Worley Laboratories
405 El Camino Real #121
Menlo Park, CA 94025 USA

Polk Manual

Worley Laboratories
405 El Camino Real #121
Menlo Park, CA 94025
USA
http://www.worley.com

Chapter 1 Introduction

The James K. Polk Plugin Collection (JKP for short) is designed to help extend the power of NewTek's LightWave 3D. JKP has been developed over a period of three years, in close association with the largest LightWave sites. This association has helped enormously in identifying the extensions most needed in LightWave Layout, and has resulted in the software you now own.

1.1 Installation

Polk works with LightWave Layout version 8.3 or later. All of the Polk tools are contained in the single **polk.p** file. The software uses about one megabyte of hard disk storage. Installation is the same for PC and Mac versions of LightWave.

You can always get the latest version of Polk from our web site. You should visit our downloads section online to make sure you have the "latest and greatest" version.

To install the plugin, copy the *polk.p* file into your LightWave plugin directory (you can pick any subdirectory that you like). Once copied into this directory, LightWave still needs to be told about the installation. First close all running copies of LightWave. Start a fresh copy of LightWave and use the "Layout/Plugins/Add-Plugins" chooser and use the file requester to select the *polk.p* file.

Licensing

Piracy is unfortunately a major problem with plugin tools. They are small, easy to copy, fun to play with, and are given less respect than a standalone application, which makes plugins attractive to "share." This is extremely illegal, but unfortunately not uncommon.

The impact of piracy is lower sales and higher prices. This isn't what we want; we want to give you under priced and overpowered tools!

The most practical way we can do this while still making some profit for future software releases is to protect the plugins from piracy. Polk uses the dongle you already have for LightWave as its method of protection. Polk is *not* tied to a specific PC, just a specific *dongle*. Some people install LightWave both at home and work, and just carry their dongle with them. There's nothing wrong with this, and Polk will have no problems.

The licensing won't affect your work after it is set up. Licenses for multiple dongles, usually at a studio, are also easily handled, even if the plugin file is being shared over a network.

After you install Polk, the plugin can load, save and render even if it has not been not licensed yet. However, the *interface for each Polk panel will not be accessible until the plugin is licensed.* These interfaces can be opened in several ways but the easiest panel to get to for initial licensing is via the "Pixel" panel. To reach this panel, use the "Window popup" menu on the top left side of your LightWave interface and select "Image Processing". Use "Add Pixel Plug-in" in this LightWave panel, and find VFog in the plugin list.

When you try to open the VFog interface (by double-clicking the "VFog" name) the plugin licensing panel will open. It tells you the version and date of Polk, and your "Machine ID," which is your internal dongle number.{There may be one or more numbers printed on your dongle itself, but this is not used by LightWave or plugins. I think these numbers can be used to predict lottery winners, though.}

You need a license code from Worley Laboratories to unlock the plugin interface. The license code will work only with your specific dongle, so we need to know the Machine ID number the plugin reports to create it. We also need your plugin serial number, which is printed on the back cover of this manual.

You can email, call, or FAX us to get your license code. The license screen has a button which should automatically start your mail program and compose email to us.{But it may not work on all systems, especially if you don't have a default mail program configured. In that case, you'll have write the email yourself. We just need your Machine ID and plugin serial number.} You need to add the serial number from the back of your manual and send it to us. We'll reply with the license code as quickly as possible; it should never be more than 24 hours.

If you have multiple plugin licenses, just repeat this procedure for each machine with a dongle. The plugin knows how to properly store multiple license codes even over a network.{If you have a *lot* of licenses (more than 5 or so) you can email us and we'll describe how to license all of your dongles at once by editing the stored license file directly. This will avoid having to manually visit each machine, which is difficult and annoying for larger studios!}

Once you get your license code, just type it into the license panel. After licensing, the protection is transparent. You won't see the requester again unless you install on a new PC.

ScreamerNet machines do not need to be licensed. You can use the plugin on any number of ScreamerNet nodes as you wish. These nodes do not need licensing at all.

ScreamerNet and Multiple Machines

If several computers share the same LightWave directory over a network, make sure all of them have exited LightWave before installing any plugins. If you don't, the other copies of LightWave will overwrite the **LW.cfg** file.(LightWave uses this file to store plugin configuration information.)

If multiple machines share the same plugin file (because you have purchased multiple licenses, or you are using ScreamerNet on a render farm), be careful that each machine has a properly configured **LW.cfg** file. This is often done by manually installing the software on each computer and ensuring that each machine has its own copy of LightWave, **LW.cfg** , and plugin files.

A better alternative is for a single master server to hold these files and configure each machine to access it through the network. In this case, make sure the pathnames of the plugin files are not machine-specific. This is often done by setting up a network drive letter (like **Z:**) to store the LightWave and plugin files. Every machine, including the host server, mounts this network drive as **Z:** and always refers to it with that leading drive prefix. An absolute path like **C:\newtek\plugins\Polk.p**, causes trouble because it's only valid for the server machine; the other machines don't necessarily have that file on their *own* **C:** drive and you'll get errors when they try looking there.

Network setup problems are especially hard to diagnose because ScreamerNet failures are difficult to analyze. ScreamerNet *silently ignores errors* such as plugins that it cannot find. If your Screamernet renders seem to be totally missing Polk effects, probably your plugin paths are incorrect and LightWave just can't find Polk. Network installation of LightWave itself is a difficult topic that NewTek's technical support department can help you with if you're having a problem configuring your systems.

1.2 Legal Stuff

In simple terms, the software may be used on a single computer by a single user at one time. The single exception is the use as part of a rendering network via ScreamerNet. Purchase of a single copy allows *unlimited* use as a ScreamerNet client.

Worley Laboratories makes no warranties, expressed or implied, with respect to the quality, performance, or fitness for any particular purpose of the enclosed software. In no event will Worley Laboratories be liable for direct or indirect damages due to use of this program. This software may not be rented, lent, or leased. You may not distribute any part of the software or its documentation to any party. It's common sense.

Each license of the software is linked to one specific LightWave dongle you specify by obtaining a license code from Worley Labs. Your license allows you to use the software only on the machine equipped with that dongle. Your license allows the use of only a single copy of the software at one time, unless you have purchased multiple licenses.

