orientation of the constrained object.

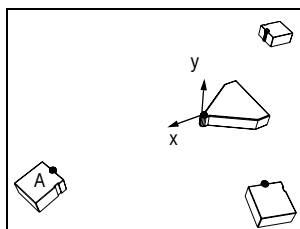
The constraining objects flash red as you pick them.

4. Right-click to end the mode.

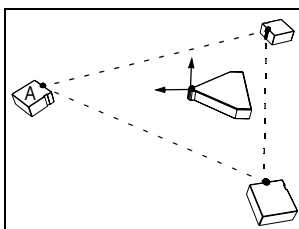
The constrained object is repositioned to be between the two constraining objects and its x-axis is repositioned in the direction of the second constraining object's centre. As you move the constraining objects (one or both), the constrained object readjusts to keep between them.

Between Two Points

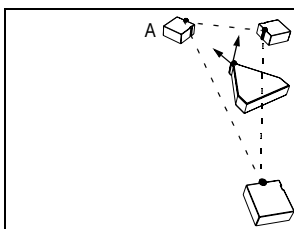
The **Constraint → Three Points** command positions an object to be constrained centrally between three constraining objects or nulls in a type of invisible triangle, and orients the object's y-axis to the normal of the plane defined by the three constraining objects. The x-axis of the constrained object is oriented toward the centre of the first constraint object you pick.



Three point constraint
Constrain triangle to three blocks.
Pick A first.



Move blocks
Triangle remains at midpoint.



Note the orientation
X axis is oriented toward A.
Y axis is oriented to normal of plane defined by three blocks.

This constraint is useful for converting three positions from motion capture data into an orientation. When you have motion curve data for positions only, such as three points on the hips and trunk of an actor, you can infer the rotation of its body by using this constraint.

1. Select the object to be constrained. If no object is selected when you pick the three constraints, a triangular face is automatically created with its end points attached to the constraints (each corner follows its corresponding constraint).
2. Choose the **Constraint → Three Points** command in the Actor or Motion module.

3. Pick three constraining objects.

The order in which you select your constraining objects is important to get the correct orientation of the constrained object. The first constraint defines the x-axis and the three constraints together define the orientation of the plane for the y-axis.

The constraining objects flash red as you pick them.

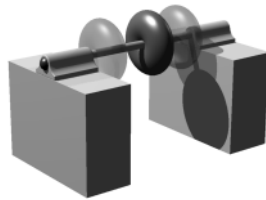
4. Right-click to end the mode.

The constrained object is repositioned to be between the three constraining objects. Its x-axis is repositioned in the direction of the first constraining object's centre, and its y-axis is oriented to be perpendicular to the plane formed by the three constraining objects. As you move the constraining objects (one or all), the constrained object readjusts to keep properly positioned and oriented.

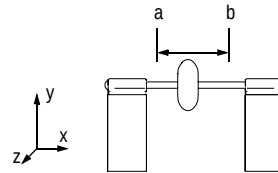
Creating Constraint Limitations

Limiting the Position

The **Constraint → Position Limits** command sets a constraint on the range of position of one or more objects relative either to another object or to the centre of the world (origin). This means that objects cannot be translated to a position further away from the constraining object than a specified distance. A damping effect can also be set, adding resistance to the object's movement so that abrupt stops are avoided.



Position limits
Bead is restricted to moving along bar and between two blocks.



Settings
Use a and b for the minimum and maximum x values.
Set y min and max to equal height of the bar.
Set z min and max to equal z position of bar.

1. Select the object(s) to be constrained.
2. Choose the **Constraint → Position Limits** command in the Actor or Motion module. The Position Limits dialogue box appears.
3. Set the **Limit** type.



When setting the Limit type's MIN and MAX translation values for each axis, select its check box first, then type the desired values into the corresponding text box. Only the values for selected check boxes are applied.

4. Select the **Reference** point of the constraining area.
5. Set the **Damping Width** and **Damping Strength**.
6. Click **Ok** to confirm the settings and exit the dialogue box.
7. If another object is to act as a constraint, pick the object following the mouse button instructions in the status bar.

The constraining object flashes red. The constrained object(s) adjusts its position according to the specified values, so that its centre

is either the specified distance from the centre of the constraining object or from the centre of the world.

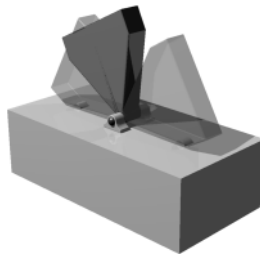
8. Select and translate the constrained object as desired.

The constrained object does not move further from the constraining object than the specified values.

For more information about the parameters available in the Position Limits dialogue box, see the description of **Constraint → Position Limits** command in the *Reference Guide*.

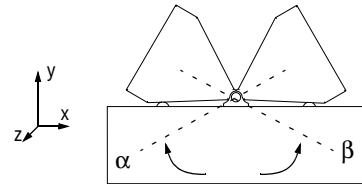
Limiting the Rotation

The **Constraint → Rotation Limits** command sets a constraint on the range of rotation of one or more objects of a chain, such as restricting a head from rotating 360 degrees (unless this is what you want!). Global coordinates are always used for the constraint unless the object is parented, in which case the parent's coordinates are used.



Rotation limits

Triangle can only rotate about its axis, and not beyond surface of block.



Settings

Determine minimum and maximum allowable values for z-rotation. Restrict rotations in x and y.

A damping effect can also be set, adding resistance to the object's movement so that abrupt stops are avoided. Damping is apparent if the object has its rotation constrained in some way (such as with rotation or direction constraint). For example, this allows the eyes of a character to follow a moving object and to damp when they reach a rotation limit. If you animate the eyes so that they rotate to the back of the head, damping and limit obviously won't apply.

Damping can be applied on joints only for forward kinematics (rotations), but not applied for inverse kinematics for a joint. In this case, only the crude limit can be applied.

1. Select the object to be constrained. You can only select one object at a time.
2. Choose the **Constraint → Rotation Limits** command. The Rotation Limits dialogue box is displayed.
3. Set the **MIN** and **MAX** rotation values for each axis desired by first selecting a check box, and then typing the desired value into its corresponding text box.



Only the values for selected check boxes are applied to the constrained object.

4. Set the **Damping Width** and **Damping Strength**.
5. Click **Ok** to confirm the settings and exit the dialogue box.
6. Select and rotate the constrained object as desired.

The constrained object never rotates beyond the minimum and maximum values set in the dialogue box.

For more information about the parameters available in the Rotation Limits dialogue box, see the description of **Constraint → Rotation Limits** command in the *Reference Guide*.

Creating Custom Constraints with Expressions

You can set constraints in SOFTIMAGE 3D using commands in the **Constraint** menu. When you set a constraint using this method, an internal expression is created which describes this constraint. For example, the position constraint between a cube and a sphere is written as the following expression:

```
cube.pos = sphere.pos
```

Although it's easy to set a constraint using the **Constraint** commands, you may want to create customized constraints. What if you want to constrain one object to another, but have it lag behind five units on the z-axis? What if you want to add a jitter to the resulting motion? How do you constrain a cone angle of a light to follow the translation of an object?

Using expressions, you can set a permanent relation between the parameter of one element and a value, such as constraining the scaling in x of a cube to always be equal to 4. You can also set a relation between one or many other parameters (like constraining a cube's colour to the average position of two models and a light). A constraint created with expressions behaves like the ones you create using the **Constraint** commands, meaning it is updated, saved, and restored with the object.

For example, you can constrain the visibility of an object to its distance from the camera in the scene.

1. Create a cylinder.
2. Select the cylinder, choose the **Info** ➔ **Selection** command, and name the cube "Goodbye."
3. Select Goodbye and choose the **Expressions** ➔ **Edit** command in the Motion module.
4. In the **Affected Element** text box, type `Goodbye.nvisib`. This means that the expression affects Goodbye's node visibility parameter.
5. In the **Expression** text box, type:

```
ctr_dist_cam( Goodbye.etrnx ) < 40
```

This expression states that Goodbye will disappear every time the camera is more than 40 units away.

6. Translate the camera in your scene and watch the cylinder disappear and reappear!

For more information on using expressions to create constraints, see *Expressions* on page 197.

Using Constraints with Inverse Kinematics

You can use constraints with inverse kinematics to create realistic character movements simply. For example, assume you want to animate an inverse kinematic character (skeleton) riding a bicycle. You can keyframe the rotation of pedals on the bicycle, then constrain the effector of the character's leg skeleton to the pedal. Now the leg of your character follows the keyframed rotation of the pedals, creating the illusion that the character is pedalling the bicycle.

For more information on inverse kinematics, refer to the *Kinematics* chapter, starting on page 127.

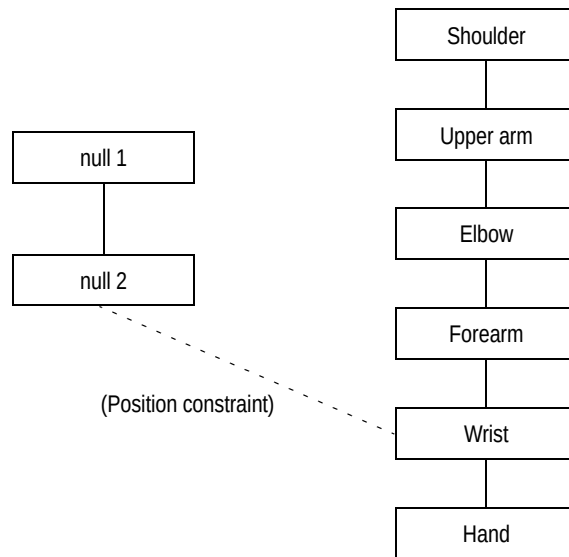
For example, if you animate a breaststroke swimmer, when its arms are extended, its elbows are positioned at the same height as its shoulders. You can apply an Up Vector constraint to its arm, located to the side of its shoulder, at the same height.

1. Draw a 2D chain of two joints in the Top window (use the **Skeleton → Draw → 2D Chain** command).
2. Choose **Get → Primitive → Null** to create a null and position it next to the shoulder.
3. Select the first joint of the chain, and choose the **Skeleton → Up Vector Cns** command. Pick the null as the constraining object for the 2D chain.

Using the null, you can now change the general orientation of the chain. For example, when you move the swimmer's hands near to the chest, the elbows must be lowered, so you lower the constraining object to move the elbows.

Using Constraints with Channel Drivers

Constraints are also useful when animating with channel-captured motion data. For example, assume you want to capture the complex movements of a dancer and apply them to a 3D model of a robot. Instead of making your channel connections directly to the components of the robot model, you can connect the channel driver to a series of null objects. During your channel capture, the null objects receive the animation data. You can then constrain the components of the robot model to the null objects:



This may sound like a complicated way to animate, but it is very efficient because the robot model may not have the same physical proportions as the real-life dancer from which you are capturing the motion. In other words, it is much simpler and faster to calibrate the animated nulls to achieve the exact motion you want instead of having to change the physical dimensions of the 3D model to fit the proportions of the real-world source of the animation. For more information about using constraints with channel captured animation data, see *Motion Capture and Control* on page 213.

Removing a Constraint

You can specify whether to remove *all* of a constrained object's constraints, or all constraints applied to one constraining object.

1. Select the constrained objects whose constraints you want to relax.
2. Choose the **Constraint** ➔ **Relax** command.

If there is only a single constraint applied to the object or one active constraint in the scene, it is relaxed.

If not, follow the mouse instructions in the status bar:

- Relax the constraints of a selected object by picking the constraining objects with the left mouse button.
- Relax *all* the constraints of a selected object by middle-clicking in a window.

The constrained objects flash red as the constraints are relaxed.



Open a DopeSheet window to keep track of your operations and view the results.

If you want to delete only one of several constraints applied to an object, use the DopeSheet to deactivate it. For more information on using the DopeSheet, see *Editing with the DopeSheet Window* on page 53.

Activating and Deactivating Constraints

You can completely deactivate and reactivate an individual constraint using the DopeSheet window.

You can mix path animation, explicit animation, and constraints for a single object by deactivating segments of animation and/or a constraint in time using the DopeSheet.

For example, assume you want to animate two characters throwing a ball back and forth, in 50 frames. The ball can be constrained to the hand of the first character from frame 1 to frame 15, then animated along a path from frame 16 to frame 35, then constrained to the second character's hand from frames 36 to 50.

To activate a constraint, click the red box to the left of the track with the constraint.

The box turns green to indicate that the track is activated.



A dotted green box indicates that some of the tracks beneath the currently displayed level are deactivated.

For more information on using the DopeSheet, see *Editing with the DopeSheet Window* on page 53.

C H A P T E R E L E V E N

Expressions

Introduction

Expressions are mathematical expressions that you can use for any parameter in SOFTIMAGE 3D that can be animated. This includes parameters such as translation, rotation, scaling, explicit translation, material, texture, etc. (see the commands in the **SaveKey** menu cell, starting on page 1241 of the *Reference Guide*).

An expression is simply a string of characters that defines a relationship between elements. For example, $a = b$ is a simple expression showing that a is equal to (constrained to) b .

Expressions have the flexibility to let you create almost any connection you like between parameters. They support references to any parameter that can be animated, predefined variables, user-defined local variables, standard math functions, predefined math macros, and random generators.

If you are knowledgeable about math functions, you will find that expressions allow you the freedom to perform animation to exact specifications. Obviously, the more complicated the task you want to perform, the more sophisticated the expression and your knowledge of functions need to be.

If you're a novice to math, there are some basic expressions you can perform without having to understand math functions well. For example, a simple constraint can be achieved, such as constraining the colour of one object to the rotation of another. In this case, all you have to know are the names of the objects and their appropriate function curves.

How are Expressions Used?

There are three ways to use expressions: as a custom constraint, to generate or edit a function curve, or to use with channels. These methods correspond to these commands in SOFTIMAGE 3D:

- Use the **Expressions** ➔ **Edit** command in the Motion module to create a custom constraint and general expression editing.
- Use the **FcrvEdit** ➔ **Expression** command in the Motion module to generate or modify a function curve.
- Use the **Expressions** button in the Channels Connection Setup dialogue box to be used with Channels.

Writing and Editing an Expression

What's in an Expression?

An expression is divided into three parts:

- The element and its parameter being affected by the expression.
- An operator (always the equal sign “=” for a custom constraint).
- The expression itself.

For example:

`cube.etrnx = sphere.etrnx + 5`

- `cube` is the element affected with a specific parameter defined (`etrnx`). This is shown in the **Affected Element** text box.
- `=` is the operator shown in the **Operator** button.
- `sphere.etrnx + 5` is the expression itself shown in the **Expression** text box.

The expression can contain various elements, as described in *Expression* on page 649 of the *Reference Guide* which describes the **Expression** text box in the Expression Editor dialogue box.

All possible operators, functions, scene elements, etc., in this dialogue box, that can be used in an expression, are described in their appropriate section in *Parameters* on page 647 of the *Reference Guide*.

Priority

Expressions have the highest priority over other types of animation. For example, if you have a cube that is animated using an explicit translation parameter, and you apply an expression like this to it:

`sin( Fc * 30 )`

then the value of the explicit translation parameter is overwritten by the `sin( Fc * 30 )` expression.

Resulting Value of an Expression

The returned value of an expression is always a floating point value (meaning it can be a number like 2.4 instead of an integer like 2). There is no validation on the returned value. That is, if the limit of a parameter is 0 and 1, then if the expression returns 2, that value is

not adjusted to fit into the limit. A good way to avoid this is to “enclose” an expression within a MIN or MAX function.

Example of Writing an Expression

In this example, you can create a simple constraint of a sphere’s scaling on x be constrained to a cone’s scaling on y. Obviously, you can come up with much more exciting possibilities, but this shows how simple it is to create a custom constraint.

To set up your scene, create a sphere, create a cone, and set keyframes for scaling on the y-axis for the cone.

1. Select the sphere to be the “affected element.”
2. Choose the **Expressions** ➔ **Edit** command in the Motion module.

In the Expression Editor dialogue box appears, follow these steps:

3. The sphere’s name (**sphere1.**) appears in the **Affected Element** text box.
4. Add the sphere’s parameter to be constrained. Click at the end of the **sphere1.** and then click the **Fcurve** button. Select the **scalx** parameter (function curve) from the Function Curves dialogue box that appears.

The parameter name is added to the end of the sphere’s name so that **sphere1.scalx** appears.

5. Now create your expression. Click in the **Expression** text box and click the **Scn Element** button. In the Select Element browser that appears, select the **cone1** element.
6. Add the cone’s parameter name that will act as the constraint. Click at the end of **cone1.** and then click the **Fcurve** button. Select **scaly** from the Function Curves dialogue box.
7. Click **Validate** to make sure that you have a valid expression.
8. Click **Ok** to apply this expression to the sphere.

Voila! The constraint is applied to the sphere. When you play back your scene, you will see the sphere scaling on its x-axis at the same rate as the cone scales on its y-axis.

Copying an Expression

You can copy the expressions affecting one element to another element of the same type (in the same chapter) such as from a camera to a camera, or an object to an object. The selected element inherits the expressions affecting the element that you have selected.

If the expression to be copied affects a parameter that is already referenced in the destination element, this expression is not copied. If an expression generates a cycle, it is not copied either.

1. Select the element that you want to receive the expression (the destination element).
2. Choose the **Expressions** ➔ **Copy** command in the Motion module. This puts you in the Copy Expression mode.
3. Pick the element whose expression you want to copy. You can pick more than one element.

The destination element inherits the expressions from the elements you pick.