The use or installation of the software implies agreement with these terms. Enforcement of these restrictions is via the copyright laws of the United States and the State of California.{This legal stuff is really tiresome but necessary. You don't have to read it. Hmm, but you probably already did, since this footnote is at the end. Sorry!}

1.3 Updates and Bugs

Plugins are often affected by bugs or problems with LightWave itself. NewTek periodically releases LightWave updates to fix software bugs on its web site. Check NewTek's web site to ensure you have the latest LightWave version.

We release patches to our plugin software when we fix bugs or add new features. Our support section online contains the details.

We are admittedly perfectionists about bugs, and Polk has very few known problems. However, we have lists of LightWave problems which, in some cases, still affect Polk. We try to use workarounds for these problems in the program, and document other problems here.

If you see strange behavior, first check the FAQ section. If you do find a new Polk bug, *we want to hear about it immediately!*

Reporting Bugs

When you find a bug, first make sure it *is* a bug and not a common problem. Read the FAQ section of this manual, and also read our web site's Support section which has lists of known bugs and workarounds. Also make sure you have the latest versions of both Polk and LightWave to insure you're running the most modern versions.

Bugs that remain are easy to understand once they are isolated. If you do find a problem, start to *simplify* your scene to isolate it. This process can often teach you what the problem is, whether it's a software problem or not, and often whether there's an easy way around them.

Simplifying the problem is done step by step, just eliminating each extra layer of extraneous complication. Remove any objects that don't affect the problem. Remove extra lights if they don't matter. Turn off shadows if they don't matter. Take off surface textures. Change lights to simple point lights. Lower the rendering resolution to something fast like 320 by 200 as long as the problem is still visible. Turn off multithreading. Turn off reflection and transparency if possible. Remove unneeded plugins from the scene. Replace objects with a single polygon or at least remove extra geometry that's not needed. Try removing Polk and re-adding it to reset the values to Polk defaults. Try removing Polk altogether! (Sometimes the bug is inherent in LightWave, not Polk). The basic idea is to simply remove as much extra distraction as possible and really isolate the problem in its simplest case.

It's extremely likely that somewhere during this process you'll find a step that gets around the bug. This doesn't mean there's no problem, but it does give a valuable clue, such as if the problem only shows up when using both multithreading and area lights. These clues can often let you find a workaround (maybe as easy as turning off multithreading, for example). They also give *us* clues for where the problem lies and how to solve it.

If you've simplified the scene but the problem still remains, contact Worley Labs. If you have found a new bug, your simplified scene is probably very small and easy for us to test. This will help us fix any Polk problem quickly. If you've found a LightWave problem, it will help the LightWave programmers fix their error as well.

Chapter 2 LightWave and Plugins

Polk's interfaces are designed to be attractive and usable. The plugins share some styles of controls, including object pickers and animated numeric parameters. Since these common controls are used consistently by all of Polk's tools, they are important to understand.

Each plugin has an "About" screen accessible by clicking its title bar. This will tell you the exact version and release date of the plugin you're running. The about screen also prints your dongle's internal Machine ID number.

2.1 LightWave Plugins

A single file such as polk.p is sometimes referred to as a "plugin file," but really it's just a container holding the 23 Polk plugins. Try not to get confused when even our own documentation calls Polk a "plugin" when it's really a *collection* of plugins.

2.2 LightWave and Bugs

Plugins are often affected by bugs or problems with LightWave itself. NewTek periodically releases LightWave updates to fix software bugs on its web site. Check NewTek's web site to ensure you have the latest LightWave version.

We release patches to our plugin software when we fix bugs or add new features. Visit our support section online.

2.3 Item Picker

Many of the plugins have controls which refer to items in the scene. For example, the plugin *Dangle* asks you to pick an object to attach a rope or hose to. Polk uses an item picker which is more advanced than LightWave's, using its own popup panel which allows you to pick any object, light, bone, or camera.

When picking bones, the panel shows two lists. The left hand list chooses which object's bones to view, and the right list shows the bones themselves.

2.4 Saving Settings

Every plugin has a set of four buttons at the bottom of its panel. The simple "OK" button is used when you've finished changing the plugin settings and wish to return to LightWave. The "Cancel" button also returns you to LightWave, but *any changes you have made to your plugin settings will be discarded.*

Settings for any plugin can also be stored in a disk file for loading later. The "L" and "S" buttons at the bottom of each plugin interface allow you to load and save settings files. You can pick any name and extension for these settings files you wish.

You can save a *temporary* settings file to RAM by holding the "shift" key while pressing the "S" button. You can then paste this attribute snapshot into another plugin of the same type by pressing "shift-L." When an attribute is available for pasting, the "L" button will be highlighted to remind you.

2.5 Animated Numeric Parameters

Most of the controls in each plugin's interface are numeric and are very similar to the standard numeric gadgets you use in LightWave. The biggest difference is the way that these values may be animated.

Any numeric control with an "A" button can be animated. This button brings up an interface allowing many different ways to animate the parameter value over time.

The most common animation method is to assign the parameter to use a null object's key framed position as the control value. This allows you to use LightWave's powerful envelope editor to define any animated value you want.

Percentage controls are defined over the range of 0 to 1, not 0 to 100.

More than one control can be set by the same null object, which is useful when many objects are doing the same thing at once. You can choose the same null object in each plugin, and that single null will change the value in every plugin simultaneously.

A graph at the bottom of the screen shows the behavior of the animated value over time. You can change the plot's range with the two numeric controls below the graph.

You can quickly remove the animated control from the main plugin panel by shift-clicking the highlighted "A" button. This will revert the control back to an unanimated value.

2.6 Demonstration Scenes

Polk has a large collection of example scenes and objects that you can download from our website..The plugin descriptions in this manual describe the scenes. These demonstration scenes help show off specific features of the different tools to give you ideas of how you can use Sasquatch in your own work. You don't need to install or view the examples, but we recommend you do just to learn about the features.

The example scenes are compressed as a "zip" file. To install the demonstration scenes, extract the files using a "zip" decompression program. They can be installed into any directory.

Chapter 3 Motion Plugins

LightWave's motion plugins can be added to an object, light, bone, or even the camera.

One common problem when using any motion plugin is a slowdown in Layout's responsiveness. In most cases, the plugin is actually not very slow, but Layout may be evaluating it hundreds of times in order to draw the motion path of the selected item in Layout. If you lost significant responsiveness in Layout the first thing you should try doing is disabling the "Show Motion Paths" option in Layout's Motion Panel.

3.1 Blink

Blink is a motion plugin that automates actions that are periodic but have random speeds and delay between repetitions. Think of a person's eye blinking. The motion is simple: the eyelid goes down and back up in about a tenth of a second. This repeats itself about every ten seconds. The applications of this plugin go far beyond eye blinks, but I'll use that as an initial example.

An eye blink doesn't occur exactly every 10 seconds. Sometimes a person may wait 5 seconds to blink, sometimes 15; it just averages 10 seconds. The blink itself usually takes about a tenth of a second, but sometimes the actual blink may be a little slower.

The Blink plugin automates exactly this kind of periodic motion which has random pauses and cycle rates.

To use Blink, take your object and key frame a full motion cycle in LightWave. In the case of an eye blink, this cycle would be the motion of the eyelid moving down and the back up. Your can use as many frames as you like to define the motion; don't worry about keeping it at exactly 30 frame per second. the default cycle period Blink expects is 100 frames, which his convenient since you can thing of the frame number as a percentage of the time in the cycle. It's likely that you want the position at the end of the cycle (frame 100 in this case) to be the same as the position at the start (frame 0) to prevent any "jump" when the motion loops back to the beginning. After you have the cycle set up, add the Blink plugin to the object.

The first parameter in blink's interface, **Cycle Definition Length (frame)** is just the number of frames you used to define the cycle, which defaults to 100. It does not have anything to do with how fast the cycle is played!. This is just to tell the plugin how many frames you've used to set up the motion.

The next Blink parameter is **Repetition Period (sec)** . this is the number of seconds between each action. In the case of my eyes blinking, this is about 10 seconds.

Repetition Period Variation lets you randomize this repeat period. If I want to make my eye blinks spaced randomly from 7 to 10 seconds apart, I would us a 30% variation. A 5-10 second range would be a 50% variation, and 1 0-10 second range would be a 100% variation.

The **Cycle Duration (sec)** control defines how long the action takes. My eyes blink in about a tenth of a second, so I'd use 0.1 seconds. This can be randomized as well with the **Cycle Duration Variation** control. I could add a 10% duration variation to make an eye blink take between 0.09 to 0.10 seconds each time.

You can key frame the periods and variations over time. This allows you to use an animated envelope to make the blink rate faster or slower over time. This might be useful in a scene where a cloud of sand starts blowing around and you want the characters to start blinking more quickly.

Each object uses its own decision of what random variations to use by default. It would be wrong to have 10 people in your scene all blinking in absolute unison. Blink uses a random number seed that can be set differently for different times to make sure they do not share the correlation. This is the default behavior of Blink, as set by the **Unique** Seed button.

Sometimes you want to make two separate objects share their random behavior. In the eye-blink case, I have two eyes that blink together, and I want them to always activate simultaneously. I can make this happen by turning off "Unique Seed" and then setting any **Seed Value** I want (perhaps 11223344) and making sure each eyelid has the motion plugin with the seed value set to that same number. In my crowd of blinking people, each person would have a seed value that each of their eyelids share, but which is different for each person. The exact number does not matter; it is just a way of labeling the motion so you can mach up the random seed with other items. Giving a seed of 1 to the first person, 2 to the second, and 3 to the third is a good and logical way of specifying seeds; a set of number like 22562, 81, and 77779 would also work.

If you use a cycle duration longer than the repetition period, there is no pause between the cycles. This makes the cycles continuous but with a different rate for each sequential cycle. This might be applied to a cycle of a man chopping a piece of wood with an axe. You could make each swing of the ax have a lightly different period, but with no pauses in between.

You can restrict which of the nine channels the plugin affects with the toggle buttons under the **Applied Channels** heading. This can be useful for some simple effects where on motion channel is independent from the others and that's the only one you want automatically controlled. If you use this option it's usually best to have the cycle definition placed at some distant time offset by using **Cycle Start Frame #** If you don't do this, you'll often change your cycle key frames when you don't mean to.

Blink FAQ

Question: My item pauses after every cycle. What's happening?

Answer: this is the default behavior of Blink. If you want a continuous cycle, set the cycle duration to a large number like 9999 seconds. You'll get a continuous motion with different speeds for each repetition.

Blink first determines the timing for the next cycle repetition. It then computes the time to complete a cycle. If the cycle won't fit in the repetition "window" it shortens the cycle time until it does. This is why a large cycle duration like 9999 seconds makes the motion continuous.

Question: I've made a scene where my characters are blinking, but if I look closely, sometimes their eyes never close completely, even though I key framed them at the closed position.

Answer: This is actually a temporal aliasing effect. The full cycle is being performed, but the fully-closed-eyelid position may be happening at an in between frame of 7.5 instead of exactly at frame 7 or 8. If you use LightWave's motion blur, the in between motion will render properly.

If you are concerned about this kind of quick motion, you should increase your cycle duration to see the motion more clearly.

In the specific case of a blink, you may also want to key frame you blink to close the lids, pause a moment, then open. The pause would help insure that the blink is "full strength".

Question: I've tweaked my cycle an now the motion is all messed up. What's wrong?

Answer: You must disable Blink before making any changes to your cycle. If you don't, the key frames you set won't correspond with your original cycle key frames.

Possible Blink Applications

- A crowd of people, all clapping. This would be a lot like the axe chopping, since there would be no pause between claps.
- A ship-of-the-line's cannons could fire at irregular intervals.
- A gopher could pop his head out of his hole periodically.
- Granny's rocking chair. Her rocking isn't as exactly regular as clock.
- A flapper valve randomly bouncing open on the top of a diesel exhaust.
- An occasional chicken crossing the road. There's really only one chicken, but he starts out of view on one side of the road, and the cycle ends after he gets across the road (and again out of the camera's view.) When the cycle repeats, the chicken teleports back to the starting position, and crosses the road again. This makes the final motion look like chicken after chicken crossing the road. The random speeds and pauses make the motion look natural and unscripted.
- A water or oil drip, using the same teleport trick as the chicken example. It could also be applied to aquarium bubbles, balloons rising in the background, marathon runners crossing a bridge, or mice scuttling across the floor.
- Civil war soldiers loading, aiming, and firing their guns. Every soldier would be out of sync from his neighbors, so it would look much more natural.