4. Right-click or press Esc to end the Copy Expression mode.

Creating a Custom Constraint

Using expressions, you can set a permanent relation between the parameter of one element and a value, such as constraining the scaling in x of a cube to always be equal to 4. You can also set a relation between one or many other parameters (like constraining a cube's colour to the average position of two models and a light). A constraint created with expressions behaves like the ones you create using the **Constraint** commands, meaning it is updated, saved, and restored with the object. However, by creating constraints with expressions, you are unlimited in constraining any animated parameter to another.

For example, you can constrain the visibility of an object to its distance from another object. To do this:

1. Create a cube and duplicate it twice. Translate the duplicated cubes so they are beside the original cube.
2. Select the first cube and choose the **Info → Selection** command and name it “Elmo.” Follow the same procedure for the second and third cube, and name them “Joe” and “Carlo.”
3. Select Joe and choose the **Expressions → Edit** command in the Motion module.
4. In the **Affected Element** text box, type `Joe.bvisib`. This means that the expression will affect Joe's branch visibility.

5. In the **Expression** text box, type:

```
ctr_dist( Joe.etrnx, Elmo.etrnx ) > 4
```

This expression states that Joe will disappear every time Elmo is translated nearby.

6. Select Carlo and choose the **Expressions → Edit** command.
7. In the **Affected Element** text box, type `Carlo.bvisib`. This means that the expression will affect Carlo's branch visibility.
8. In the **Expression** text box, type:

```
ctr_dist( Carlo.etrnx, Elmo.etrnx ) > 4
```

9. Select Elmo and translate the cube in x, crossing Joe and Carlo. Notice that both Joe and Carlo disappear as Elmo moves nearby.

Editing Function Curves

You can generate a function curve for an object by creating an expression that describes that curve. This method could be an alternative to setting keyframes using the **SaveKey** menu cell (however, you must have a keyframe set before you can select the function curve in the Expression Editor). Using an expression, you can create function curves that are easily expressed using a known mathematical function. For example, an undulating (wavy) curve can be written as a simple expression using the sine function. You can also draw a curve that depends on another curve. For example, a selected function curve in the Fcurve window could be equal to the animation curve of another element plus 5 units.

You can modify an existing function curve with expressions. For example, you could add jitters to an existing scaling curve by adding an expression to it. Using expressions in conjunction with the Fcurve window brings up the same Expression Editor as when using the **Expressions → Edit** command; however, with function curves the expression is not retained as a permanent link between elements: it is merely used as a tool to draw and edit function curves. You can, of course, save any expression to a file to use it again.

1. Select an animated element. This can be an object, a light, a camera, etc. – anything with a function curve. This is called the “affected element.”
2. Choose **FcrvSelect** and the appropriate command relating to the type of function curve you want to modify.
3. Choose the **FcrvEdit → Expression** command.

The Expression Editor dialogue box appears.

There are many possibilities for using the parameters in this dialogue box, but the following example shows some of the basic steps you can do to edit an function curve.

4. The element and its function curve you selected in the scene appears in the **Affected Element** text box. This cannot be changed.
5. Now create your expression. If there is a predefined expression you want to use, click the **Select** button and choose that expression from the browser that appears.

To create an expression from scratch, follow these steps:

- If you want to create an expression that doesn't relate to any other element in the scene, skip steps 6 and 7.
 - If you want to reference the values or animation of other elements, select the element and then do steps 6 and 7.
6. Click in the **Expression** text box and click the **Scn Element** button. The Select Element browser appears in which you can choose any element from the scene.
 7. Add the scene element's function curve that will be used to affect the element's function curve. Click at the end of the element's name (after the period) and then click the **Fcurve** button. Select one of the possible function curves from the Function Curves dialogue box that appears.
 8. At this point, you could add an operator and some numeric values to this expression, such as + 3. You could also use the predefined math **Functions** such as cosine, logarithm, or conditional, or **Variables** such as current frame, first frame, or a random value.
 9. If a segment of the expression needs to be repeated, you can define it in the **Local Variables** text boxes. This lets you keep the main expression simpler. To refer to a local variable, use the corresponding letter (A, B, C, or D) in the main expression.
 10. Click **Validate** to make sure that you have a valid expression.
 11. If you want to save the expression to a file, click **Save** and specify a name for it in the Save Expression dialogue box that appears.
 12. Click **Ok** to apply this expression to your affected element or click **Cancel** to exit without saving any changes.

Experimenting with Expressions

To help you get up to speed with modifying function curves using expressions and understanding the effects of expressions, try the following "tutorial." You can open the Fcurve window to see the effect of each expression.

Start by following these steps:

1. Get a B-spline or NURBS circle and a cube.
2. Use the circle as a path for the cube (**Path** ➔ **Pick Path**) with 0 as the Start frame and 100 as the End frame.

3. In the playback box, put 0 as the start frame and keep 100 for the end frame. Make sure that the time line pointer is at frame 0 before you start.
4. With the cube selected, select its translation function curve (**FcrvSelect** → **Object** → **Translation**).

You see that the function curve values go from 0 to 100 on the vertical axis in the Fcurve window. This defines the percentage of action performed corresponding to the relative frame position of the cube on the path.

Now modify this function curve with expressions to see the effects on the animation of the cube:

5. Choose **FcrvEdit** → **Expression**. This displays the Expression Editor dialogue box in which you will define the expression.
6. In the **Expression** text box, type 50. Keep the equal sign (=) as the **Operator** and click Ok.

The function curve now has one key per frame, all at 50%, which keeps the cube at 50% of the path.

7. Choose **FcrvEdit** → **Expression** again. You can middle-click the **FcrvEdit** menu cell to choose the command last used, which is **Expression**. The Expression Editor dialogue box appears again.
8. In the **Expression** text box, delete the number 50 and type **Fc** this time. You can also select **Current Frame (Fc)** from the **Variables** menu, which puts this variable in the **Expression** text box for you with the correct syntax.
9. Click Ok.

Now the curve is linear, going from frame 0 to 100. This is because **Fc** represents the current frame, so at frame 0 it has the value 0, at frame 30 it has the value 30, etc.

However, if you change the start and end frame in this expression from 0 and 100 to 100 and 200, you'll see that the curve now goes from 100 to 200, which you don't want!

Put the start and end frames back to 0 and 100.



Each time you modify your curve, you can use the **SWAP** option in the Fcurve window title bar to return to the previous function curve (shown in dotted lines) if you want to start over with different values. This is a handy tool that

lets you practice changing the curve until you get the exact result you want, and undo mistakes that you don't want to keep!

10. Open the Expression Editor dialogue box again and delete the contents of the **Expression** text box by clicking the **Delete** button. Then type:

$100 * Fc/Fe$

Fc/Fe is the division of the current frame by the end frame value.

11. Click Ok.

Now the function curve should be the same as before, going from 0 to 100 because Fc/Fe is the division of the current frame by the end frame value (which is fixed at 100). Simply speaking, $0/100$ at the start frame has the value 0, and $100/100$ at the end frame has the value of 1. By multiplying this by 100, it gives a value between 0 and 100 again.

12. Open the Expression Editor dialogue box again and edit the contents of the **Expression** text box to be this:

$100 * Fc/(Fe - Fs)$

Subtracting by Fs (first frame) lets you create an expression that is relative to the first frame of the sequence.

13. Click Ok.

Now the function curve is the same as before (going from 0 to 100) except if you now change the start and end frames to 100 and 200, it will stay the same. This is because subtracting by Fs (start frame) allows you to have an expression relative to the first frame of the sequence. Put the start and end frames back to 0 and 100.

Note

You had to put parentheses for $(Fe - Fs)$ because of the priority of the operators. A division has higher priority than a subtraction, so having no parentheses would give this result: $100 * (Fc/Fe) - Fs$.

Try one more modification with this last expression:

14. Change the **Expression** text box to this:

$100 * (1 - Fc/(Fe - Fs))$

15. Click Ok.

Now the function curve goes from 100 to 0, because instead of using the fraction $F_c / (F_e - F_s)$, which goes from 0 to 1, you used $1 - F_c / (F_e - F_s)$, which goes from 1 to 0.

If you've had enough expressions for one day, you can stop here. If you want to do more, carry on.

Want to Try More?

If you're starting to get the hang of this, try something a bit different this time.

1. In the Expression Editor dialogue box, delete the contents of the **Expression** text box and type:

```
sin( Fc * 4 ) * 50
```

You can also choose the Sine function from the **Functions** menu to have it placed in the **Expression** text box with the correct syntax. This expression produces a sine curve, which is a wave that takes 360 degrees to complete a cycle. A cycle length is called a period.

2. Click Ok.

You should see one "wave" curve which has the value 0 at frames 0, 45, and 90, a value of 50 at frame 22.5, and a value of -50 at frame 67.5. As mentioned before, a sine curve is a wave that takes 360 degrees to complete a cycle, so by multiplying the input value of $\sin()$ by 4, you made the wave complete a cycle in 90 frames (4 times faster). By definition, a sine returns a value from -1 to 1, with $\sin(0) = 0$. When you multiplied it by 50, the curve went from -50 to 50.

Using this same expression, try some different values.

3. In the **Expression** text box, change the line to:

```
sin( Fc * 20 ) * 50 + 50
```

The curve has a shorter cycle length, and the curve goes from 0 to 100. The amplitude of the curve is still 50 units.

4. Now try:

```
cos( Fc * 20 ) * 50 + 50
```

A cosine is the same as a sine, except that its cycle is 180 degrees apart, so $\cos(0) = 1$ whereas $\sin(0) = 0$.

5. Now try:

$$\sin(Fc * (30 * Fc/Fe)) * 50 * (1 - Fc/Fe) + 50$$

Phew! You should get an oscillation that decreases in amplitude and in period (it oscillates faster). By looking at this expression one section at a time, you should understand (using the previous steps) why it gives that result.

6. Play back the resulting motion.

Try adding a value to the curve to make the cube stop at the bottom of the circle. To do this:

7. Change the **Operator** from = to + .

8. Replace the whole expression with the number 12 and click Ok.

The entire curve moves 12 units. By using the addition operation, the curve was simply modified instead of being totally overridden. Play the animation.

Now try something else with multiplication.

9. In the playback box, change the end frame to 500 so you can see the effect better.

10. In the Expression Editor, choose the = Operator.

11. Change the expression to:

$$\sin(Fc * 40) * 50$$

12. Click Ok.

You now have a straight wave curve again, going from -50 to + 50.

13. Now choose the * Operator and apply this expression:

$$\sin(Fc * 2)$$

This modulates the existing sine curve with another sine curve that has a different period. The effect is a curve that looks like a rhythmic audio signal. Multiplication lets you modify the overall amplitude of your existing curve with another curve. For example, you could multiply your curve again with:

$$1 - \sin(Fc / 4)$$


Remember that each time you modify your function curve, you can use the SWAP option to return to the previous curve to start over with different values.

For the finale, use a random value to obtain jitters on your complex function curve.

14. In the Expression Editor dialogue box, keep the * Operator.

15. Delete the previous expression and leave the cursor at the beginning of the **Expression** text box.

16. From the **Variables** menu, select **Random: 0.0 to 1.0** to insert it in the **Expression** text box. Replace the <seed> by any value. For example:

```
rand_0_1( 3 )
```

17. Multiply this expression by 2 to increase the resulting random values from 0 to 2. You should have something like this:

```
rand_0_1( 3 ) * 2
```

This is one way to add jitters to an existing curve with random values.

Using Expressions with Channels

Any time you scale or offset your channel connections, you are creating a simple expression. You can also create your own custom expressions that control the animation in your scene through channels. For example, you can create a conditional expression that states that you must hold down both the mouse and the F1 key to use the mouse to translate an object in your scene.

1. Create a cone.
2. To activate the mouse channel driver for the cone, choose the **Channel → Channel Setup** command in the Motion module. The Channel Setup dialogue box appears.
3. To add the mouse driver to the **Driver** scroll box, click **New** and select the driver from the browser.
4. Follow the same steps to activate the keyboard driver.
5. Choose the **Channel → Connection Setup** command. Select the **mouse.xpos** and **keyboard.key_F1** in the Source scroll box and **cone1.etrnx** in the Destination scroll box. Click **Connect**.

An expression is created because you are connecting two sources to one destination. To view the expression, click **Show Expressions**.

```
(CHN mouse.xpos + CHN keyboard.Key_F1) / 2 → cone1.etrnx
```

You can modify this expression to create a condition in which you must hold down the F1 key to use the channel connection.

6. Click the **Expression (verify)** button at the bottom of the dialogue box. The Channel Expression Editor appears.
7. Type the following into the Expression text box:


```
cond( CHN keyboard.Key_F1 == 1, (CHN mouse.xpos * 10) -5, 5)
```
8. Click **Ok** to apply the expression to your channel connection. Exit the Channel Connection dialogue box by clicking **Ok**.
9. Choose the **Channel → Channel Run → Sequence I/O** command.

Notice that if you do not hold the F1 key down, the mouse does not do anything, and the object translates by 5 units then stops. When you press the F1 key, you can move the mouse to translate the object.

C H A P T E R T W E L V E

Motion Capture and Control

Introduction

Using Channels for Motion Capture and Control

Channels are links between SOFTIMAGE 3D and external devices. Channels are used for stop-frame animation and to capture real-time animation from live motion, voices, music or other digital signals. A channel is a single connection to a degree of freedom of an external device. For example, the `mouse.xpos` parameter represents a single channel.

You can also use channels to control or manipulate external devices, where animation data is output from SOFTIMAGE 3D and used to control an external device (such as an electronic puppet). The same tools are also provided for loading prerecorded motion files of several formats.

The basis of all channel-driven animation is defining a source of animation data and connecting it to a destination element. You can make capture, control, and direct channel connections in SOFTIMAGE 3D.

Using Channels with Virtual Sets and Actors

You can use channels drivers and SOFTIMAGE 3D or the SOFTIMAGE Live viewer to control animated characters and to allow them to interact with live actors. You can create a scene in which you have designated all the channel connections of a character, then either do a channel run in SOFTIMAGE 3D or in the SOFTIMAGE Live viewer.

For more information about creating a virtual set and designing computer-generated characters to use in a virtual application, see the following chapter, *SOFTIMAGE Live* on page 251.

Overview of Channel Connections

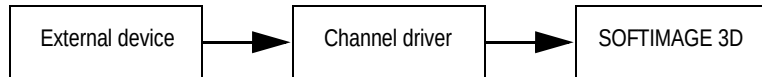
Types of Channel Connections

The basis of all channel-driven animation is defining a source of animation data and connecting it to a destination element. You can make three types of channel connections in SOFTIMAGE 3D:

- Channel to function curve connections (capture).
- Function curve to channel connections (control).
- Channel to channel connections (direct).

Channel to Function Curve Connections

These are also called *input* type connections since data is transmitted from an external device into SOFTIMAGE 3D. This is the type of connection used to capture animation from an external device. For example, assume you want to use your mouse to rotate an object around its y-axis in SOFTIMAGE 3D. The source of the animation data is the mouse channel driver, and the destination is the y-axis rotation parameter of the selected object:



SOFTIMAGE 3D automatically generates raw function curves from the stream of captured data. These function curves can then be connected or constrained to objects, lights, the camera, or any parameter that can be animated in a scene.

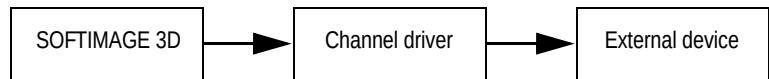
When a channel driver captures animation, the stream of data is fed into SOFTIMAGE 3D as fast as possible. SOFTIMAGE 3D sets a key for each frame of animation data as it enters from the channel driver. Since the capturing frequency is independent of the animation frequency (one key per frame), the result may be that as many keys as possible per frame are created; it may be one key per frame or one key for every six frames, depending on the type and speed of the input devices. In other words, the channel driver interprets the animation data and sets keyframes automatically, generating function curves as fast as the driver can receive the information from the input device.

An advantage to using channels for motion capture is that you can process up to 1000 channels simultaneously for any number of

external devices, which allows you to generate raw function curves very quickly. Channels also permit you to view the resulting animation on screen simultaneously with the capture for immediate feedback. Afterward, you can edit the function curves to achieve precisely the effects you want.

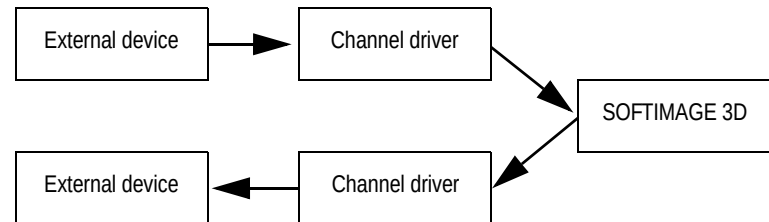
Function Curve to Channel Connections

These are also called *output* type connections since data is going out of SOFTIMAGE 3D to control an external device. This type of connection is used for motion control, where a function curve controls an external device:



Channel to Channel Connections

These are another form of output type connections, sometimes called *direct* connections. In this situation, one channel driver inputs information into SOFTIMAGE 3D from an external device, and another channel driver exports the data to another external device:



Selecting and Assigning a Channel Driver

A channel driver is a program which imports or exports the stream of data between an external device and SOFTIMAGE 3D. Some channel drivers can also import animation data from other animation files that are saved on your hard disk.

A channel driver may be an input driver, which has only input channels, an output driver that has only output channels, or an input/output driver that has both types of channels.

Each channel driver presents you with a set of named channels, which usually represents the degrees of freedom of the device. For example, if you are making an input connection, and your input device is a mouse, you have two degrees of freedom: you can move the mouse horizontally on the x-axis or vertically on the y-axis.

A channel setup consists of a group of drivers, the drivers' available channels, and a set of existing channel connections. A channel setup may contain an arbitrary number of channel drivers, all of which could run simultaneously. All the channels for all the drivers can be connected to any object's function curve.

Note

If you frequently use the same channel configuration for a particular device, you can “pre-connect” setups of channel connections and save them in dummy scenes. Then when you want to use that device to capture motion, you load the dummy scene and away you go!

You can set your drivers to be synchronized with the SOFTIMAGE 3D channel run update/refresh operation to be free running or running at a specific frequency. Drivers may also run remotely on another machine. For more information about channel driver options, see *Channel ‘Get Channel Setup* on page 160 of the *Reference Guide*.

Selecting a Channel Driver

1. Choose the **Channel → Channel Setup** command in the Motion module. The Channel Setup dialogue box appears.
2. Select one or more drivers from the **Drivers** scroll box.

These parameters available vary depending on the driver you have selected. See the *Channel Driver Reference* in Appendix A for more information.

3. Click **Ok** to accept any changes or **Cancel** to exit the dialogue box.

Once you have selected a driver, you are ready to connect it to source and destination parameters.

Creating Capture Connections

Before making a capture connection, you must have included an input channel driver using the **Channel → Channel Setup** command. For more information, see page 137 of the *Reference Guide*.

1. Select an element (object, camera, light, etc.) in your scene.

To select materials and textures, you must be in the Schematic window with the Matter mode selected. Depth-fading function curves can be accessed only if you select a camera. Refer to the **FcrvSelect** commands starting on page 711 of the *Reference Guide* for a description of all available object or item parameters that can be animated.

2. Choose the **Channel → Connection Setup** command in the Motion module. The Connection Setup dialogue box is displayed.

Note

You can also open this dialogue box by double-clicking on an existing channel connection listed in the Channel Setup dialogue box.

3. Select the **Capture (In)** option at the top of the dialogue box.
4. If you have correctly assigned your channel driver using the **Channel → Channel Setup** command, the degrees of freedom for your channel capture device are listed in the **Source** scroll box.

Degrees of freedom basically dictate the animation possibilities of your channel driver. For example, the mouse channel driver has two degrees of freedom: you can move the mouse horizontally on the x-axis or vertically on the y-axis.

5. Select the degrees of freedom in the **Source** scroll box that you wish to apply to the animatable parameters you selected in the **Destination** scroll box.
6. Pick an animatable parameter for the object you selected from the **Destination** scroll box. The destination is where the function curve is generated and applied to the animatable parameter of the object.

Note

You can also select multiple sources and destinations before making the connection. The text for the **Connect** button changes to display how many connections have been made. For example, **>> Connect >>** means that more than one source and only one destination have been connected.

7. Select the **Connect** button to assign the Source (the channel driver's degrees of freedom) to the Destination (the animatable parameter of the selected object). A new channel connection is added to the channel connection scroll box beneath the **Connect** button.
8. If you are dissatisfied with the connection you made, select the connection in the **Channel Connection** scroll box and click the **Disconnect** button. Now you can connect another degree of freedom to another parameter that can be animated.
9. You can scale or offset the incoming or outgoing data, if desired. Simply type the values you want in the **Scale** and **Offset** text boxes at the bottom left of the dialogue box. You can also select multiple channel connections and set the offset and scaling, and activate or disconnect multiple channel selections simultaneously.
10. You can also use expressions to modify the function curves generated during the channel capture by clicking the **Expression** button beneath the **Channel Connection** scroll box, or double-clicking on a channel connection. When you make a simple channel connection, an expression is created:

$f_{cv} = CHN$

If you edit the offset or scaling values, the expression becomes:

$f_{cv} = CHN * (scaling) + (offset)$

If you make a connection by supplying multiple inputs to one destination then the expression averages the inputs:

$f_{cv} = (CHN1 + CHN2 + CHN3) / 3$

These expressions may be viewed either by selecting the **Show Expressions** option in the **Channel ➔ Connection Setup** dialogue box, or by selecting the connection in the Channel Connection scroll box and clicking the **Expression** button to open the Channel Expression Editor.

Note

The Channel Expression Editor is similar to the Expression Editor dialogue box accessed from the **Expressions ➔ Edit** command. The only restriction is that you cannot include an output channel in an expression. Output channels are reserved for the target or affected element of control or direct connections.

In the Channel Expression Editor, you can enter any type of expression, even expressions not involving a channel, such as:

`obj1.fcv = obj2.fcv`

In this case, `obj2.fcv` will have data recorded during a sequence input (even though it is really an object to object connection rather than a channel to object connection).



All channel expressions specified in the Channel Expression Editor are only evaluated during a channel run and override constraints and expressions specified elsewhere in the system.

For more information about using expressions, refer to page 197.

- 11.If you want to reconnect the source elements of an existing connection to a different destination, select the existing connection, click on the **Reconnect** button, and select the new destination. The source element is reconnected to the new destination and the previous connection is disregarded. This saves you time in reconnecting existing channel connections.

Note

If you want to create a new channel connection using the same source elements as another existing connection, select the connection, click the **Reconnect** button and then hold the Shift key while selecting the new destination. The existing connection remains the same and a new connection is added to the connection list.

- 12.Click Ok to accept the channel assignment, or Cancel to exit.

Creating Control Connections

Before making a control connection, you must have selected a channel driver with output channels using the **Channel → Channel Setup** command. For more information, see page 137 of the *Reference Guide*.

1. Select an object whose animation you want to use to control an external device. This object can be animated using function curves, or you can interactively control the object's animation by using input drivers, constraints, or expressions.
2. Choose the **Channel → Connection Setup** command. The Channel Connection Setup dialogue box is displayed.



You can also access the dialogue box by double-clicking on an existing channel connection listed in the Channel Setup dialogue box.

3. Click on the **Control (Out)** option at the top of the dialogue box.
4. Pick a function curve from the object(s) you selected in the **Source** scroll box.
5. If you have selected an output channel driver using the **Channel → Channel Setup** command, the output channels for your channel driver will be listed in the **Destination** scroll box. Select the channels in the **Destination** list that you wish to be controlled by the function curves you selected in the **Source** scroll box.
6. Select the **Connect** button to assign the **Source** (the function curve of the selected object) to the **Destination** (the output channel). A new channel connection is added to the channel connection scroll box beneath the **Connect** button.
7. If you are dissatisfied with the connection you made, select the connection in the channel connection scroll box and click the **Disconnect** button. Now you can connect another function curve to another output channel.
8. You can scale or offset the output data from the function curves, if desired. Simply type the values you want in the **Scale** and **Offset** edit boxes on the bottom left of the dialogue box.
9. You can also use expressions to modify the output data from the function curves by clicking on the **Expression** button beneath the **Channel Connection** scroll box. See page 220 for information on using expressions with channel connections.

10. Click **Ok** to accept the channel assignment, or **Cancel** to exit.

Creating Direct Connections

Before making a direct connection between external devices through SOFTIMAGE 3D, you must have defined the channel drivers using the **Channel → Channel Setup** command. For more information, see page 137. Remember to define the input channel driver and the output channel driver.

1. Choose the **Channel → Connection Setup** command. The Channel Connection Setup dialogue box is displayed.



You can also access the dialogue box by double-clicking on an existing channel connection listed in the Channel Setup dialogue box.

2. Click on the **Direct (Through)** option at the top of the dialogue box.
3. Pick the channels from the Input channel in the **Source** scroll box.
4. Select the Output channels in the **Destination** scroll box that you wish to be controlled by the Input channels you selected in the **Source** scroll box.
5. Click the **Connect** button to assign the **Source** (the input channel) to the **Destination** (the output channel). A new channel connection is added to the channel connection scroll box beneath the **Connect** button.
6. If you are dissatisfied with the connection you made, select the connection in the **Channel Connection** scroll box and click the **Disconnect** button. Now you can connect another input channel to another output channel.
7. You can scale or offset the data being sent to the output channel driver, if desired. Simply type the values you want in the **Scale** and **Offset** text boxes on the bottom left of the dialogue box.
8. You can also use expressions to modify the data being sent to the output channel by clicking on the **Expression** button beneath the **Channel Connection** scroll box. See page 220 for information on using expressions with channel connections.



You can also combine function curves and input channels for the source of the connection. For example, $CHNout = FCV + CHNin$.

9. Click Ok to accept the channel assignment, or Cancel to exit.

Activating and Deactivating Channel Connections

The **Channel → Activate Selection** command activates the channel connections of all selected objects in a scene.

1. Select a channel object.
2. Choose the **Channel → Activate Selection** command in the Motion module.

A message is displayed in the status bar confirming activation. If you did not select a channel object, a warning message is displayed.

The **Channel → Deactivate Selection** command deactivates the channel connections of all selected objects in a scene. A message is displayed in the status bar confirming deactivation. If you did not select a channel object, a warning message is displayed.

Resetting the Channel Setup

The **Channel → Reset All** command resets the channel setup and all item setups to the default values.

When you choose this command, all drivers and connections are deleted from the Channel Setup and Connection Setup dialogue boxes. The default mouse driver remains.

Displaying the Refresh Rate

You can toggle the display of the refresh rate in the status bar during frame and sequence input. The **Display Refresh Rate** toggle is available with both the **Channel → Channel Run** and **Channel → Take Recording** commands.

Activating and Deactivating Supra Keys

The Supra Keys Enabled toggle lets you enable or disable Supra key functions during a channel capture or control.



It may be important to disable the Supra keys when you are using a keyboard driver.

When they are enabled, you can use any of the Supra keys described here in any channel mode.

Object Transformations

The transformation modes are active only while the key is pressed. It is also possible to change the corresponding transformation modes below the timeline while the key is pressed. These operations are the important ones for calibration.

Play Control

Left mouse can be used to move the time line pointer and change any of the play modes in the playback box.

Arrow keys are used to go to the first frame (up arrow), last frame (down arrow), next frame (right arrow), or previous frame (left arrow).

Channel Operations

h - toggles between PAUSE and RUN mode.

For more information about keyboard shortcuts and Supra keys, refer to the **Info → Supra Keys** command on page 909 of the *Reference Guide*.

Types of Motion Capture Devices

Magnetic Tracking Systems

The basic concept behind a magnetic 3D tracker is to calculate the position and orientation of the sensors within a magnetic field. A magnetic system is comprised of two main devices: an antenna (transmitter) and sensors (receiver). Each component is linked to a central unit via electrical cables. Positional and rotational data is captured in real time.

The following drivers are used with magnetic devices:

- polhemus (Polhemus Fastrack)
- ultratrak (Polhemus Ultratrak)
- ascension (Ascension Flock of Birds®)

Advantages of a Magnetic System

- Each sensor has six degrees of freedom (one magnetic sensor equals three optical sensors). This is the most important advantage.
- Since the magnetic field can pass through the human body, the sensors are always “visible” to the antenna. In fact, the magnetic field passes through everything, even water, except metallic objects.
- Moving the antenna(s) and its magnetic field increases the range of the capture. It’s even possible to capture the motion of one antenna by attaching a sensor of a second antenna!

Disadvantages of a Magnetic System

- Metallic objects interfere with the motion capture.
- Cables linking the sensors to the transmitter can considerably limit the movement of an actor. Despite this limitation, actors have done back flips while hooked up to eight sensors!

Optical Tracking Systems

With an optical motion capture system, the performer wears reflective markers – unlike magnetic capture devices, there are no cumbersome cables attached to the performer. The markers are lightweight and there may be as many as 50 or 100 markers for a single capture session. Since the performer is untethered, it is

possible to capture motions which would be impossible with a tethered system, such as gymnastics or animal behaviour.

The markers are illuminated and filmed by one or more cameras. From each camera view a set of 2D marker points is computed, and these 2D point-sets are cross-correlated to compute a set of 3D positions. A tracking procedure is required to maintain a point's identity and to recover from occlusion. In other words, the tracking procedure is used to compensate for a loss of motion data if the marker is obscured from the view of one or more cameras.

It is not usually possible to automate the 3D tracking perfectly and you will probably need to clean the data. Optical 3D tracking, therefore, is usually an off-line process where the result is a high-precision motion data file. Some systems also specify a skeleton or hierarchy format as well as the motion data.

The following drivers are used with optical devices:

- Acclaim
- MotionAnalysis
- Biovision
- Superfluo

Advantages of an Optical System

- Facial animation can be captured by using very small reflectors glued to the face of an actor.
- Optical devices allow you to record objects of any size in fast or slow motion, but the cameras and reflectors must be appropriately sized.
- Better spatial and temporal precision.

Disadvantages of an Optical System

- The opacity of the moving elements (legs, arms) or environment (chairs, tables, columns) randomly hides the reflectors during a motion capture session.
- Since the system captures the position of the reflectors in space, the orientation of the sensors must be calculated. With two points, you can calculate a vector defined by two angles of rotation (yz). By adding a third point, you can deduce the orientation of the (x) axis. To obtain six degrees of freedom (PosXYZ and RotXYZ) of a point

in space, you need three sensors. However, a larger quantity of sensors introduces the risk of confusion.

- Reflective materials such as plastic, glass, and mirror can create phantom points, which add noise to the capture.
- An optical system does not work through translucent materials that deform the image of the performer and the sensors.
- If you choose to use several cameras during the motion capture, it may take up to two days to install the cameras. If the setup is very complex, it may become difficult for the actor to move around.

Electro-mechanical, Electrostatic, and Electronic Devices

These capture systems are used for many different applications. Aside from the mouse, microphone, keyboard, dial box, and data glove, a multitude of systems have been designed for specific applications. Any type of measurement tool that has its own sensor can be used to capture motion. Linear and rotational potentiometers, and coding wheels are good examples of devices that can be used to capture data.

Examples of measurements that can be captured using these devices:

- Geometrical measurements: distance, angle, surface, volume
- Weight measurements: mass
- Time measurements: duration, frequency
- Mechanical measurements: speed, acceleration, strength, energy, power, pressure
- Electrical measurements: intensity, tension, power, resistance
- Thermal measurements: temperature
- Optical measurements: intensity, flow, light, colours
- Acoustical measurements: frequencies, noise
- Midi
- Alleffects Waldo

Capturing Motion in SOFTIMAGE 3D

There are two modes to capture motion within SOFTIMAGE 3D: using the **Channel → Channel Run** commands or the **Channel → Take Recording** commands.

During a channel run, drivers are executing and transmitting or receiving data to or from objects in SOFTIMAGE 3D. The following modes are all channel run modes:

- Frame Input
- Sequence/Input
- Fast Sequence Input

Take Recording versus Channel Runs

Take recording differs from channel run sequence input recording, in which SOFTIMAGE 3D gets the data from channel drivers and immediately creates function curves for the captured motion. During a Sequence channel run, the time line is moving, and channel data is transmitted directly to function curves. Take recording simply dumps the channel data into a file from which it can be retrieved later. One benefit of doing a take recording is that you can do a long take session and load only portions of the data back into SOFTIMAGE 3D since the data is saved to file. You can also do channel runs and take recording motion capture sessions from the SOFTIMAGE Live Viewer. See *Recording Takes* on page 263 for more information.

Take recording allows you write raw high-resolution motion files directly from the channel drivers during a channel capture (in the SOFTIMAGE Live viewer or in Take Record mode in SOFTIMAGE 3D), and defer loading them into SOFTIMAGE 3D (and the creation of function curves, etc.) until a later time.

When you set your channel drivers to be free-running, this means that each of the channel drivers in a take record can run quickly at their own rate, and write motion data as fast as they can into saved files. The drivers are not constrained to how quickly they can run by each other or by SOFTIMAGE 3D, and no data is missed (every data item that is set is saved to file). At the same time, you get lower resolution visual channel feedback showing what is happening in the viewer or SOFTIMAGE 3D as the take is being done. For more information about drivers, see page 7 of *Appendix A*.

Doing a Channel Run

The **Channel → Channel Run** commands in the Motion module allow you to create animation generated by all active input devices for a single frame, and to record animation generated by all active input devices for a sequence of frames in real time. These commands also allow you to drive output devices.

Frame Input and Output

The **Channel → Channel Run → Frame I/O** command allows you to create keyframed animation generated by all active input devices for a single frame.

1. Perform the necessary channel and connection setup operations.

See page 137 and page 143 of the *Reference Guide* for more information on Setup commands.

2. Move to the desired frame using the time line pointer.
3. Select the object(s) you want to animate.
4. Choose the **Channel → Channel Run → Frame I/O** command.
5. Move or activate the input devices using the following mouse buttons:
 - The left mouse button has no function.
 - The middle mouse button is **Keyframe**. Use it to set the current channel's values for the keys corresponding to the current frame.
 - The right mouse button is **End**. Use it to exit the mode.

When in frame input mode, you can:

- Change the current frame position of the animation by dragging the time line pointer or by clicking anywhere along the time line to move directly to another frame.
- Change the current frame position of the animation by using the arrow keys. The left and right arrow keys move one frame at a time forward or backward, while the up and down arrow keys jump to the first and last frames of the animation. When the left and right arrow keys are held down for more than 3/4 of a second, the auto-repeat function is activated.
- Play the animation by clicking on the forward arrow in the playback box, or select any of the playback box icons.

- Perform camera operations using the **z** (zoom), **a** (frame all), **f** (frame selected), **o** (orbit), and **p** (dolly) Supra keys. You can also select other objects by using the space bar Supra key, and perform object transformations using the **x** (scale), **c** (rotate), and **v** (translate) Supra keys. You can also change transformation modes when **x**, **c**, or **v** are pressed.