- Breathing, by moving a bone in the character's diaphragm.
- Doors or shutters banging in the wind.
- To control other plugin parameters!. This might be a texture that flickers in strength over time, like a bad neon light. You just us a control null object, but attach Blink to the null to make it vary on and off.
- Add random jolts and jiggles to a vehicle traveling over bumpy terrain.
- A nervous twitch on a characters face.
- Random lightning.
- Random drumming on a wind-u toy drummer.
- Use with the Whip plugin to randomly move branches of a tree in the wind.
- A school of sharks circling their prey.
- Randomize motion of a carpenter.. swinging his hammer or cutting with his saw.
- Dribbling a basketball.
- A character rowing a boat, canoe or scull.

Blink Example Scenes

Blink-Eyes.lws A simple scene with three sets of eyelids. Each eyelid has Blink applied to it. All four eyelid objects(upper and lower eyelids of left an right eyes) share the same Seed Value to make sure they blink in unison. The three paris of eyes have different seed values. The JKP Track plugin is used on the pupils.

Blink-Bubbles.lws This scene shows how the teleport trick can be used to make an "infinite" number of continuously rising bubbles. Each bubble has a motion which is simply a rising motion. At the end of the motion (and out of view from the camera) the cycle ends. When the cycle begins again, it "teleports" to the start position at the bottom again, and begins to rise. Notice that the begriming and end positions are hidden from the camera view so the teleportation is not seen. Since the bubble's cycle is randomized in both rising speed and time between rises, the pattern of rising bubbles never repeats. In practice, the motion of the bubble could be any key framed motion you want. A complex object like a marathon runner could even have a fully key framed LightWave run cycle for each foot, and a "Blink" cycle that repeats only after 100 footsteps.

3.2 Limiter

Limiter is a very simple motion plugin which constrains the motion of an object to remain with certain limits. One example would be animating a drinking glass being placed onto a table. You want to make sure the glass never is set below a certain Y value since that would make it penetrate the table. While you could simply use LightWave's motion graph to manually fix any key frames that allow the glass to dip too low, the Limiter plugin can automatically clip the motion for you.

The Limiter plugin can also restrict rotations to lie only within certain angles, or scaling to lie within defined ranges. the limits themselves can be key framed to allow different limitations at different times.

You can define minimum and maximum values for each of LightWave's nine motion channels, and specify which of these clippings are active. For example, you may have some channels unconstrained, some which apply only a minimum value for a position, and some that apply a minimum and maximum joint angle.

Limiter Example Scene

Limiter-BreakOut.lws This scene shows a bouncing ball. As it moves to the right, it passes between two plates and the ball would naturally bounce right through them. the allowed Y range of the eyeball is set to constrain the bouncing to never pass through the obstacles.

3.3 Link Variants

The main design of these plugins allows a cycled object motion to be controlled in many advanced methods.

Cycles of motion are very common; they're simply repetitive motions that start from some defined pose, move in a certain path, and repeat. LightWave's built-in cycling forces a steady repetition of motion. The Link plugins allow much more control of speed, including stopping or even reversing the motion.

One common motion cycle is a "walk cycle", where a human figure takes one step after another. Each step is just like the previous one as the figure moves along. A bird flapping its wings is also a cycle, though it's much simpler. A machine gun firing also forms a cycle; the shell is fired, the chamber opens, a new shell loaded in, the chamber closes, and the cycle repeats. One of the simplest cycles is the hand of a clock circling at a regular rate.

Setting up a cycle is not hard in LightWave. It usually just involves key framing the object(s) through an entire cycle. If you use LightWave's "End Behavior" selection of "Repeats", the objects will repeat the same motions in an endless cycle.

The LInk plugins allow more control over these kinds of repeating cycles. What if you don't want the motion cycle to repeat endlessly, but want it to vary in speed? A bird may stop flapping to glide, but restart flapping after it has finished cruising. A machine gun may fire 10 rounds in a row, but pause a couple of seconds before firing a few more. Tank treads move in a cycle but they are intimately related to the tanks motion; if the tank slows, so do the treads. The treads may even go in reverse if the tank backs up.

The Link plugins allow all of these kinds of cycle manipulations; they allow you to link the cycle to many different types of external controllers. These controllers are what distinguish the five types of Link plugins.

PhaseLink is the most general of the Link plugins. It allows you to select where the cycle is by a key framed phase. Usually you'll control the state of the cycle by moving a null control object. One example of PhaseLink in use might be if you had a weird machine like a hydraulic elevator lift. The lift might have cylinders that extend, gears that turn, and moving levers. You could key frame the behavior of the elevator so that it rose from the lowest position up to its top extension, and define this as the cycle. After this cycle was defined, you could use PhaseLink to move the elevator in its cycle in any way you wished, usually be using a null object in the Motion Graph editor. You could key frame the elevator rising at different speeds, stopping at floors, or even reversing direction, and at all times the animation of the parts(the cylinders, the gears, the levers) would be properly performed, no matter how many times you want to raise and lower the elevator.

DistLink allows you to control a cycle based on the distance an object travels. This most useful for items like cars where you want the (cycle) tires to rotate based on the motion of the car. The motion can be much more complex than a rotation; it may be a fully key framed path like a plate in a tank tread moving around the full tread. This plugin could be used for wheels, tank tread, a steam locomotive's pistons and tie rods, or an odometer in a dashboard. It can also be used for walk cycles of creatures. One limitation is that the cycle's behavior doesn't change based on speed; a human will change gait when in a walk, jog or run. Am multiple-legged animal doesn't change it's walk much based on speed especially an insect or spider, so the DiskLink plugin works more effectively for these walks.

ThrottleLink is the most often used of the five Link plugins. It allows you to define the velocity of the repeating cycle. This could also be done with key framed phases and the PhaseLink plugin, but it is easier to define the speed at different times instead of the phase in most situations. This might be used on a machine where you just want to say "Stay slow until this frame, then stop, pause for this number of frames, then reverse, then go forward quickly." This could be applied to almost any machinery, perhaps the propeller of a ship that slows and stops as the ship sinks. It's easy to simply key frame a constant speed that then slows to stop. ThrottleLink does the actual determination of the phase at every frame.