For more information on keyboard shortcuts and Supra keys, refer to the explanation of the **Info → Supra Keys** command on page 909 of the *Reference Guide*. For more information on transformation modes, refer to page 14 of the *Reference Guide*.

Sequence Input and Output

The **Channel → Channel Run → Sequence** command allows you to record animation generated by all active input devices for a sequence of frames, in real time.

1. Perform the necessary channel and connection setup operations.

See page 137 and page 143 of the *Reference Guide* for more information on setup commands.

2. Select the objects you want to animate.
3. Choose the **Channel → Channel Run → Sequence I/O** command.
4. Move or activate the devices as desired.
 - The left mouse button is **Run** with a countdown. This gives you a time delay before initiating and performing your sequence input or output.
 - The middle mouse button is **Run** or **Pause**.
 - The right mouse button is **End**. Use it to exit the mode.

When the end of the frame range is reached, the capture mode is not automatically ended. Instead, it goes into Pause mode so you can choose the next operation.

If an Fcurve window is displayed, captured information is added to the function curves as the capture proceeds. You can improve the capture rate if it is run with no Fcurve window displayed because the function curves are not updated until after the capture has been completed.

A status message is displayed in the status line telling you that the scene is updating, and all raw captured information is then added to

the function curves. If high density function curves are captured, there is a delay before the function curves are updated.

See *Frame Input and Output* on page 230 for more information about what you can do when you are in sequence input pause mode.

Fast Sequence Input and Output

The **Channel → Channel Run → Fast Sequence I/O** command is similar to the **Sequence I/O** command. It allows you to record animation generated by all active input devices for a sequence of frames, but the speed of the recording is increased because the screen is not refreshed after every keyframe.

1. Perform the necessary channel and selection setup operations.

See page 137 and page 143 of the *Reference Guide* for more information on setup commands.

2. Select the object(s) you want to animate.
3. Choose the **Channel → Channel Run → Fast Sequence I/O** command.
4. Move or activate the devices as desired.
 - The left mouse button is **Run** with a countdown, giving you a time delay before initiating the Fast Sequence input or output.
 - The middle mouse button is **Run** or **Pause**.
 - The right mouse button is **End**. Use it to exit the mode.

When the end of the frame range is reached, the capture mode is not automatically ended. Instead, it changes to Pause mode so you can choose the next operation.



When in the fast sequence mode, you can perform the same Supra key and mouse operations as in Sequence Input mode.

Doing a Take Recording

A take is a collection of raw motion files for a group of channel drivers participating in a channel run.

1. Select your channel drivers in the **Channel → Channel Setup** command.
2. Choose the **Channel → Connection Setup** command. In the Channel Connection dialogue box that appears, set your connections. For more information about connecting channels, see page 216.
3. Select the **Channel → Take Recording → Setup** command to specify the directory in which you want to save the take data.
4. Select the objects you want to animate.
5. Choose the **Channel → Take Recording → Run** command. The status bar displays what each mouse button does. Click the left mouse button to begin the take recording.

Note The time line does not advance, so you can do take recording runs of unlimited length. If the **Channel → Take Recording → Display Refresh Rate** command is selected, then the status bar displays the time advancing as the recording continues. If this command is not selected, the text in the status bar pulses, indicating that recording is active.

6. To end the take recording, click the left mouse button again.

The take recording data is written directly to a file in either binary or ASCII format. You can use the `cvt_take` standalone to convert ASCII and binary files to and from each other. Since binary take recording files take up approximately 35% less space than ASCII files, this is a useful tool.

Take Directory and Data Format

A take is a collection of raw motion files for a collection of channel drivers participating in a channel run.

The take name is used to create a take directory, and the `.CHD` extension is attached. Each channel driver is listed in this take directory, and also has the `.CHD` extension. For each subsequent recording of take data, the take number next to the driver name is automatically incremented.

Suppose you have specified that the take directory is `/usr/tmp` (the default) and the take name is `MondaySet1`. Also suppose that your scene has two channel drivers (mouse and keyboard) and you record three takes. Your final take directory structure will look like this:

```
/usr/tmp/MondaySet1.CHD
```

```
TAKE.0001
mouse.00.chd
keyboard.00.chd
TAKE.0002
mouse.00.chd
keyboard.00.chd
TAKE.0003
mouse.00.chd
keyboard.00.chd
```

A directory is created for the take name with the `.CHD` extension.

Inside that directory is a numbered TAKE directory. Each recording results in a new TAKE directory which is automatically incremented.

Inside the TAKE.000n directory is one file for each driver in the recording session.

Here is an example data file:

```
Channel_Driver_Take_File V1.10
```

```
#
# Creation: Wed Feb  7 16:41:42 1996
# User      : bob
#
#
FILE_TYPE    : SI_TAKE_FILE_ASCII
TAKE_FILE    : /var/tmp//lucy.CHD/TAKE.0002/
mouse.00.chd
SCENE_NAME   : no_name.1-0
DRIVER_NAME  : mouse
NUM_DOFS     : 2
MIN_TIME     : 3.6908
MAX_TIME     : 4.1037
START_TC     : none
```



```
END_TC      : none
DOF_GROUPED : FALSE
DOF_COUNTED : FALSE
START-OF-DATA
#           time DOF      count      value
3.690757    0      0      0.198593
3.691655    1      0      0.240469
3.767534    0      1      0.204848
3.767646    1      1      0.229717
3.810026    0      2      0.216575
3.810130    1      2      0.214076
3.853613    0      3      0.236122
3.853720    1      3      0.187683
3.895246    0      4      0.267396
3.895350    1      4      0.160313
3.941459    0      5      0.333073
3.941570    1      5      0.129032
3.982623    0      6      0.385457
3.982829    1      6      0.123167
4.023767    0      7      0.440969
4.023872    1      7      0.1261
4.063893    0      8      0.492572
4.063999    1      8      0.144673
4.103721    0      9      0.536357
END-OF-DATA
```

Loading the Take Data

You must choose the **Channel → Take Recording → Load Take Data** command to load the data you have recorded during your take recording. This is not done automatically when you end the take recording run.

1. Choose the **Channel → Take Recording → Load Take Data** command. The Load Take Data dialogue box appears.
2. Specify the portion of take data you want to load.
3. Click Ok to load the data into SOFTIMAGE 3D.

The take data that is loaded at the current frame time and the re-evaluation that occurs places evenly spaced keys for all channel function curves. When you load take data, raw function curves are generated (one key per frame) regardless of the spacing of the data samples in the file. In other words, the scene is evaluated at each frame, and keys are generated for all channel-connected function curves.



Once you have loaded the take data, you can convert the raw function curves to spline if you find that the key points are too dense. To do this, choose the **FcrvEdit → Fit Spline to Fcurve** command.

Saving a Channel Setup File

The **Channel → Save Channel Setup** command in the Motion module allows you to save a channel setup file. A channel setup file consists of the device configuration and the function curve assignments that you created with the **Channel → Channel Setup** command.

1. Choose the **Channel → Save Channel Setup** command. The Save SETUP_CHANNELS dialogue box is displayed.
2. Enter a name for the channel setup, and a prefix if desired.
3. Click **Save** and the channel setup is saved to the CHANNELS chapter of the current database.

Copying Function Curve Connections

The **Channel → Copy Fcurve Connections** command in the Motion module allows you to copy the function curve setup defined for one object or item to another of the same type.

Note Only Input channel connections are copied from one object to another, since many objects may be driven by the same channels. Output connections are not copied.

1. Select one or more objects to which you want to copy channel connections.
2. Choose the **Channel → Copy Fcurve Connections** command.
3. Pick the source object (the one from which you want to copy channel connections) following the mouse instructions in the status bar.

The picked object flashes red and the number of channel connections successfully copied is displayed in the status bar. You can continue to pick as many objects as desired.

When the operation is completed, the total number of channel connections copied from the source to the selected object(s) will be displayed.

4. Right-click to end the mode.

Note If the selected object already has some of the same channel connections that you are copying to it, they are not overwritten.

Loading Motion Capture Files

The **Channel → Get Motion File** commands load motion capture files of different formats. At present, all file formats represent motion data captured using optical technology.

The **Channel → Get Motion File** commands are implemented as custom effects. Their associated executables and **.cus** files are found in the **/chnlDriver** subdirectory of the directory where SOFTIMAGE 3D is installed. Ensure that the directory specified by the **SI_CHNLDRIIVER** environment variable contains these files. See the *SOFTIMAGE 3D Installation Guide* for more information about environment variables.

1. Choose the **Channel → Get Motion File → file type** command. A dialogue box appears with specific options for that type of motion capture file.
2. Enter the information into the dialogue box and then click Ok. The motion file is loaded with whatever setup is necessary (skeleton file, nulls, etc.)

For more detailed information about the motion capture data files, see the *Channel Reference* in Appendix A.

You can load the following types of animation files:

Acclaim_Skeleton Files

Acclaim is a company that specializes in motion capture, and has developed its own optical capture technology. The **Acclaim_Skeleton** custom effect is used to automatically create a skeleton composed of 3D chains in SOFTIMAGE 3D, from a skeleton file. The **Acclaim_Skeleton** custom effect also creates a channel setup file which contains all the required channel connections to capture on the skeleton.

Biovision_Animation Files

Biovision is a company that uses optical technology to capture motion data. They have a proprietary file format for motion capture data with the file suffix of **.bva**. The **Biovision_Animation** custom effect reads a Biovision data file and transmits the animation data to SOFTIMAGE 3D.

MotionAnalysis_Animation Files

MotionAnalysis_Animation loads an animated set of nulls under a single parent. The Motion Analysis file format specifies translational animation.

Superfluo_Animation Files

The **Superfluo_Animation** custom effect loads an animated set of nulls under a single parent. The Superfluo file format specifies translational animation.

Calibrating your Animation Data

Calibration is very similar to frame input: animation data is captured from the external device, allowing you to view the transformations on the connected objects in your scene updated at every frame. The main difference between calibration mode and frame input mode is that you cannot make a keyframe or capture a sequence. In other words, no function curve is actually generated. Calibration lets you preview the connections between the 3D objects and the motion capture device, and then make adjustments to your channel connections or object transformations if necessary.

The **Channel → Calibration → Calibrate** command is designed for interactively calibrating transformations (translation, rotation and scaling) from motion-captured animation data.

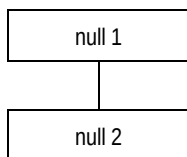
Most of the animatable parameters can be adequately calibrated for a channel capture using the **Offset** and **Scaling** settings in the Connection Setup dialogue box. However, except in some rare cases, you cannot calibrate rotations and scaling in this manner. For calibrating position, rotation, and scaling values, you can get much better control using the calibration method.

The calibration method uses a parent null to receive the animation data, and uses its child null as an intermediary to transfer the data to a constrained model. This may seem somewhat complicated, but it has certain benefits:

- The original captured animation data is preserved. You can capture very realistic human motion and apply the motion to objects without having to edit the original function curves. The original motion data is captured onto the parent null and is unmodified, so it can be applied to other characters using a different set of calibration nodes. In contrast, when using offset, scaling, or channel expressions, the original unmodified motion data is lost. Editing the transformation of the child null and then constraining your model to the child null is easier than editing the function curve of the captured animation data to fit your models.
- The size and proportions of the model are preserved. You don't have to scale or rebuild your model to fit the captured animation: you simply edit the position, scaling, and transformation of the child null and constrain your model to the child null.

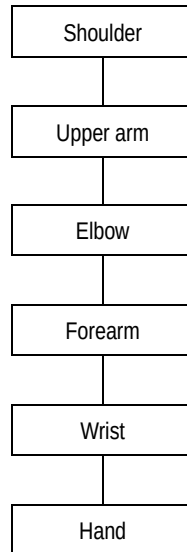
For example, suppose you want to capture the movement of a human arm and apply it to a 3D model of a robotic arm, using a motion capture device like the Ascension Flock of Birds or the Polhemus system. This is the procedure you would follow:

1. The Polhemus system and the Ascension flock of birds consist of series of sensors that are fastened to a live performer, and connected by wires to a processing device that interprets the position and rotation of the sensors as the person moves around. This motion data is then sent into SOFTIMAGE 3D via the Ascension or Polhemus channel driver. For the following steps, assume you have one sensor attached to the performer's wrist.
2. Create two nulls in your scene. Parent one null to the other using the **Parent** menu cell in the Model or Actor module. Your hierarchy should look like this in the Schematic window:



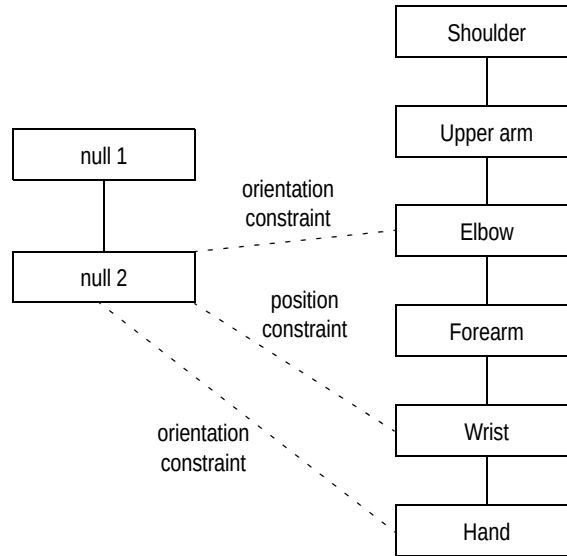
3. The parent null will receive the captured motion. Connect the channels from the sensor to the animatable parameters of the parent null in the Connection Setup dialogue box (for more information on making channel capture connections, see page 216). The child null is the calibration null: it automatically receives all the animation information from its parent. The robotic arm will then be constrained to the child null to receive the animation data.

4. Once you have made your channel assignments between the sensors and the parent null, load the model of your skeleton, arranged in a logical hierarchy. The Schematic window showing your hierarchy should look something like this:



5. Activate the calibration mode by choosing the **Channel → Calibrate** command. When you first enter the mode, all the channel drivers are running and the parented nulls are being updated by the data coming from the device. In other words, as the performer moves their hand, the nulls on the screen move accordingly. The middle mouse button is used to toggle between **pause** and **run**. When the drivers are paused the driver processes are still running but the data does not update the scene.
6. Before constraining the robotic arm model to the child null (calibration null), you must check that the range of motion of the child null is in proportion to the size and position of your model on the screen. With the **Channel → Calibrate** command active, compare the movement of the nulls to the position, orientation, and relative size of your model hierarchy. If the range of the captured motion is too great, or the position of the moving nulls is too far away from the model or improperly oriented toward your model, you can translate, rotate, or scale the child null to fit in with the rest of your scene.

7. Constrain the elements of the robotic arm to the child null (calibration null). In this case, you can use the following constraints: the wrist element is constrained to the child null with a **Position** constraint to match the position of the wrist sensor on the performer; the elbow and the hand can then be constrained to the child null with **Orientation** constraints. This way, the elbow and the hand are always oriented toward the wrist.



Resetting Offset and Scaling Values

Scaling and Offset values modify the incoming driver data in the following way: $[DOF = scale * rawDOF + offset]$

When you reset the offset and scaling, the **Offset** is set to 0 and the **Scaling** is set to 1. This means that one DOF equals one rawDOF.

Resetting Selected Objects

The **Channel → Calibration → Reset Selected** resets the offsets and scaling values of all the selected objects' channel connections back to their default values.

The **Offset** is reset to 0 and **Scaling** is reset to 1. A message is displayed in the status bar confirming that the operation has been completed.

Resetting All Objects

The **Channel → Calibration → Reset All** command resets the offsets and scaling values of all channel connections back to their default values.

The Offset is reset to 0 and Scaling is reset to 1. A message is displayed in the status bar confirming that the operation has been completed.

Note

More complex channel connections involving expressions are not reset when you choose the **Channel → Calibration → Reset Selected** or **Channel → Calibration → Reset All** command.

Controlling Animation and Devices

You can use channels to control or manipulate external devices, where animation data is output from SOFTIMAGE 3D and used to control an external device (such as an electronic puppet).

Types of Devices You Can Control

You can control the following types of devices:

- Servo-motor devices that drive an animatronics puppet.
- Force feedback through the mouse.
- Some audio drivers, such as using the soundOut driver to trigger short audio sequences.
- Motion control cameras.
- Midi devices.

Controlling Virtual Characters with Channel Connections

You can use channel connections to puppeteer virtual characters. For example, in this scene, the Ruby parrot character is controlled entirely with the mouse.

To load the Ruby scene, download it from the SOFTIMAGE 3D *On-Line Documentation* CD (#4), which contains the sample scene for Ruby the parrot. The scene is in the `/Anim_Guide` directory/`RUBY` subdirectory.

To try puppeteering Ruby, choose the **Select → All** command, then choose **Channel → Channel Run → Sequence I/O**. The channel run starts and you can click the left mouse button to record the motion. Click the right mouse button to end the channel run.

To view Ruby's channel connections, open the Schematic window to select the element that corresponds to one of Ruby's body parts and choose **Channel → Connection Setup**. Notice that different movements are created for each body part by changing the scaling and offsetting values of each connection.

Ruby's channel connections are as follows:

Bottom of Ruby's Body

Offset	Scaling	Source	Destination
0.50	-1.00	CHN mouse.xpos	FINAL_RUBY-BodyBot.etrnx
10.00	-20.00	CHN mouse.xpos	FINAL_RUBY-BodyBot.rotz
1.00	-1.00	CHN mouse.ypos	FINAL_RUBY-BodyBot.etrnz
10.00	-20.00	CHN mouse.ypos	FINAL_RUBY-BodyBot.rotx

Top of Ruby's Body

Offset	Scaling	Source	Destination
-20.00	70.00	CHN mouse.xpos	FINAL_RUBY-TopBot.rotz
-10.00	20.00	CHN mouse.ypos	FINAL_RUBY-TopBot.rotx

Ruby's Head

Offset	Scaling	Source	Destination
-10.00	20.00	CHN mouse.xpos	FINAL_RUBY-Head.rotz
-20.00	40.00	CHN mouse.xpos	FINAL_RUBY-Head.rotz
-20.00	40.00	CHN mouse.ypos	FINAL_RUBY-Head.rotx

Ruby's Left Wing

Offset	Scaling	Source	Destination
<i>Part A</i>			
0.00	80.00	CHN mouse.xpos	FINAL_RUBY-LWing_a.rotx
0.00	-80.00	CHN mouse.xpos	FINAL_RUBY-LWing_a.rotz
<i>Part B</i>			
10.00	-80.00	CHN mouse.ypos	FINAL_RUBY-LWing_b.rotz
<i>Part C</i>			
0.00	-20.00	CHN mouse.ypos	FINAL_RUBY-LWing_c.rotz

Offset	Scaling	Source	Destination
<i>Part D</i>			
0.00	-20.00	CHN mouse.ypos	FINAL_RUBY-LWing_d.rotz
<i>Part E</i>			
0.00	-20.00	CHN mouse.ypos	FINAL_RUBY-LWing_e.rotz
<i>Part F</i>			
0.00	-20.00	CHN mouse.ypos	FINAL_RUBY-LWing_f.rotz

Ruby's Right Wing

Offset	Scaling	Source	Destination
<i>Part A</i>			
0.00	80.00	CHN mouse.xpos	FINAL_RUBY-RWing_a.rotx
0.00	-80.00	CHN mouse.xpos	FINAL_RUBY-RWing_a.rotz
<i>Part B</i>			
10.00	-80.00	CHN mouse.ypos	FINAL_RUBY-RWing_b.rotz
<i>Part C</i>			
0.00	-20.00	CHN mouse.ypos	FINAL_RUBY-RWing_c.rotz
<i>Part D</i>			
0.00	-20.00	CHN mouse.ypos	FINAL_RUBY-RWing_d.rotz
<i>Part E</i>			
0.00	-20.00	CHN mouse.ypos	FINAL_RUBY-RWing_e.rotz
<i>Part F</i>			
0.00	-20.00	CHN mouse.ypos	FINAL_RUBY-RWing_f.rotz

Synchronizing your Animation with Sound

The Effects ➔ **Splound** custom effects command in the Motion module helps you synchronize the motion of an animated scene with an AIFF sound file. The Splound effect generates a linear curve that represents the amplitude curve of the sound file, which is then synchronized with the animation time line. When you click the playback button, the sound file is played back, with an indicator on the curve showing the amplitude of the audio at every frame. You can then manually edit the animation of objects in your scene, matching the amplitude of the sound with the movement of the objects.

For example, assume you want to animate a talking character. Once you have modified the character's face and mouth, you could use the Splound effect to generate the amplitude curve from a soundtrack recording of the character's voice. You could then keyframe the position of your character's mouth opening and closing at different frames to match the peaks and valleys of the amplitude curve displayed at those frames.

For more information on using this effect, see its description on page 632 of the *Reference Guide*.

CHAPTER THIRTEEN

SOFTIMAGE Live

Introduction

SOFTIMAGE Live is a standalone, run-time version of SOFTIMAGE 3D, optimized for viewing and animating scenes in real time. SOFTIMAGE Live is intended for use in real-time applications such as:

- Director-controlled computer graphics animation.
- Postproduction previewing.
- Virtual Studios and sets.
- Live on-air computer graphics characters.
- 3D Game prototyping and simulation.
- Run-time 3D playback.
- Virtual Reality (walkthroughs, pre-visualization).
- Interactive data visualization.

Some of these applications require Onyx or Onyx-like graphics capabilities.

SOFTIMAGE Live allows you to use the sophisticated animation and deformation tools in SOFTIMAGE 3D for prototyping (and authoring) interactive content directly, with no need for translation, exporting, or recompilation. SOFTIMAGE Live provides features such as:

- Standard viewing operations such as pan, zoom, orbit, and walkthrough.
- Different display modes (wireframe, shaded, textured, matte, and stereo).
- Playback of animation saved with scenes.
- Channel runs and take recording. SOFTIMAGE Live supports all the new channels features, including the different channel driver modes (Free Run, Frequency, Synchronized, and Remote).

There are two versions of the SOFTIMAGE Live executable: **liveGL** and **liveOGL**. The **liveGL** executable uses IRIS GL for rendering, while **liveOGL** uses OpenGL. Two versions are provided because **liveGL** performs better on certain SGI platforms and IRIX versions, while **liveOGL** performs better on others. For example, **liveOGL** runs 25 to 30 percent faster than **liveGL** on an SGI Impact.

Getting Started

To start SOFTIMAGE Live, you specify a resource path, a database name, and a scene name:

```
liveGL -R <resource_path> -D <database_name>  
-S <scene_name>
```

where

- **<resource_name>** is the resource path, which is typically `$SI_LOCATION/3D/rsrc`
- **<database_name>** is the name of a database listed in your `DatabaseDir.rsrc` file.

If you omit the `-D` option, SOFTIMAGE Live tries to load the scene from the first database listed in your `DatabaseDir.rsrc`.

To load a database that is not listed in your `Database.rsrc` file, include a full database path after `-D`.

- **<scene_name>** is the name of the scene you want to load. By default, SOFTIMAGE Live loads the most recent version of the scene.

SOFTIMAGE Live provides a single window with standard window elements (such as a title bar, resizable borders, and the system menu). When you resize the window, its contents are automatically adjusted to the new size, including a recalculation of the aspect ratio if necessary.

You can use the `-S` command line option to set the size and position of the SOFTIMAGE Live window. For example,

`-s "400x600+10+20"` opens a window 400 pixels wide and 600 pixels high, and positions it 10 pixels from the left side of the screen and 20 pixels down from the top.

What you see in SOFTIMAGE Live, in terms of how the objects are drawn, is identical to what you see in the Perspective window in SOFTIMAGE 3D.

Interacting with SOFTIMAGE Live

In SOFTIMAGE Live, you use keyboard shortcuts and Supra keys to perform operations. For example, to change the display mode, you press **s** to change to Shade view mode, **t** to change to Texture mode, and **w** to change to Wireframe mode. To pan the camera, you press **z** and use the left mouse button to drag the mouse cursor in the SOFTIMAGE Live window.

Getting Help

The command line option **-h** prints a description of the command line options, and of the keyboard shortcuts and Supra keys available during a viewing session.

To get help while running SOFTIMAGE Live, press the **?** key. This displays the usage message in the shell.

Viewing Scenes

Like the SOFTIMAGE 3D camera, the SOFTIMAGE Live camera is defined by its position, the position of its interest, and its field of view.

SOFTIMAGE Live supports a variety of viewing operations, such as panning, zooming, orbiting, and dollying. In SOFTIMAGE Live, you perform viewing operations the same way you do in SOFTIMAGE 3D using Supra keys and the mouse.

Panning the Scene

Panning allows you to move the camera laterally to see scene elements that are currently not in the field of view. To pan the camera, press the z key and use the left mouse button to drag the mouse cursor.

When you pan, the positions of both the camera and the camera interest are updated. They actually both move by the same 3D vector. The field of view does not change.

Zooming In and Out

Zooming increases or decreases the field of view angle of the camera so that scene elements appear larger or smaller. When you increase the field of view by zooming out, you can see more elements, but they appear smaller. When you decrease the field of view by zooming in, you can see fewer elements, but they appear larger.

To zoom in, press the z key and middle-click. To zoom out, press the z key and right-click. How far you zoom in or out is controlled by how long you hold down the mouse button down when you click it.

Zooming does not modify the positions of the camera or the camera interest.

Zooming is similar to the zoom operation of a real camera lens, so when you zoom out considerably, you see a distortion of the perspective (the fish-eye effect) which is not always desired. To avoid this distortion of perspective, dolly the camera instead of zooming.

Orbiting the Camera

Orbiting the camera rotates the camera about the camera interest, giving you a different view of the scene elements. For example, if a scene includes a 3D car and the camera is initially pointing at the front of the car, you can orbit the camera around the car to view its trunk, wheels, or roof.

To orbit the camera, press the **o** key and a mouse button and drag the mouse cursor. Use the left mouse button for a free orbit (vertical or horizontal), the middle mouse button for a vertical orbit, and the right mouse button for a horizontal orbit.

Orbiting modifies the position and orientation of the camera, but does not modify the camera interest.

You can also orbit the camera interest around the camera by pressing the **Shift-o** key combination instead of the **o** key. This corresponds to what happens to your eyes when you look left to right or down to up.

Dollying the Camera

Dollying the camera consists of moving the camera and camera interest with the same displacement along the line of sight. Like zooming, dollying makes elements appear larger or smaller. But unlike zooming, dollying does not change the field of view, so there is never any distortion of perspective.

To dolly the camera, press the **p** key and drag the mouse cursor. The left mouse button dollies slowly, the middle mouse button moderately, and the right mouse quickly.

Walkthrough

In walkthrough mode, you move the camera along the *xz* plane (the ground) to view different parts of a scene, just as if you were walking through the scene. Unlike the other viewing operations, a single walkthrough session can take a certain amount of time, depending on factors such as the size and complexity of the scene.

For example, you can use walkthrough mode to visit a 3D house. The session could start with the camera approaching the house, then you could walk around the house to view the exterior. Then you could enter by a door and explore the interior of the house.

To enter walkthrough mode, press **Shift-w** (you don't have to hold the keys down). Pressing **Shift-w** again exits walkthrough mode.

In walkthrough mode, you walk forward and backward, or left and right, by pressing the left mouse button and dragging the mouse cursor in the direction you want to move.

You use the right mouse button to raise the camera vertically above the ground level. You can also use the Ctrl-left mouse button combination to walk laterally.

You use the middle mouse button to orbit the camera interest as if you were looking up, down, left, or right while walking without changing your walking direction. The r key resets the camera position and interest so that you are looking in the same direction you are walking.

The field of view does not change unless you zoom in or out while in walkthrough mode.

Picking Scene Elements

Picking allows you to select elements in a scene. You can then frame the selected elements in the field of view. For example, if a scene includes many elements but only one is selected, pressing the f key adjusts the positions of the camera and camera interest so that the selected element occupies the entire view.

To pick an element, press the space bar and click the element. You can deselect the element by clicking it again (while pressing the space bar). Unlike in SOFTIMAGE 3D, there are no branch or tree selection modes. To pick an entire hierarchy, you must pick each element individually.

Framing Scene Elements

Framing adjusts the camera so that target elements acquire the field of view. The elements themselves are not moved or rotated.

SOFTIMAGE Live provides two ways to frame elements:

- Framing the scene. When you press the a key, the entire scene acquires the field of view.
- Framing picked elements. When you press the f key, the picked elements acquire the field of view.

Framing does not change the field of view (angle).

Controlling Scene Display

Setting the Display Mode

SOFTIMAGE Live has four different display modes:

- Wireframe (**w** key)
- Shade (**s** key)
- Texture (**t** key)
- Matte (**m** key).

The Wireframe and Shade display mode are the same as in SOFTIMAGE 3D. In Texture display mode, SOFTIMAGE Live displays an element using its texture. If the element has no texture, SOFTIMAGE Live displays the element in Shade mode.

In Matte display mode, SOFTIMAGE Live displays an element using a grey level based on the value of its matte attribute. A matte value of 1 is white, and a value of 0 is black. The background is black. The Matte display mode also supports the garbage matte feature. See *Using Mattes and Garbage Mattes* on page 260 for more on garbage mattes.

You can use the `-d mode` option to set the display mode from the command line (where *mode* is 1 for wireframe, 2 for shade, and 3 for texture.) You cannot set the Matte display mode from the command line.

Displaying Statistics

To display the refresh rate, press the **i** key. Press **Shift-i** for more detailed statistics.

Antialiasing

Normally, 3D lines including polygon edges appear aliased, which is a typical side effect of using a raster display (that is, using pixels). In the worst cases, a 3D line looks like a stairway. SOFTIMAGE Live includes provides two ways to deal with antialiasing problems:

- Hardware antialiasing. For high-end platforms (such as the SGI Reality Engine), the antialiasing can be done in real time by the hardware with a negligible speed penalty. By default, SOFTIMAGE Live automatically enables hardware antialiasing.

- Software antialiasing. SOFTIMAGE Live includes a software solution for the aliasing problem, with different levels of quality possible. The algorithm consists of multiple drawing passes, so you can expect a speed penalty.

To enable software antialiasing, you use the command line option `-a` which sets the antialiasing level. For example, `-a 0` gives you no antialiasing, and `-a 7` gives you the best (but slowest) antialiasing.

In SOFTIMAGE Live, you can toggle antialiasing on and off by pressing the `l` key.

Setting Surface Precision

Patch, NURBS, and face elements all have a Step attribute that defines a level of quality for surfaces computed from control points. A larger Step value gives a smoother surface, but it takes longer to draw the element.

SOFTIMAGE Live, like the Shade view in SOFTIMAGE 3D, allows you to assign a single, global value for the Step. This global value overrides the local Step value when an element is tessellated.

In SOFTIMAGE 3D, you can assign high Step values to elements to get high quality renderings. Then, in SOFTIMAGE Live, you can set a lower surface precision to speed up the display.

To set the global surface precision for SOFTIMAGE Live, you use the `-l` (lower-case “L”) command line option. For example, `-l 1` uses the lowest level of precision, `-l 2` uses the next highest level of precision, and so on. To use the local Step values, specify `-l 0`.

Using Mattes and Garbage Mattes

SOFTIMAGE Live supports a matte layer. If you have alpha bitplanes in your platform, the matte values of scene elements are drawn into the alpha planes. Together with a SIRIUS box and a video keyer, this allows you to perform real-time compositing of computer graphics and live video.

Use the Matte display mode (`m` key) to visualize the contents of the matte layer, which is reliably the same as the alpha bitplanes contents.

Displaying Sirius Video Textures

If you are connected to an SGI SIRIUS box, you can map video data onto textures of scene elements. The video data can be the output of a live camera or the playback of a video tape.

To do this, you must first set the texture type to SIRIUS in SOFTIMAGE 3D (using the Spreadsheet window). Then start SOFTIMAGE Live with the `-v` command line option.

For example, suppose you want to connect a real camera to the screen of a virtual television in a scene. First, assign a texture to the television screen in SOFTIMAGE 3D. Then open the Spreadsheet window and set the `t2sirius` attribute of that texture.

To view the scene in SOFTIMAGE Live, use the `-v` command line option to turn on video texture mapping. If you connect a real camera to a SGI SIRIUS box, the image as seen by the camera is reproduced on the screen of the virtual television.

SIRIUS video mapping usually degrades performance substantially, so you may want to use the `v` key to toggle video mapping on and off during a viewing session.

Displaying a Video-Alignment Cross

When you need to resolve video alignment problems, you can use a video-alignment cross to match the corners of a computer graphics image to a video image.

To display a video alignment cross, press the `x` key. Press the `x` key again to turn off this option.

Viewing Stereo Images

You can use SOFTIMAGE Live, together with external viewing hardware such as the CrystalEyes shutter glasses from StereoGraphics, to view stereo images.

Note

For stereo viewing, use `liveGL`. The `liveOGL` standalone does not support stereo viewing.

To run SOFTIMAGE Live in stereo mode, you must use the `-x` command line option. In stereo mode, SOFTIMAGE Live runs in a full-screen window and changes the video output format to full-screen stereo mode (see the manual pages for `setmon(1G)` or

`stereo(7)` for more information on the stereo video output format).

You can set the inter-ocular distance (the stereo separation) with the `-i <distance>` command line option. If you do not specify an inter-ocular distance, SOFTIMAGE Live uses a distance of 0.5 units. The best inter-ocular distance depends on the scale of your scene and the viewing distance. Use a smaller inter-eye separation setting to view objects from up close, or if the scale of your scene is very small. Use a wider separation for large scenes viewed from longer distances.

After you start SOFTIMAGE Live, you can use the `+` and `-` keys to interactively adjust the inter-ocular distance until the stereo display comes into focus.

When you exit SOFTIMAGE Live, it restores the original video output format. If, for some reason, it cannot restore the video output format, type the following command in a shell to restore the non-stereo output format:

```
/usr/gfx/setmon -n 60HZ
```

(If you prefer, use 72HZ.)

Animation and Channels

Playing Back an Animation

To play back the animation stored in a scene, press the right arrow key. SOFTIMAGE Live plays back the animation using the same start and end frames as were set in SOFTIMAGE 3D.

To display the refresh rate during playback, press **i** or **Shift-i**.

Performing Channel Runs

In SOFTIMAGE Live, you can use the channel connections stored in a scene to perform channel runs. Like SOFTIMAGE 3D, SOFTIMAGE Live uses the drivers defined in the scene and searches for them using the environment variable **SI_CHNLDRIVER**.

To start a channel run, press the **c** key. (This corresponds to the **Channel → Channel Run → Frame I/O** command in SOFTIMAGE 3D.)

To end the channel run, click the right mouse button in the SOFTIMAGE Live window.