SpeedLink isn't used very often but is very handy for some kinds of motions. It ties the cycle's phase to the velocity of an object. This could be used for a speedometer in a dashboard; the defined cycle is the needle rotating around the dial from 0 to the upper speed limit. As the control object (probably the car itself) moves faster, the needle moves farther along its cycle and indicates a faster speed. If the car slows or stops, the needle goes back to the beginning of its cycle, the 0 mark. This plugin could also be used in machinery cycles; perhaps a rotating governor on a steam engine could use a cycle where the balls get extended out more horizontally based on the speed of a control object. The speed that is used for controlling the phase might be distance base velocity, or a rotation one.

The **AccelLink** plugin is very similar to SpeedLink, except it keys based on object acceleration. This can be useful for rocket thrusters(which may have flames that grow in size as they accelerate), or for making a motorcycle do a wheelie as it accelerates.

The interfaces for each of the five plugins are similar. The basic procedure is to define the cycle of your object first, with no plugin applied. It is convenient to make the cycle range from frames 0 to 100 (since you can view the frame number as a percentage) but you can choose any range you wish. The cycle doesn't have to start at frame 0; sometimes it's convenient to "hide" the cycle in a distant frame range like frame 1000 to 1100. Tell the plugin the start of your cycle by setting the **Start of cycle definition** frame number and the **Cycle definition length**. For a cycle defined from frames 1000 to 1100, theses two controls would be set at 1000 and 100 respectively. The default is 100 frame cycle starting at frame 0, since this makes it easy to think for the frame number as a percentage.

The **End of cycle** buttons instruct the plugin what to do if the controller asks for frames outside the range of the cycle. For example, you may be using PhaseLink with a control null that asks for a cycle position of 60%. This sets your item to the position corresponding to 60% of the way through the cycle. But what happens if you ask for a phase of 100%. There are four options:

Repeat makes the cycle wrap around and repeat again and again. A 110% phase would wrap around and would result in a position 10% of the way into the cycle. 6522% would result in a position corresponding to 22%.

Ping-Pong makes the motion continuous by reversing the cycle after it reaches the end. After 100%, the effective cycle phase starts decreasing, so a 110% phase would result in a net phase of 90%. 210% would produce 10%.

Stop Clips the behavior. Any phase value above 100% is treated as being 100%. Phases lower than 0% are clipped to 0%.

Extend Allows the key frames from outside the defined cycle range to be used. If your cycle is defined from frames 0 to 100, a phase of 110% would use the key framed position from frame 110, even though that's outside the original cycle range.

PhaseLink

In the PhaseLink plugin, the cycle phase is controlled by the behavior of a control object. This object is selected in the top left of the interface, and may even be the object itself.

You can choose one of the control object's attributes (such as X position, or Bank angle) to define the applied cycle's phase. In the simplest (and most common) case, you might define the X position of a control null to range between 0.0 and 1.0 meters. When you animate the control null, a position of 0.75 meters in X would correspond to 75% of the way through the object's cycle.

The control object is usually a null object you edit via LightWave's motion graph. It is convenient to use the default **Control Value for Cycle Start** and **Control Value for Cycle End** values of 0 and 1, which let you view the cycle phase as as fraction, but you can change the range if you wish.

One example using PhaseLink might be setting up hydraulic pistons to extend when the elbow joint of a robot opens. You could use the elbow joint as the control object, and use a range of angles (say 20 to 70 degrees, if that's the range of the elbow's rotation as it opens) and have the pistons follow the joint's motions automatically.

DistLink

DistLink uses an object's net motion to determine a cycle's phase. This is extremely useful for vehicles or walking characters. As the objects advance forward, their wheels or legs can be animated automatically.

DisLink objects often use themselves as their own control item. A rolling wheel will use its own world motion to decide its rotation.. This is important if you want differential speed effect when car (or insect) goes around curves, since it's the different distances that each wheel travels that causes different rotation rates.

The effect of the motion is determined by one of the four distance options. **World Distance Traveled in +X Direction** is probably most common. If your wheel (or whatever) is moving in its local +X direction, this is the option you'd use. The direction is in local coordinates. A motion perpendicular to this direction will have no effect. For example, a wheel that is moving in its +X direction(with the +X distance option) would not rotate if it was skidded in its +Z or lifted in its +Y direction. The cycle will "unwind" if you move in the -X direction, so reversing wheels ar automatically handled.

If you choose **Total World Distance Traveled** the cycle will never reverse. It's like a car's odometer which always keeps increasing as you move.

Note that DistLink can do everything JKP's Wheelie can but is much more powerful. Wheelie is simpler and much more convenient if you just have the common case of a circular wheel.

The single control **Distance for one cycle** determines how far the control item needs to advance before one cycle is finished. You can use a negative value to reverse the cycle.

You can change the starting phase of the cycle with **Starting Cycle Phase**. This is especially useful to get a wheel or leg at the correct starting position at frame 0.

Unlike the other four Link plugins, DistLink has to perform extra computation to determine the entire history of the control item. This can make the plugin slower to compute. You can speed the computation by using the **Sampling Quality** control. Lower values make computation faster but less accurate.

ThrottleLink

ThrottleLink is the easiest of the Link plugins to use. Its phase is controlled by just tow numeric controls.

Phase Velocity %/sec is the speed at which the cycle proceeds. If you use a larger number, the cycle will repeat faster. A value of 0 will halt the cycle. This velocity is often animated by using the numeric "A" button.

Just like DistLink, you can change the starting phase of the cycle with **Starting Cycle Phase**

SpeeLink and AccelLink

These tow plugins are most similar to PhaseLink. Instead of the position or rotation of a control, Speedlink and AccelLink use the speed or acceleration of a control instead.

The velocity or acceleration is always measured in world coordinates. It usually uses only the veloctiy in a local direction, similar to DistLink's options. You can use an absolute acceleration or velocity if you wish.