During a channel run, you can use the Take Recording feature to save take data to disk. Later, you can load the take data into SOFTIMAGE 3D and generate high resolution (in terms of the number of key points) function curves.

To turn take recording on and off during a channel run, press **Shift-c**.

Recording Takes

You can use SOFTIMAGE Live to do long take runs or multiple Take Recording channel runs. This allows you to defer the creation of function curves until the post-production phase of your project.

You use command line options to do a Take Recording channel run in SOFTIMAGE Live. The channels options you set in SOFTIMAGE 3D are used during the take recording.

1. Set up your scene in SOFTIMAGE. This includes setting the channel drivers and their connections, and any other options.
2. Save the scene and then load it into the SOFTIMAGE Live. Use the **-T** option to specify the take directory and take name:

```
liveGL other_options -T /usr/camille,TestRun
```

3. Press **c** to start a channel run.
4. Press **Shift-c** to start a Take Recording channel run. Press **Shift-c** again to end the Take Recording.

Refer to page 233 for more information about Take Recording.

Parallel Processing

In normal single process mode, SOFTIMAGE Live runs in the following fashion (this example assumes a channel run):

1. Compute all animation and dependencies in the scene (not involving camera).
2. Compute all animation and dependencies required to update the camera.
3. Draw the scene.

On a multiple CPU platform, you can use the **-p 2** command line option to take advantage of parallel processing. This duplicates the scene so that each process works on its own version of the scene.

Process A

1. Update scene A based on the high-priority update list.
2. Draw scene A.
3. Repeat steps 1 and 2 until process B finishes updating scene B. Then switch to scene B and start again at step 1.

Process B

1. Update scene B based on the main update list.
2. Switch to scene A and start again at step 1.

The high-priority update list is a set of operations, usually not too many, required to update the camera based on factors such as animation, channel connections, and constraints. By updating the high-priority list and drawing the scene in the same process, SOFTIMAGE Live minimizes the latency between incoming channel data and their visual effect on the camera.

This is necessary in a virtual studio application where a real camera is connected to the virtual camera via channels. Otherwise, the motion of the virtual camera motion is uneven and distracting.

Disabling Animation Optimizations

To speed up the update and the display, SOFTIMAGE Live automatically performs some internal optimizations on the scene contents, the animation, and the channel connections.

These optimizations work correctly for most scenes, but in some cases they may introduce a delay of one frame during an animation. In such cases, you can use the command line option `-o` to disable all the optimizations.

Setting the Refresh Rate

Sometimes, the graphics load of the scene varies significantly over time, which causes the refresh rate to fluctuate in the same manner. For such scenes, you may want a more stable refresh rate, even if it means a lower rate.

You can use the command line option `-f` to increase the minimum swap buffers interval so that the refresh rate is locked to a multiple of the vertical retrace rate.

For example, if an application runs at 60 Hz 50% of the time and at 15 Hz the other 50% of the time, then `-f 4` locks the refresh rate at a stable 15Hz (assuming the vertical retrace rate is 60Hz).

Optimization Tips

Display Optimizations

- Use as few lights as possible. Infinite and Sun lights are the fastest lighting types, Point is slower, and Spot is the slowest. The fastest lighting environment (but the least realistic) is a single infinite or sun light.
- Make sure textures are as small as possible. Filling up texture memory on machines with hardware texture acceleration results in drastically reduced speed. However, texturing should be as fast as the shaded view on an Onyx, as long as the textures remain within the texture memory limit.
- If you have models with large number of polygons, performance may suffer. You can try to simplify those models using the **Effect → Cleanup** command or the Polygon Reduction custom effect (**Effect → PolygonReduction**).

You can also use the `-l` (lower-case “L”) command line option to lower the precision of all patch, NURBS, and face surfaces to reduce the number of triangles being drawn.

- The Constant shading model for materials is the fastest and the Lambert is the second fastest. Using one of these models may benefit performance.

Animation Optimizations

- Heavy expressions for channels or constraints can slow down a channel run in the viewer. For instance, using sine, cosine, square root, or logarithm functions can slow things down drastically.
- If you want to optimize the SOFTIMAGE Live refresh rate, try running channel drivers in Free Run or Frequency mode.
- Be careful about running fast, non-blocking drivers in Free Run mode. They consume too much processing time and reduce the refresh rate of SOFTIMAGE Live, especially if you have several Free Run drivers on a single processor machine. In this case, try Frequency mode instead.
- Serial drivers are good candidates for Free Run mode, especially if they are blocking on a serial access.
- Use Free Run or Frequency for channel drivers that access a device that has to be serviced at a certain rate.
- On a multi-processor machine, channel drivers can run in Free Run or Frequency mode with less risk of crowding out SOFTIMAGE Live.
- Run SOFTIMAGE Live in parallel on a multi-CPU machines as **root**. When SOFTIMAGE Live is run in parallel on a multi-CPU machine, it restricts and isolates the two processors used for display and animation update. The refresh and update rates are much more stable if you run SOFTIMAGE Live as **root**.

Creating a Virtual Set with Real and Computer-Generated Actors

You can synchronize live action and computer-generated (CG) animation in real-time to create a virtual set by using SOFTIMAGE channels and SOFTIMAGE Live. This involves creating a set and virtual characters in SOFTIMAGE 3D to interact with real actors that are positioned in front of a blue screen. Real people control the movements of the virtual character and the camera in real-time!

An actor is a real person that is in the field of view of the real camera. The resulting camera image is then combined with the computer generated image (scene and virtual characters). The walls, floor, and ceiling of the real set are usually painted chroma-key blue so that they do not appear in the final composited image.

Creating Characters in SOFTIMAGE 3D

For a Virtual theater project, you can create a scene in SOFTIMAGE 3D and assign matte values or matte function curves to objects. A matte value is updated in the alpha bitplanes when the hardware can support this feature, and is interpreted as a matte layering function.

The matte value of an object can have a value between 0 and 1, and the Matte view mode displays the matte values such that a value of 1 is white and a value of 0 is black. All the values in between 0 and 1 are shown as levels of grey in the matte view. See page 45 of the *Reference Guide* for information about Matte view.

When compositing a computer generated scene with live image, a SOFTIMAGE model with a matte value of 1 is superimposed on the live image. A model with a matte value of 0 is hidden by the live image.

Sometimes, you may need to animate the matte value of an object so that you can interactively place the CG actor in front or behind the live video. This is very useful for creating effects such as live actors entering through CG doors, or sitting behind a virtual desk. For example, you can connect the keyboard driver to control the matte value of an object.

Using a keyer, you can combine live actors and CG. SOFTIMAGE 3D allows you to set the matte values of objects, and you can use channel connections to toggle the matte value of an object.

In the following example, the Ruby parrot CG character is controlled entirely with the mouse.

To load the Ruby scene, download it from the SOFTIMAGE 3D *On-Line Documentation* CD (#4), which contains the sample scene for Ruby the parrot. The scene is in the `/Anim_Guide directory/RUBY` subdirectory.

Setting Up the Matte Keying with Channels

In SOFTIMAGE Live, using channel connections, you can interactively change the matte values that you give your objects. This allows you to interactively place CG characters in front or behind other objects or live actors. In the following steps (in SOFTIMAGE 3D), you are connecting the branch matte function curve for Ruby to the F1 key.

Steps to do this:

1. Open the Schematic window and select the root of Ruby.
2. Choose the **Channel → Channel Setup** command. The Channel Setup dialogue box appears.
3. Click the **New** button and double-click on keyboard to select the driver from the browser. Click **Ok** to exit the Channel Setup dialogue box.
4. Choose the **Channel → Connection Setup** command. The Connection Setup dialogue box appears.
5. Connect the Source and Destinations as follows:

CHN I keyboard.Key_F1 to RUBY_1.bmatte

When you go into the viewer and do a channel run, hold the F1 key down to toggle the branch matte value of Ruby. You can view the results of a matte animation by using the Matte view mode in SOFTIMAGE 3D and SOFTIMAGE Live.

Setting Matte Values for Background Set

Using the **Info → Selection** command, you can set the matte value of an object in a scene that does not need to be changed interactively. This allows you to place this object either in front of or behind a live actor or prop. For example, the objects that are the background part of your scene should have a static matte value of 0. You do not need

to be able to interactively change their values because they always remain in the background.

Note

You can also access the matte parameter from the Spreadsheet window.

Defining the Virtual Set

Before you can begin a channel run in SOFTIMAGE Live, you need to identify the real-world size and coordinates of your scene in SOFTIMAGE 3D. This allows you to match the position, orientation, and scale of your real actors in their computer-generated environment. For example, depending on the effect you want to create, you may or may not want your real actor to sit in a virtual chair that is only ten centimeters high!

Once you have created your scene and it is scaled properly to match the live actors, you can define the volume in your SOFTIMAGE 3D scene that matches the live space where the live actors and any other props will be located (blue screen set). SOFTIMAGE Live hides the rest of the real world that falls outside the blue screen area once you have defined a garbage matte area.

The purpose of the garbage matte is to replace any visible areas outside the blue screen set with the computer generated scene. This includes hiding things such as cables or other equipment (or the crew!), that would be displayed when a real camera is panning beyond the edges of the blue screen area.

1. To define the garbage matte in SOFTIMAGE 3D, create a model that represents the blue screen area. Typically, a single cube is sufficient to define the bounding volume.
2. Choose the **Info → Selection** command and activate the Garbage Matte option. The garbage matte area is actually the area that is outside of the garbage matte model (grids, in this case) that you have drawn.
3. To view the garbage matte area, activate the **Garbage Matte** option in the Shade view Setup dialogue box or select the Matte view mode. See page 34 of the *Reference Guide* for more information.

Note

The garbage matte attribute can also be accessed through the Spreadsheet window.

Setting Up Your Camera

To match the real-world camera viewing frustums with the SOFTIMAGE 3D camera, there are a number of parameters that are important to understand and to take into consideration. The most important parameters are the position and orientation of the camera and its field of view.

To characterize the field of view, or viewing frustum, SOFTIMAGE 3D uses the metaphor of a projection plane located some distance away from the focal centre of the camera. This projection plane is oriented perpendicularly to the viewing direction of the camera.

Using this model, there are three important parameters that define a simple symmetrical viewing volume. They are the x and y-size of the projection plane, and its perpendicular distance from the focal point. Two other features in SOFTIMAGE 3D support more complex, asymmetrical (or off-axis) viewing projections. These parameters are the horizontal and vertical offsets of the projection plane.

Changing the Camera's Projection Plane Parameters

You can change the camera's projection plane parameters to fix distortion problems in imported scenes, to match the parameters of the real camera to a virtual camera when compositing live action with a CG scene, or to simply to create strange effects in SOFTIMAGE 3D when you animate your scene.



You can access this option only from the Spreadsheet. The following steps outline how to set up your query to use the video texturing option.

1. Open the Spreadsheet window and click QUERY in the title bar. The Set Query dialogue box appears.
2. At the bottom of the dialogue box, in the **Edit Clause** area, set the following clause using the **Field**, **Oper.**, and **Value** options:

chtyp == CAM

3. Click **Accept**. Be sure that you are editing the main query, do not add another clause or a subquery.
4. Click the **Columns** button to open the Columns dialogue box, and expand the camera parameters in **Available Columns** by double-clicking on CAMERA.

5. Scroll to the bottom of the list to see the camera projection plane attributes. Highlight the following:
 - cmpdst (camera projection plane distance)
 - cmpsxx (camera projection plane size in x)
 - cmpszy (camera projection plane size in y)
 - cmpofx (camera projection plane offset in x)
 - cmpofy (camera projection plane offset in y)
 - cmpact (camera projection plane active)
6. Click **Add** to place them in the **Displayed Columns** list.
7. Click **Exit** to return to the Set Query dialogue box. If you want to save your query, see page 93 of the *Working with SOFTIMAGE 3D User's Guide* for more information.
8. Click **Ok** to run your query. The Spreadsheet displays columns for editing the projection plane values, and enabling and disabling the **cmpact** (camera projection plane active) parameter. Middle-click the box to toggle this parameter on and off.

Here's an example to illustrate how to change the camera parameters:

Say that one SOFTIMAGE unit equals one millimeter. For your 35 mm camera with a 50 mm lens, once you have activated the projection plane parameters, you would use the following values:

- Camera Projection Plane Distance: 50
- Size of X Projection Plane: 36
- Size of Y Projection Plane: 24
- Offset of X Projection Plane: 0
- Offset of Y Projection Plane: 0

Note

If the aspect ratio (picture format) of your camera is not set to 1.5 as it is in the previous example, the image will look squashed, stretched horizontally or vertically.

Connecting the SOFTIMAGE 3D Camera to the Hybrid Vision Tracking System

When you are using the Hybrid Vision tracking system, you need to reflect what's happening in its lens. Since the Hybrid Vision system uses an asymmetrical camera projection, you must use asymmetrical camera projection planes for the SOFTIMAGE 3D camera if it is driven by this camera. In other words, the projection plane is shifted laterally a bit so the left side or bottom side of the line of sight is not symmetrical to the right side or top side.

To connect the camera:

1. Select the camera in your scene.
2. Open the Spreadsheet window, and verify that the camera projection plane view is active. You do not need to change any other camera parameters unless you want to (see the previous section for steps on how to do this).
3. Choose the **Channel → Channel Setup** command. The Channel Setup dialogue box appears.
4. Click the **New** button and double-click on **hybridVision** to select the driver from the browser. Click **Ok** to exit the Channel Setup dialogue box.
5. Choose the **Channel → Connection Setup** command. The Connection Setup dialogue box appears. Connect Source and Destinations as follows:

Source	Destination
CHN hybridVision.pos_x	CAM Camera1.cmetnx
CHN hybridVision.pos_y	CAM Camera1.cmetny
CHN hybridVision.pos_z	CAM Camera1.cmetnz
CHN hybridVision.lookat_y	CAM Camera1.cmeitx
CHN hybridVision.lookat_y	CAM Camera1.cmeity
CHN hybridVision.lookat_z	CAM Camera1.cmeitz
CHN hybridVision.roll	CAM Camera1.cmroll
CHN hybridVision.fov	CAM Camera1.cmfov

Source	Destination
CHN hybridVision.ppdist	CAM Camera1.cmpdst
CHN hybridVision.ppsize_x	CAM Camera1.cmpsrx
CHN hybridVision.ppsize_y	CAM Camera1.cmpsy
CHN hybridVision.ppoff_x	CAM Camera1.cmpofx
CHN hybridVision.ppoff_y	CAM Camera1.cmpofy

6. Click **Ok** to exit the Channel Connection Setup dialogue box.
7. When you load the scene in SOFTIMAGE Live, the camera in the scene is controlled by the Hybrid Vision tracking system, allowing you to create an interactive virtual scene.

Creating a Video Texture

You can link a texture of an object to a live video signal if you have the hardware that is capable of doing this, such as SGI's Sirius box. You can map video onto any polygon mesh object using video texturing. To view video texturing of an object, you must set up your scene in SOFTIMAGE 3D and then use SOFTIMAGE Live to display the video texture.

Note

You can access this option only from the Spreadsheet window. The following steps outline how to set up your query to use the video texturing option.

1. Open the Spreadsheet window and click **QUERY** in the window title. The Set Query dialogue box appears.
2. At the bottom of the dialogue box, in the **Edit Clause** area, set the following clause using the **Field**, **Oper.**, and **Value** options:

chptyp == T2D

3. Click **Accept**. Be sure that you are editing the main query, and do not add another clause or a subquery.
4. Click the **Columns** button to open the Columns dialogue box, and expand the camera parameters in **Available Columns** by double-clicking on **TEXTURES2D**.
5. Scroll to the bottom of the list to see the **t2sirius** attribute. Select **t2sirius** and click **Add** to place in the **Displayed Columns** list.

6. Click **Exit** to return to the Query dialogue box. If you want to save your query, see page 93 of the *Working with SOFTIMAGE 3D User's Guide* for more information.
7. Click **Ok** to run your query. The Spreadsheet displays a column for enabling and disabling the **t2siruis** parameter. Middle-click on the box to toggle the parameter on and off.

To view the video texturing, load the scene into SOFTIMAGE Live. However, you need the following hardware and setup to be able to view the texture:

- Sirius box
- An input video stream connected to the breakout box
- Configure the video panel to select the input and the video format, etc.

See the Sirius Video documentation for more information about setting up the hardware.



This attribute does not affect the software rendering in any way.

Loading the Scene into SOFTIMAGE Live

Once you have finished setting up your scene in SOFTIMAGE 3D, and you have saved it with all its channel connections, you are ready to start SOFTIMAGE Live and load the scene.

To start SOFTIMAGE Live, you must specify a resource path, a database name, and the scene name. To start the viewer and load Ruby's scene, type the following at the command line (all one line):

```
liveGL -R /usr/softimage/rsrc -D WorkingDB  
-S RubySceneName
```

For more information about loading a scene, see page 256.

Doing a Channel Run in SOFTIMAGE Live

To experiment with puppeteering Ruby in SOFTIMAGE Live, press the **c** key to begin the channel run. When you do a channel run in SOFTIMAGE Live, all the channel connections in your scene are activated. To end the channel run, right-click in the viewer window.

Hardware Needed to Create a Virtual Application

- Sirius board
- Hybrid Vision camera tracking system
- Video Keyer (or mixer)
- Four CPU Onyx with RE 4 or RE 5 graphics
- Any number of devices to drive your animation, such as the Magellan™ spacemouse, a joystick, and other devices to capture motion of an actor for facial animation, body movements, etc. to be applied to a CG character.