The **Lookahead Frames** allows you to smooth the computed velocity or acceleration. This is important because LightWave's motion splines are very "ragged" in that their accelerations discontinuous. A larger number of lookaheads frames can smooth the velocity and acceleration effects to help hide these artifacts.

Link FAQ

Question: I've tweaked my cycle and now the motion is all wrong.

Answer: You should disable the plugin before making changes to the cycle.

Question: In DistLink, when would you use Starting Cycle Phase?

Answer: A good example is the DistLink-Spider demonstration scene, where the legs each share the same cycle. The Starting cycle Phase is et to 50% to force three of the legs to be in their proper start orientation.

Question: I've set up a car with a speedometer using SpeedLink. At frame 0, the speedometer is showing a velocity.

Answer: It's likely that the car is already moving at frame 0. You can change the object's first key frame's tension to 1.0 to make it start off at rest.

Question: I've got a scene with Motorcycle doing a wheelie using AccelLink. At frame 0, there's a lot of acceleration already.

Answer: As with the previous question, the motorcycle is probably accelerating even at frame 0. This sis a little harder to fix, but you can do it by adding another key frame near the start and using a "linear" spline to make the speed smooth and constant.

Question: I've got my penguin waddling at a 45 degree angle between the +X and +X axis. Which WOrld Distance Traveled direction do I us in DistLink?

Answer: It depends on the penguin. The orientation that the plugin is asking about is the penguin's, not the world's. If the penguin is modeled facing +X, you should use "World Distance Traveled in +X direction". The distance is always measured in world coordinates; the direction is always in local coordinates.

Some PhaseLink Applications

- When a space fighter gets to a certain distance from the mother ship, doors will open and then close when the space fighter is inside.
- Crossing gates open and close in response to a train traveling by.
- A drawbridge opens and closes as a ship passes underneath.
- A fighter throwing punches.
- Opening and closing a hand.
- A character using sign language.
- Control the height of a fork lift truck as it picks up pallets and puts them on different storage racks.
- Pincers on a crab or bug.
- Rudder, ailerons, and elevators on an airplane.
- A dog wagging its tail intermittently(could be controlled with JKP's Blink).
- The higher the weight lifter lifts the dumbbell, the redder his face gets, using JKP's Acid.

Some DistLink Applications

- Any complex character walking cycle.
- Tank treads.
- Animate the hand bar on a handcar as it runs down the tracks.
- Animating creatures whose motions cycles ar unchanged by speed, like fish or turtles.

Some ThrottleLink Applications

- Controlling windshield wipers
- Variable speed tools like a drill, router, lathe, or jackhammer.
- A drummer banging on his drum.
- An airplane stalling
- Controlling speed of variable speed tools and machinery.

Some SpeedLink Applications

- Speedometer needle
- Transmission gears automatically changing.

Some AccelLink Applications

- Motorcycle wheelie's.
- A car's suspension tilting forwards during a fast brake.
- Rocket thrusters firing
- As a control to make JKP's Wheelie plugin skid or spin its tires during accelerations.

Link Example Scenes

AccelLink-Wheelie.lws A car lifts its front wheels into the air when it accelerates quickly.

distLink-HandCar.lws A railroad handcar automatically "pumps" as it rolls down the track.

DistLink-Spider.lws A spider walks with varying speed. Each leg uses DistLink to keep the legs properly moving appropriately.

PhseLink-Butler.lws Two pairs of double doors use the position of a moving block to automatically open and close to allow the block to move through.

PhaseLink-Gears.lws The rotation of one gear automatically rotates the other gear, with no slipping.

SpeedLink-Metered.lws A giant speedometer is mounted on the top of a vehicle.

ThrottleLink-JackHammer.lwsA rapid jackhammer vibration is controlled with a single null.

ThrottleLink-Stall.lws Two airplane propellers slow and stop.

3.4 Parent

Parent is a motion plugin for performing dynamic, multiple parent handoffs. It also allows children to "divorce" independent motion channels from their parent.

Normally in LightWave an object is parented by LightWave's parent button and the child will inherit any motion of the parent. The Parent plugin can do this as well, but has the additional ability to act like a parent only over a defined range of frames, allowing the child to be independent during the rest. You can also swap parents, changing from one parent to the next at a certain frame number. This swapping does not make the object "teleport" from one parent to the next. Not every parent needs to be an item: you can leave the world as a "stand in" parent in any slot.

There are an enormous number of applications of this tool. A helicopter could be loaded with missiles, each attached to the helicopter with Parent. At a given frame number, the parent of a missile is switched from the helicopter to a null object that's moving forward very rapidly, making the missile "launch" itself. Before the launch , the missile follows the helicopter as it moves and rotates around, but after the launch, it ignores the helicopter and follows your "missile null". You could also just un parent the missile at that frame and use key frames.

Other applications are when a character picks up an object from a table and hands it to another. You want the object to stay in the grip of the character's hand, but transfer to the next hand at the appropriate time.

You can also parent objects but use the **Divorce Channels** option. This is useful for keeping an unchanged orientation. You might make a Ferris wheel with each seat parented to the wheel, but with the orientation "divorced" so it stays upright even as the wheel rotates. You could also use this for altitude indicators inside a cockpit. A gun turret could be the child of the gun barrel. You could divorce the pitch and bank of the turret from the barrel, but leave the heading. As the barrel rotates around, the turret would rotate as well. As the barrel pitches up, the turret would not rotate with it.

It is difficult to key frame the child object in Layout due to a bug in LightWave. It's safe to set key frames in the motion graph. You can also use the Parent feature on a null object, then use LightWave's own parenting to attach your real object to the null.

the object which has Parent applied to it shouldn't be parented to anything using LightWave's parent controls.

Parent handoffs are handled by specifying the **Duration** of each parent's possession. After the parent has "held" the child for that number of frames, the child is passed to the next parent in the list. The last slot has no duration associated with it. It is the final parent which lasts until the end of your animation.

You can make a repeating cycle of parenting with the **Repeat Parent Cycle** button. This is useful for a repetitive motion such as a juggling club attaching itself first to one hand and then to the other. It could also be used in something like an assembly line, where an object is "grabbed" first by one moving belt then the next. the process could repeat itself automatically using this option. Note that the repeated parenting cycle does not make a repeated motion; it's a repeated set of parent handoffs. A very simple parenting cycle could have a moving belt and a beer can. The cycle might be 10 frames parented to the belt and 10 frames with no parent. The beer can would advance for 10 frames, pause for 10 frames, and repeat.

Some Possible Parent Applications

- Use for holding and releasing objects.
- Baseball pitcher winding up and throwing.
- Basketball player shooting a foul shot.
- Dart players throwing darts.
- Jugglers
- Quarterback throwing a pass.
- People throwing a Frisbee.
- Jockey falling off a horse.
- A person bowling.
- Shoveling dirt or snow.
- A pole vaulter.
- An archer.
- Trapeze acrobats.
- Shark grabbing a swimmer.
- Catching a ball.
- Grabbing a weapon.
- An osprey taking a fish from the water.

Parent Example Scenes

Parent-Launch.lws An aircraft maneuvers for a brief period, then launches three missiles. The missiles follow the aircraft's motion until they are launched.

Parent-Exchange.lws A ball sticks to one rotating wheel until it's handed off to a neighboring wheel.

Parent-Catch.lws Two bending rods throw a ball back and forth to each other.

Parent-Snatch.lws One very bendy rod hands off a ball to the other.

3.5 SpeedLimit

SpeedLimit is a motion plugin which forces objects to move no faster than a given speed you specify. If you key frame a motion that includes a motion which is faster than this speed, SpeedLimit will slow the object down. If this happens, your object won't match your exact key frames anymore, but it will try, making the best possible speed without exceeding your limit.

Channel Speed Limit allows you to restrict the speed of just one channel. This could be useful for something like allowing any pitch and bank changes but restricting heading to change only very slowly.

Absolute Speed Limit is only for positional velocities. This stops an object from exceeding a give speed in any direction.

Some Possible SpeedLimit Applications

- A battleship turret can only change its heading a t certain maximum rate.
- An elevator can be prevented from dropping too quickly.
- a bank vault door swinging shut.
- Any big piece of machinery that can't react quickly.
- Drag racing cars.
- A skydiver hitting terminal velocity.

SpeedLimit Example Scene

SpeedLimit-DragRace.lws Three boxes race each other with the camera following. Each has an animated SpeedLimit control that changes their maximum speed, allowing the lead of the race to change several times.

3.6 Track

Track is a motion plugin designed to allow you to align the orientation of one object to face another. It is similar to LightWave's built in Targeting option for a light or camera, except you can apply it to any object.

The controls are very simple. You choose the item you want to target using the item picker.

You can add a **Time Delay** (or advance) to the track, so you point at where the object *was*. this allows a tracking lag for applications such as antiaircraft gunners which are little slow to follow the motion of the airplane overhead.

Tracking Strength allow you to cross-fade with the untracked orientation; maybe you only want to "half way" track something. It can also be used to activate or deactivate the plugin over time, allowing an item to start tracking something then return to its "rest" position after the target has passed out of range.

Having an object track the camera can be *very* useful for some effects, especially for 2D mapping of an image onto a flat plane. the plane tracks the camera to always keep facing it.

Track can use any of the six object axes to track with. When you use **+Z** or **-Z** you can additionally specify rotation limits. This can allow as object to rotate (for example) only in heading, leaving pitch unchanged. It can be useful for something like a gun turret that can only rotate or elevate a certain amount.

+X, -X, +Y,and -Y do not allow rotation limits since a mathematical annoyance called "gimbal lock" prevents an easy way to define the rotation limits.

Some Track Applications

- Track a null whose motion path creates text and you can write with a pen/pencil
- Paint with a brush
- Write with bullet holes in a piece of metal
- A character's head/eyes tracks a person walking by.
- An image sequence explosion polygon always faces the camera.
- A hydraulic cylinder in a landing gear or joint always points toward its partner sleeve.

Track Example Scenes

Track-TrackIt.lws A very simple scene showing a gun barrel following a moving target.

Track-Write.lws A tube writes out letters by shooting spheres at a paper. The pencil aligns itself to follow a guide null object to define the letters.

3.7 Wheelie

Wheelie is a simple motion plugin designed to quickly and simply animate wheels on cars. In its simplest form, you simply model your car with the wheels as separate objects parented to the car's axle or body. apply the Wheelie plugin to each tire and then animate the main car going down the street. Wheelie will automatically compute the motion of the car and rotate the wheel to keep it locked to the ground with no sliding. As the car accelerates, stops, or reverses, the wheels will automatically roll over the ground without faster than the inner wheels.

JKP's DistLink plugin can automate this kind of effect, but is more general in application. Wheelie is designed to be a "fire and forget" kind of tool which is quick, simple and foolproof.

The options in Wheelie ar deliberately simple. The main choice is one similar to **Forward Rotate Bank** You need to tell the wheel what rotation axis it spins on (which depends on which axis you modeled the wheel around.) You also have to specify whether the wheel spins forward or backwards; the wheel may be attached to the car facing the wrong way and therefore must spin in the opposite direction.

You need to specify the direction of motion which will cause the wheels to rotate using one of the three options similar to **Base on X Movement** The direction you specify is in the *wheel's coordinate system; which axis is pointed along the car's motion, assuming the wheel has not rotated. You can key frame the angle of the wheel over time or use LightWave's "Align To Path" option to make wheels which also steer.*

Wheelie uses the *world coordinate* motion of the wheel to determine its rotation. It ignores sideways motion (such as a skid) and only "counts" motion in the direction you specify. It understands forward and reverse motion, so the wheel stays properly tied to the pavement even if your car backs up.

Wheelie assumes the wheel is circular and that it rotates around its center. You will get poor results if you model a lumpy wheel or change the shape of the tire with a bone or displacement plugin. Wheelie ignores these perturbations and uses the base shape anyway. Wheelie *does* understand and pay attention to wheels that have been scaled. These are rare exceptions, but any cylinder-like wheel should work flawlessly.

The amount of rotation is based on the velocity of the center of the wheel. This matters when dealing with differential rotations, such as when you're going around a curve and the inside and outside wheels rotate at different rates. Keep the wheel modeled with the object center at the wheel's center (not at its edge, or off the edge) and you'll be fine.