CHAPTER FOURTEEN

Particles and Custom Effects

Introduction

You can create particle effects using the **Effects → Flock Animation** command in the Motion module. This allows you to create animation such as sparkles from a magic wand, or a fountain of water spraying up into the air. You can easily tweak the flock animation by changing the values that you enter into the Flock Animation dialogue box.

There are also a number of custom effects in SOFTIMAGE 3D that allow you to create particle-type effects.

Using the Particle System

Particle is a pixel-based particle generation system that is its own program (not in the SOFTIMAGE 3D interface). It lets you create three-dimensional animation that simulates natural phenomena containing many particles. For example, Particle makes it easy to animate sparks rising from a campfire, fireworks, or the tail of a comet trailing off into space.

Although Particle is a separate program, you import models created in SOFTIMAGE 3D into it. Then when you have created your particles, you can then composite the models and the particles together in one scene.

For more information about using Particle, see the *Particle User's Guide*.

Creating Particles with Flock

The **Effects → Flock Animation** command in the Motion module allows you to animate flocks of a particle model to achieve a variety of effects, such as fireworks, snowstorms, fuses, or sprinklers. Three user-defined mesh objects are used: a source mesh, a destination mesh, and a particle model.

By default, flock animation is a shape animation and the particle must be a polygon mesh. You can, however, animate your flock by transformations. This allows you to use a patch, meta-clay element, or NURBs-based model as your particle.

Animating with Flock

1. Create three objects to be used as the source polygon mesh, destination polygon mesh, and particle model, which must be a polygon mesh.
2. Choose the **Effects → Flock Animation** command.

The Flock Animation dialogue box appears.

Flock Animation

Name:

Number of Particles: ☐ Source Vertices

Number of Key Shapes: ☐ Destination Vertices

Source Seed: Start Frame:

Destination Seed: Stop Frame:

Gravity: Repetitions:

Up Vector: x: Frame Offset:

y: Seed Offset:

z: Frames To Hide:

Jittering

☐ Scale ☐ Rotate

x: x:

y: y:

z: z:

☐ Trans ☐ Shape

x: x:

y: y:

z: z:

☐ Hide Before Start

☐ Cyclic Extrapolation

☒ Sample By Area

☐ Track Destination

☐ BackTrack

☐ Interpolate Materials

☒ BackTrack

☐ Animate by Transforms

☐ Fixed Orientation

3. Set the desired Flock Animation parameters. For the list of all parameters, see the **Effects → Flock Animation** command description on page 530 in the *Reference Guide*. For an example of using these parameters, see the next section.
4. Click Ok to accept the settings and close the dialogue box.

You are prompted to select the source mesh, the destination mesh, and the particle model, in order, using the left mouse button. Now play your Flock Animation and view the results.

Note You can cancel your flock animation after selecting the source mesh or the particle model by right-clicking.

Creating a Tinkerbell Effect

To give you an idea of how to use the Flock command, here's an example for creating a comet-like “tinkerbell” effect:

1. Choose the **Get → Primitive → Cube** command to create the polygon mesh source object. Scale it down and freeze it using the **Effect → Freeze** command in the Model module.
2. Choose the **Get → Primitive → Grid** command to create a polygon mesh grid for the destination object. Duplicate the grid.
3. Choose the **Multi** menu cell and select all of the grids. Choose the **Effect → Merge** command and click Ok to accept the default values. The grids are merged together. Choose the **Effect → Freeze** command to freeze the merged grids.
4. Open the Schematic window, and delete the individual grids, leaving only the merged object and the cube.
5. Select the cube and choose the **Parent** menu cell. Make the merged object a child of the cube.
6. Choose **Get → Primitive → Square** and scale its size down to the size you want the particle to be.
7. Choose **Effect → Convert** and click Ok to convert the square into a polygon. Freeze the scaling of the particle using the **Effect → Freeze** command.
8. In the Motion module, choose the **Effects → Flock Animation** and set the parameters of the dialogue box as follows:

Number of Particles: 10	Seed Offset: 3
Number of Key Shapes: 4	Frames to Hide: 10
Source Seed: 40	Hide Before Start: selected
Destination Seed: 17	Cyclic Extrapolation: deselected

Gravity: 0	Sample By Area: deselected
Up Vector: y=1	Track Destination/BackTrack: deselected
Start Frame: 1	Interpolate Materials/BackTrack: selected
Stop Frame: 10	Animate By Transforms: deselected or selected
Repetitions: 5	Fixed Orientation: deselected
Frame Offset: 2	Jittering - Rotate: x=0.5, y=1, z=0.5

9. Click Ok to accept the changes. Select the objects in this order: the cube as the particle source, the merged grids as the destination, and your polygon “square” as the particle object. The cube actually travels behind the target mesh, but it works well for this effect.
10. Choose the **Multi** menu cell and select the particle (polygon square) and the merged grid. Choose the **Display → Hide → Toggle & Desel. Hidden** command to hide these objects.
11. Play back your scene to view the “tinkerbell” effect.

You can animate this effect using keyframing, shape deformation (of the particle), or path animation.

Note

Playing with the **Frames to Hide** parameter gives varying “trail” lengths, depending on the number of frames and length of path. A little static blur enhances the effect if you use lots of particles, and even the **mb** (motion blur) standalone may work well if you animate by transforms (just have lots of memory handy).

Creating Particles with Custom Effects

There are a number of custom effects in SOFTIMAGE 3D that allow you to create particle-type effects. For more information on using any of them, see their command descriptions in the *Reference Guide*.

Polygon Mesh Particles

The **Effects → Cometa** command in the Motion module acts a simple particle system. You create an animated source polygon mesh and then specify the number of particles to assign to that polygon mesh. The result is a new polygon mesh object composed of particles that is animated according to the source polygon mesh animation. Each particle is a triangle.

The **Effects → Cristal** command in the Motion module is an animated particle system based on polygons. It animates the polygon mesh objects or “particles” that were created using other custom effects, such as Nebula, Cometa, and Mosaic, or Breakup (see the *Reference Guide* for more information). The Cristal effect can be used to make the particles travel through space, strike other objects, bounce off obstacles, or react to wind effects. For example, you can animate particles falling and bouncing off the floor, striking other objects, or moving with the wind.

The **Get → Primitive → Nebula** command in the Model module lets you create effects such as falling snow, moving dust, exploding text, particles moving from the floor, and compositing a face or a text, etc. Each particle is a triangle and the resulting object will be a polygon mesh. You can create several models with the same number of particles and then interpolate them with a shape interpolation creating effects like fireworks, swarms, snow, etc.

The **Get → Primitive → Zzot** command in the Model module allows you to create effects like sparkles and lightning. Both source and destination models may be curves or polygon meshes, but not patch objects.

Exploding Polygon Mesh Objects

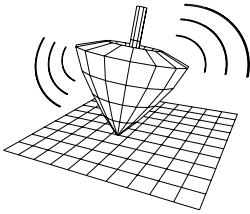
The **Effects → Explode** command in the Motion module allows you to explode a polygon mesh object into its constituent polygons. You can control and animate the magnitude, duration, and location of the explosion, as well as introduce randomness to create dramatic, realistic results. You can apply the effect to the entire object, or only

to polygons that have all vertices tagged. Used together with other effects such as Breakup, Equalize, Push, PolySkin, SplineTag, or SurfaceTag, you can create precisely the volcanic explosion or bomb blast you want!

Disappearing Polygons

The **Effects** → **PolyGrow** command in the Motion module allows you to make models appear or disappear one polygon at a time. Used in conjunction with the PolyOrder effect, you can easily create the appearance of an object materializing or dematerializing.

The **Effects** → **PolyShrink** command in the Motion module allows you to shrink individual polygons of a polygon mesh object toward the polygons' centers. Using PolyShrink, it is very easy to create an animation where the object materializes or dematerializes.



Applying Random Motion

The **Effects** → **Jitter** command in the Motion module allows you to apply random motion to a model. You can define the magnitude of the x, y, and z components for scaling, rotation, and translation, as well as a frequency parameter that determines how jittery the motion will be: slow and languorous like a feather falling to the ground, or fast and jerky like a molecule. All parameters can be animated.

The **Effects** → **ShapeJitter** command allows you to apply random motions to the control points of a model. You can specify the magnitude of the individual x, y, and z components, as well as the motion frequency to determine how “jittery” the resulting motion will be.

ShapeJitter is typically used to:

- Create random high frequency motion, such as bees buzzing about a hive, or small particles suspended in a liquid.
- Add the appearance of flexibility to parts of a character that are not deformed by the animator and look too “stiff.” These parts can be shape jittered so that they look more flexible.

Animating Custom Effects

Motion custom effects are similar to other effect types except that they are animated. This means that a Motion effect is reactivated every time the current frame is changed, whether it is changed manually by clicking on the time line or automatically during playback. An explode effect, for example, recomputes the geometry of an exploded object for each change in each frame number. This means that a Motion effect continues to have an impact on the scene even *after* you have finished using that effect.

Like all custom effects, Motion effects appear as regular menu cell commands and are distinguished from other SOFTIMAGE 3D menu cell commands by the “+” sign after their names (such as **Dissolve+**).

1. Choose the custom effect command (such as **Effects → Explode**).

The menu cell is highlighted and remains so until the custom effect is completed. The dialogue box associated with the effect is displayed.

2. Edit the dialogue box parameters as desired and exit the dialogue box.

The effect icon is displayed on the screen to indicate that the effect is active.

Note The icon is not visible in the Shade view mode or in any rendered image.

3. Pick the object(s) to which you want to apply the effect, according to the instructions displayed on the mouse line.
4. Manipulate the icon as desired.

You can manipulate the effect icon to alter the effect just like you would modify any SOFTIMAGE 3D object (animate it or deform it, for example). The design of the specific custom effect determines how it is able to use the attributes given to the icon in conjunction with the effect. For example, if the icon for an explosion effect is animated with translation and scaling, the effect will use the location of the icon to determine where the explosion originates, and its size to determine the magnitude of the effect.

The icon can also be used for the following operations:

- Delete the icon to delete a custom effect.

- Select the icon and then choose **Effects → Custom → Edit Parameters** in the Motion module to change the parameters of the effect in the appropriate dialogue box.



Unlike animation in SOFTIMAGE 3D, custom effects do not generate a function curve, so you cannot edit the effect's parameters in the Fcurve window. The interpolation between keyed values of custom parameters always uses spline interpolation.

Editing the Parameters

The **Effects → Custom → Edit Parameters** command allows you to edit the parameters of a selected effect.

1. Select the icon of the effect you want to edit.
2. Choose **Effects → Custom → Edit Parameters**. The appropriate dialogue box is displayed.
3. Make the desired modifications and exit the dialogue box.
4. The new parameters are applied to the effect.

Freezing a Custom Effect

You can freeze any custom effect in its current state. For example, an exploding mesh may be frozen at one particular frame and will not continue to be animated by the custom effect. In this way, all custom Motion effects may be used as modeling tools.

1. Select the icon of the effect you want to freeze.
2. Choose **Effects → Custom → Freeze Effect**.

The effect is frozen and the object reverts to its original geometry.



This command is not reversible: you must reapply the effect to the object.

If you have created a second effect on an object whose action is dependent on a first effect that you have frozen, the second effect's action is also frozen even though the second effect is still active on the object. For example, if you created a wave effect on an exploding mesh effect, then froze the exploding mesh effect, the *action* of the wave is also frozen. The wave effect by itself, however, is still considered to be active on the object.

Detaching and Reattaching a Custom Effect

The **Effects → Custom → Detach Effect** command allows you to temporarily detach an effect from the object to which it is applied. This is typically used while working on a scene to save computation time. You cannot save a detached effect with the scene: it will be automatically saved as if it was attached.

1. Select the icon of the effect you want to detach.
2. Choose **Effects → Custom → Detach Effect**.

The effect is detached from the object and a confirmation message is displayed.

To reattach an effect to an object:

1. Select the icon of the effect you want to re-attach.
2. Choose **Effects → Custom → Re-Attach Effect**.

The effect is reattached to the object and a confirmation message is displayed.

Note

This command can only reattach the effect to the object to which it was originally attached.

Saving Custom Effects

When a scene with an active custom motion effect is saved, all the data needed to reconstruct the effect is also saved with the icon. It is saved as a model in the **MODEL** chapter.

Note

A scene with the saved effect can easily be transferred to a different machine, but only if that machine also has the custom motion effect installed. If not, you won't be able to use that effect.

C H A P T E R F I F T E E N

Animation Files

Introduction

When you save an animated scene, SOFTIMAGE 3D also saves animation files that can then be associated to several different objects or model hierarchies. These files contain a list of function curves and a list of objects using the function curves. This lets you create a library of defined animations.

You can save animation files for all types of SOFTIMAGE 3D elements, including lights, camera, materials, and so on.

The use of animation files is best described by the following example: assume that you have created and animated a character, then saved the scene. You could then load the model and its animation separately (using the **Get → Element** command) and associate the animation to the model by setting a simple link in the schematic window. As this implies, you could create many different animations for your character, thereby creating a library of animations to choose from.

Conversely, as the animation file refers only to the name of the model to which it is linked, you can create a series of topologically different characters to use with your library of animation files.

In addition, you can also associate animation function curves selectively; that is, you can animate a character's legs using an animation file that contains animation function curves for both arms and legs.

Note

You must use explicit translations if you intend to associate the animation file to a model. Path translation is not currently compatible with this feature.

You can associate an animation file to another object very easily from the Schematic window when in the Motion mode. Animation files, called animation elements when in the Schematic window, are associated using the **FcrvCopy → Associate Anim./Disassociate Anim.** commands. You must read in an animation file directly from disk using the **Get → Element** command.

Refer to *Managing Files* on page 121 of the *Working with SOFTIMAGE 3D User's Guide* for more information.

Note

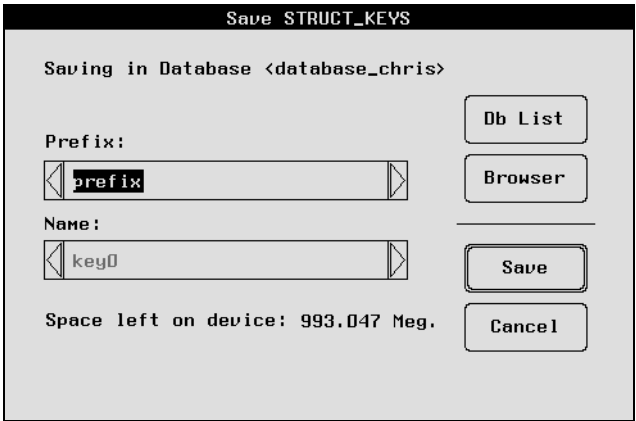
The **Get → From Library** command in the Motion module is another way of applying existing animation to a hierarchy – refer to page 840 of the *Reference Guide*.

Saving a Structure Key

Structure keys are saved animation values that can be stored in libraries as animation files. You can apply all or some of the information saved with a structure key to one or many objects with corresponding elements. Structure key values can be applied to hierarchies of models and individual elements of any kind, such as lights and materials.

Structure keys can be created using the **Save → Structure Key** command in the Motion module.

1. Select an object, element, or hierarchy for which you want to save a parameter setup.
2. Choose the **Save → Structure Key** command. The Save STRUCT_KEYS dialogue box appears:



3. Type a name for your structure key file (and a prefix if necessary) and click Save.

Note You don't need to be at a keyframe value for the selected element to save a structure key. The current parameters are saved regardless of whether keys or animation are present.

Retrieving a Structure Key

Once you have saved structure keys, you can load them and apply their values as follows:

- To a single selected element.
- To two or more elements selected within a single hierarchy.
- To one or many elements selected from many hierarchies.

You can retrieve a complete set of values or a character's posture from the library, and then apply these values to another character or to elements that you select.

1. Select an element or hierarchy.
2. Choose the **Get → From Library → Structure Key** command in the Motion module. The Load browser appears, listing the contents of the current `STRUCT_KEYS` chapter.
3. Select the desired structure key file and click **Load**.

The element or elements of the selected hierarchy are automatically set or positioned according to the values of the structure key. You can now save the position if desired by choosing a **SaveKey** command.

Retrieving Animation Function Curves

You can also apply or insert all animation function curves associated with the specified animation file to the selected elements. This allows you to apply several animation files to a single character.

1. Select an element or hierarchy.
2. Choose the **Get → From Library → Animation** command in the Motion module. The Load browser is displayed, listing the contents of the current `ANIMATION` chapter.
3. Select the animation file you want to apply and click **Load**. The Get Animation dialogue box is displayed.
4. In the **Insert at frame** text box, enter the frame number where you want the animation to begin.

The animation function curves are copied to all selected elements with corresponding names.

Setting a Track Filter

You can specify which tracks to copy from a structure key or animation file according to the type of element selected.

When you save a structure key, all track information, such as rotation, scaling, and explicit position, is automatically saved with the file.

Note You can select any element type, including lights, materials, waves, and so on.

1. Select an object or element.
2. Choose the **Get → From Library → Set Track Filter** command in the Motion module.

The Library Track Filter dialogue box is displayed.



3. Click on the tracks to deselect the ones that you do not want to copy; all tracks are selected by default (appearing as white text on a black background).

The filter setting is saved for each type of element selected for the duration of the work session.

Note

The **Get → From Library → Use Track Filter** command must be active for the filter to be applied. This option is active by default.

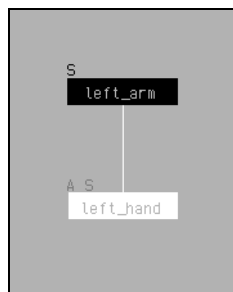
For more information about using Structure Key libraries, refer to the **Get → From Library** command description on page 840 of the *Reference Guide*.

Note

The **SI_keys_lib** database that is supplied with SOFTIMAGE 3D contains examples of structure keys that can be applied to models. A sample character model, a dog model, and a hand model are provided as a base for using structure keys.

Structure key values are mapped to the selected elements as follows:

- If you select a single object, then the values for the first element in the structure key hierarchy are applied. If the structure key specified contains only a single element, then all its values are applied to the selected object.
- If you select a hierarchy, such as a complete character, then the values for each structure key hierarchy element are mapped to the element with the corresponding name in the hierarchy you selected. For example, if you want to apply the values from a structure key you saved for a character called *ERNIE (prefix)* to the elements of the selected character called *BERT*, both ERNIE and BERT must have elements of the same name. The values for ERNIE's *left-arm* and *left-hand* will be applied to BERT's *left-arm* and *left-hand*, or any selected character with elements named *left-arm* and *left-hand*.



Polygon Info	
Prefix :	re jean
Name :	left_hand
Modifu :	SRTGH F

Note

Elements within a hierarchy are identified as part of that specific hierarchy by the prefix name. Refer to the **Info** menu cell commands starting on page 856 of the *Reference Guide*.

- You can also selectively apply structure key values. For example, you can select only a character's leg elements, and then only the leg element values from the selected structure key will be applied.

Notes

- You can apply structure key values to more than one selected object hierarchy or hierarchy elements at a time. For example, you can apply the structure key leg values from a dancer character to the leg elements of a whole chorus line at once.
- When you duplicate a hierarchy, only the prefix name changes, if there is one. If there is no prefix, the hierarchy name changes. For example, if you duplicate the ERNIE hierarchy, the new hierarchy will have a prefix called *ERNIE1* – the element names do not change. The elements that compose *ERNIE1* will have the same names as those in ERNIE. This makes it very easy to animate using library elements.
- You can specify which types of values will be retrieved from the structure key file (such as only rotations). For more information, see *Set Track Filter* on page 843 in the *Reference Guide*.

Index

Numerics

2D and 3D chains 131

A

Acceleration

plotting fcurves 85

Quick Stretch 109

Acclaim Skeleton animation files 239

Activating

channel items 224

constraints 196

DopeSheet track 56

Animation

articulated chains 129, 130, 133

channels 216, 229

clusters 91, 98

constraints 169

controlling 22

curve or surface deformation 105

custom effects 286

customizing 15

deformation

along a curve 107

along a surface 106

creating 106

lattices 100

DopeSheet 53

dynamics 147, 279

editing 29

editing fcurves 15

expressions 199

Fcurve window 30

files 239, 291

acclaimSkeleton 239

Biovision 239

motionAnalysis 240

forward kinematics 129, 133, 143

ghost window view 64

inverse kinematics 129, 133

keyframes 7

lattices 100, 101

creating 100

offsetting 24

optimizations, disabling 265

particle 279

custom effects 284

flock 280

paths 69

playing 20

in SOFTIMAGE Live 263

previewing 25

Quick Stretch 109

random motion 285

see also *Fcurves*

shape 91

stretching 24

trajectories 79

viewing 26

waves 119, 122

Antialiasing in SOFTIMAGE Live 259

Articulated chains 130

2D and 3D 131

animating 129, 133, 143

constraining 193

controlling the orientation 137

creating 132

critical zone 130, 138

effectors 131

forward kinematics 143

joints 131

changing the preferred angle 140

preferred axis constraint 138

roots 130

up vector constraint 139

using single joint chains 132

Associating, see also *Linking*

Attaching wave to an object 121

Auxillary fcurves 47

B

Biovision animation files 239
Branches, attaching wave to 121

C

Calibration
 captured channel data 241
 resetting keyframes 244, 245
Camera
 constraining 176
 projection planes 270
Capturing channel data 219, 229, 241
Chains, see *Articulated chains*
Channel drivers
 Free Running 266
 Frequency 266
Channels 199, 215
 activating items 224
 animating with 216
 animation files 239
 acclaimSkeleton 239
 Biovision 239
 motionAnalysis 240
 calibrating data 241
 capturing data 219, 229
 connections 216
 constraining 194
 controlling with fcurves 222
 copying fcurve setup 238
 device links 223
 drivers 199, 215
 expressions 211
 input 219
 input drivers 199, 215
 input/output 223
 matte keying 268
 output 222
 output drivers 199, 215
 parallel processing 264
 recording animation sequence 231, 232
 resetting
 calibration keyframes 244, 245

 setup values 224
 runs 263
 saving setup file 237
 Supra keys 224
 take recording 229, 233
Clearing fcurves from window 33
Clusters 91
 animation 91
 removing 99
 constraining 182
 cluster centres 183
 to objects 182
 creating 98
 deleting 98
 interpolation 92
 object constrained to 182
 saving keyframes 98
Collisions 159
 bbox obstacles 161
 bplane obstacles 161
Computer-generated characters 267
Connecting channels 216
Constant interpolation 33
Constant shading model 266
Constraints
 activating 196
 animating 169
 animation 169
 between points 185
 channel drivers 194
 cluster centres 183
 cluster to object 182
 clusters and objects 182
 custom 191
 cutting links 195
 deactivating 196
 direction 173
 tangent 175
 up vector 176
 expressions 191, 200, 203
 inverse kinematics 193
 joints 135
 multiple 170

- object centres 171
- object to cluster 182
- objects 169
- planes 180
- position limits 188
- removing 195
- rotation 172, 189
- scaling 179
- skeletons 137
 - preferred axis 138
 - up vector 139
- three-point 186
- Control icons, Quick Stretch 114
- Controlling
 - animation 22
 - channel data 229
 - channel device with another 223
 - channel devices with fcurves 222
- Controls, see *Physical forces*
- Converting explicit path to trajectory 80
- Copying
 - channels fcurve setup 238
 - expressions 202
- Critical zone 138
- Curves
 - animating deformation 105
 - deformation 105
 - paths for animation 69
 - trajectory 79
- Custom effects
 - animating 286
 - deformation 124
 - detaching 288
 - editing 287
 - freezing 287
 - particle 284
 - reattaching 288
 - saving 288
- Cutting constraint links 195
- Cycles of fcurves 43

D

- Deactivating
 - constraints 196
 - deformation 108
 - lattices 104
- Deformation
 - along a curve 107
 - along a surface 106
 - animating with curve or surface 105
 - animating with lattices 100, 101
 - by object's motion, see *Quick Stretch*
 - clusters 91, 98
 - creating 106
 - custom effects 124
 - deactivating 108
 - lattices 100
 - Quick Stretch 109
 - removing 108
 - resetting 108
 - shape 91
 - waves 119
- Deleting
 - fcurves 32
 - keyframes 61
- Detaching
 - custom effects 288
 - wave from object 121
- Direction
 - constraining 173
 - tangent, constraining 175
 - up vector, constraining 176
- Display mode, SOFTIMAGE Live 259
- Displaying fcurves 30
- DopeSheet window 53
 - deleting keyframes 61
 - editing 56
 - linking keyframes 62
 - segments
 - activating 58
 - copying 60
 - cutting 60
 - deactivating 58

- moving 59
- pasting 61
- playing 58
- scaling 58
- tracks 53
 - activating 56
 - filter 294
 - levels 54
- unlinking keyframes 63
- zooming 55
- Dynamics 147, 279
 - animating 148
 - collisions and obstacles 159
 - physical forces 154
 - physical properties 150, 156
 - real time collisions 161
 - simulations 158
- E**
- Ease-in and ease-out fcurves 13
- Editing
 - animation 29
 - Dopesheet window 53
 - fcurves 30
 - Spreadsheet window 65
 - custom effects 287
 - DopeSheet 56
 - fcurves 15, 30, 44
 - waves 122
- Effectors, articulated chains 131
- Elasticity, see *Physical properties*
- Explicit translations 9
 - plotting 81
- Expressions 266
 - acceleration curves 85
 - animating with 199
 - channels 211
 - constraints 200
 - contents 200
 - copying 202
 - creating custom constraints 191, 203
 - editing fcurves 204
 - fcurves 205

- editing 204
- plotting 81
- velocity curves 85
- External devices 199, 215
- Extrapolating fcurve values 42, 43

F

- Fans 154
- Fast playback 23
- Fcurve window 30
 - editing fcurves 44
 - editing key points 44
 - zooming 31
- Fcurves 12
 - auxillary 47
 - clearing from window 33
 - constant (CON) interpolation 33
 - converting 48
 - copying channel setup 238
 - cycles 43
 - deleting 32
 - deleting key points 44, 47
 - displaying 30
 - ease-in 13
 - ease-out 13
 - editing 30
 - key points 44
 - waves 122
 - expressions 204
 - extrapolating values 42, 43
 - freezing shape 43
 - grouping 32
 - inserting key points 44
 - interpolation 92
 - linear 34
 - plateau 34
 - slope zero 34
 - spline 34
 - linear (LIN) interpolation 34
 - locking key points 46
 - moving key points 44, 47
 - plateau (PLA) interpolation 34
 - plotting

- acceleration 85
- explicit translation 81
- expressions 81
- rotation 83
- velocities 85
- Quick Stretch 85
- repeated shapes 43
- retrieving animation 293
- scaling key points 47
- selecting 32
- setting key points on frames 45
- shape 42
- shape of slope 42
- slope zero (SL0) interpolation 34
- spline (SPL) interpolation 34
- tagging key points 46
- unlocking key points 46
- untagging key points 46
- Files
 - animation 239, 291
 - channel setup 237
- Flexing, see *Quick Stretch*
- Flipbook, viewing animation 26
- Flock animation 280
- Forces, physical, see *Physical forces*
- Forward kinematics 143
- Frames, ghost 64
- Framing scene elements, SOFTIMAGE Live 258
- Free Running channel drivers 266
- Freezing
 - custom effects 287
 - effect of wave 123
 - fcurve shape 43
- Frequency channel drivers 266
- Function curves, see *Fcurves*

G

- Garbage Mattes 260
- Ghost
 - frames 64
 - view mode 64
- Gravity 154
- Grouping fcurves 32

H

- Hardware, SOFTIMAGE Live 275
- Hierarchies, attaching wave to branch 121
- Hybrid Vision camera system 272

I

- Input channels 219, 223
- Interpolation 7
 - clusters 92
 - shape 92
- Inverse kinematics
 - constraining 193
 - invoking 129, 133

J

- Joints 131
 - constraining 135
 - friction, see *Physical properties*

K

- Key points
 - deleting 44, 47
 - freezing shape of fcurve 43
 - inserting 44
 - locking 46
 - moving 44, 47
 - scaling 47
 - setting on frames 45
 - shape of fcurve 42, 43
 - tagging and untagging 46
 - unlocking 46
- Key shapes 91
 - defining 93
 - moving between 103
- Keyframes 7
 - animating objects with 7
 - clusters 98
 - deleting 61
 - linking 62
 - resetting calibration 244, 245
 - unlinking 63
- Kinetic roughness, see *Physical properties*

L

- Lambert shading model 266
- Lattices 100
 - animation 100, 101
 - creating 100
 - deactivating 104
 - key shapes 103
 - removing 103
 - resetting shape 104
- Lights, SOFTIMAGE Live 265
- Linear interpolation 34
- Linking
 - cutting constraints 195
 - keyframes 62
 - object to trajectory 79
- Live video 273
- liveGL 253
 - starting 254
- liveOGL 253
- Loading take recording data 236

M

- Mass, see *Physical properties*
- Mathematical expressions, see *Expressions*
- Matte 260
 - background set values 268
 - display mode 259, 260
 - keying 268
- Motion
 - capturing 199, 215
 - input 199, 215
 - output 199, 215
 - take recording 229, 233
- Motion vectors
 - plotting Quick Stretch 115
 - see also *Quick Stretch control icons*
- MotionAnalysis animation files 240
- Moving DopeSheet segments 59

N

- Nails 154
- Nodes, attaching wave to 121

O

- Objects
 - animating clusters and shapes 91
 - animating with
 - channels 229
 - deformations 105
 - keyframes 7
 - lattices 100, 101
 - attaching wave to 121
 - constraining 182
 - centres 171
 - clusters to 182
 - to clusters 182
 - detaching wave from 121
 - linking to trajectory 79
 - plotting shape 84
- Obstacles 159
- Offsetting an animation 24
- Orientation, constraining 172
- Output channels 222, 223

P

- Parallel processing 264, 266
- Particle 279
 - custom effects 284
 - polygon mesh 284
 - system 279
- Paths
 - animation 69
 - assigning an object to 70
 - converting to trajectory 80
 - trajectories 79
- Physical forces
 - fan 154
 - force 154
 - gravity 154
 - nail 154
 - wind 154
- Physical properties
 - assigning 150
 - elasticity 150
 - joint friction 151

- kinetic roughness 150
- mass 150
- static roughness 150
- Picking scene elements, SOFTIMAGE Live 258
- Planes, constraining to 180
- Plateau interpolation 34
- Playback box 19
- Playing an animation 19
 - DopeSheet segments 58
 - fast playback 23
 - real time playback 22
 - time line 20
- Plotting
 - acceleration 85
 - animation 79
 - explicit translation fcurve 81
 - expressions 81
 - rotation fcurves 83
 - shape 84
 - trajectories 79
 - velocities 85
 - velocity 85
- Points
 - constraining 185
 - three-point constraint 186
- Polygon mesh, particle animation 284
- Position, constraint limits 188
- Preferred axis constraint 138
- Previewing
 - animation 25
 - Quick Stretch 113
- Profile, wave 123
- Projection planes 270

Q

- Quick Stretch 109
 - animated objects, see also *Quick Stretch*, *motion vectors*
 - anticipation 116
 - control icons 114
 - deformation modes
 - flexing 110

- stretching 110
- yielding 111
- editing 112, 113
- motion component 109
- motion vectors 115
- plotting velocity and acceleration 85
- setting up 110
- tricks 116
- twisting an object 117
- viewing 113

R

- Random motion 285
- Raw fcurves 48
- Real time
 - collisions 161
 - playback 22
- Reattaching custom effects 288
- Recording animation sequence, channels 231, 232
- Refresh rate
 - setting 265
 - SOFTIMAGE Live 259
- Removing
 - cluster animation 99
 - constraint 195
 - deformation 108
 - lattices 103
- Repeating fcurve shapes 43
- Resetting
 - calibration keyframes 244, 245
 - channel setup values 224
 - deformation 108
 - lattice shape 104
- Retrieving animation fcurves 293
- Roots, chain 130
 - see also *Articulated chains*
- Rotation
 - constraining 172, 189
 - plotting fcurves 83
- Roughness, physical property 150

S

Saving

- channel setup file 237
- cluster keys 98
- custom effects 288

Scaling

- constraining 179
- DopeSheet segments 58

Scenes

- loading in SOFTIMAGE Live 254, 274
- viewing in SOFTIMAGE Live 256 to 258

Segments

- activating 58
- copying 60
- cutting 60
- deactivating 58
- moving 59
- pasting 61
- scaling 58

Selecting fcurves 32

Sequences, recording with channels
data 231, 232

Shade display mode 259

Shading models 266

Shape animation 94

- defining key shapes 93
- interpolation 92

Shapes

- animation 91
- lattice 104
- plotting object 84

Shaping fcurves 42, 43

SI_CHNLDRIVER environment

variable 263

Simulation

- automatic 156
- deleting 161
- dynamics 158
- start and end 156

Sirius video textures 261

Skeletons 130

- animation files 239

constraining the orientation of joints 137

critical zone 138

Slope zero interpolation 34

Slope, see also *Fcurves* 42

SOFTIMAGE Live

- antialiasing 259
- background matte values 268
- camera setup 270
- channel runs 275
- computer-generated characters 267
- hardware requirements 275
- hybridVision camera system 272
- loading a scene 274
- matte keying 268
- shading models 266
- starting 254
- take recording 263
- video texturing 273
- virtual set 267

Spline interpolation 34

Spreadsheet window 65

Static roughness, see *Physical properties*

Stereo images, viewing 261

Stretching an animation 24

Stretching, see *Quick Stretch*

Structure keys 292

- animation 292
- library 292, 293
- retrieving 293

Supra keys, channels 224

Surfaces, deformation 105

T

Take recording 229, 233

- in SOFTIMAGE Live 263
- loading data 236

Texture display mode 259

Textures

- in SOFTIMAGE Live 265
- video 261

Time line 20

Tracks

- DopeSheet window 53

filter 294
Trajectories 79
 from an explicit path 80
 linking object to 79
Translation
 explicit 9
 plotting explicit fcurve 81

U

Unlinking keyframes 63
Up vector constraint 139, 176

V

Velocity
 plotting fcurves 85
 Quick Stretch 109
Video alignment 261
Video output format 262
Video textures 261, 273
Viewing animation 25, 26
Virtual Set
 background matte values 268
 camera setup 270
 computer-generated characters 267
 creating 267
 hybridVision camera system 272
 matte keying 268

W

Walkthrough mode, SOFTIMAGE Live 257
Waves
 animation 119, 122
 attaching to branch 121
 attaching to node 121
 changing profile 123
 creating 119
 detaching from object 121
 editing 122
 freezing effect 123
Wind 154
Windows
 DopeSheet 53

fast playback 23
Fcurve 30
Wireframe display mode 259

Y

Yielding, see *Quick Stretch*

Z

Zooming
 DopeSheet window 55
 Fcurve window 31