The **Rotation Rate** control allows you to override the standard rotation rate of the wheel. This can be increased to values over 100% to make the wheels spring during an acceleration. it could also be reduced for a braking skid. The rate can be key framed to vary the rotation perturbation over time.

This plugin may cause annoying slowdowns in Layout if the wheel is selected as the active object. It needs to do a few milliseconds of computation per frame to determine the history of the wheel. If you have LightWave's "Show Motion Paths" option activated, LightWave will compute this for the entire scene every time you re-keyframe. If you have a long scene of 500 frames, it will compute the Wheelie effect 500 times, which may take 1/2 a second. This may be noticeable and annoying in Layout. Turn off "Show Motion Paths" in this case and it should be much better.

Wheelie Example Scenes

Wheelie-Turns.lws This simple scene shows a car making turns. If you watch the wheels carefully you'll see the wheels properly turn at different speeds as the car goes through a turn.

AcelLink-Wheelies.lws This example shows true wheelies. A car lifts its front wheels off the ground as it accelerates. Wheelie's Rotation Rate is key framed to drop to 0% when this happens to stop the front wheels from spinning when they're in the air.

3.8 Whip

The name "Whip refers to a whip antenna, not a whip like a bullwhip.

This sophisticated motion plugin makes objects react like a stiff spring, always wanting to be pointing in the same direction. The plugin actually a true physical simulation of the effect forces have on a stiff rod.

In its simplest use, you may use Whip to emulate a whip antenna. If you rotate the antenna's "bas" direction, the antenna tip will start accelerating to its new direction. If the antenna is loose and springy, the top will overshoot and bounce back and forth a few times before settling down into its new position. The speed of this attraction and how fast the motion damps out is user configurable.

The antenna whip also reacts to object motion. Put an antenna whip on a car, and have the car accelerate. The antenna will initially bend backwards, then bounce back and forth as it "catches up" with the rest position.

The whip can also react to gravity. With strong gravity settings, you end up getting a pendulum effect where the object wants to point downwards, and swings from side to side until it comes to rest at the bottom of its arc.

The whip can react to its velocity, bending it away from the direction of motion to simulate a constant wind pushing on it, sort of like having your hair be flown backwards in the wind when you're in a convertible.

This plugin doe a *lot* of math internally and therefore can take some computation time. Since it uses prior history (including examining the key framed acceleration of the whip object) to determine its behavior, it needs to "think back" to each previous frame. The means that the computation is slower for later frames.

You can speed up the plugin by modifying the **Simulation Quality** with the control of the same name in Whip's panel. Lower quality allows for faster computation. Low quality simulation increase the changes of "instability" in the simulation, which shows itself as irregular jitters in the whip's movement.

Whip is a motion plugin, and you just apply it to the object you wish to animate. Often that object will be the child of some parent, and Whip knows how to deal with this.

The whip's "natural" direction is in the +Y or *up*. The whip's spring restoration force will always try to move the whip to face in this direction, but of course it will be opposed by the other effects like *gravity* and *drag*. Note that this natural direction is in the +Y direction in the *object's local coordinates*. If you take a whip and rotate it 90 degrees to place onto a wall, the whip will "want to be" at that 90 degree heading.

It is common to parent the whip axis to a null object to rotate the "base" direction of the Whip axis. While you can do this by key framing the Whip axis itself, it can be confusing since your key frames aren't directly visible after the Whip motion superimposes itself onto them. You may wish to disable the plugin in its panel, key frame the whip's parent axis, then re-activate the whip effect after your basic setup is done.

Restoration Force determines how strong the spring is that brigs the whip back to its natural position. the stronger this value is, the faster the whip will bounce back and forth.

Gravity determines the amount of acceleration due to gravity. *Gravity always acts in the -Y direction in world coordinates*.

Damping changes the rate at which motions die out; how much friction or viscosity exists to slow the whip's motion down. Low values tend to make whips that go wild and bounce around madly. Extremely high values produce a viscous behavior. This is the most common attribute used to tweak the motion of a Whip object.

Wind Friction determines the amount of bend away from the world velocity. It's as if the air in the world is opposing the whip and is pushing it backwards.

Acceleration Sensitivity determines how much of a jolt the whip gets when it (or more likely, its parent) accelerates. You can use this to tune just how reactive the whip is.

Simulation Quality doesn't have to be changed often. If the motion of the whip becomes a little jerky, you may want to increase this value to see if the problem goes away. Larger values slow down computation. You can reduce the value to increase computation speed.

Angle Limit allows you to restrict the motion of the plugin to a limited range of angles. You may want to use a 50% limit to prevent the whip from ever rotating more than 90 degrees and penetrating your parent object.

You can limit the plugin to act in just the X, just the Z or both axis directions. This is especially useful for hinged objects that can't bend in all directions.

You can chain the Whip plugin, but it's not recommended. A chaining would involve parenting an axis with Whip applied to a *second* axis with Whip applied. This gives a cool effect where you get the second whip reacting with much more complexity than an individual whip. One useful example would be to have a tree with a large bone for the main trunk and a couple of child bones for the main branches. You could wiggle the main trunk to simulate an earthquake, and the trunk would wobble back and forth, and the branches would whip around even more.

Chaining isn't recommended because of computation times. When the motion plugins are chained, they take enormously longer to compute because of the way LightWave evaluates them. A 1000 frame animation of a single Whip plugin may take 30 seconds to build a wireframe preview, but two chained whips might take half an hour. Chaining them is admittedly fun, though. It can be useful to use the "Export Motion" button to make a key framed motion path for an object, so you can remove the whip plugin but still get the full effects. this also allows practical chaining.

Whip really like working with Flexor when bending objects.

Blink is a great tool to use with Whip to make the "goal direction" wiggle randomly with time.

Some Possible Whip Applications

- Spines on the back of a porcupine, or the big hear radiating fins along the spine of a dinosaur.
- Animation of many things like clothes by use of bones. As the character moves, the clothes will smoothly flex,. Interpenetrating can be limiting here.
- the flopping of a horse's mane as it gallops, using bones again.
- A car teetering on the edge of a cliff.
- Toys, toys, toys.
- Wings on a plane. (Ever watch your wings on a jet? the wobble about 5-6 feet at the tips!)
- Lots of small dynamic things, like a handle on a piece of luggage. As it's jostled around, itt may fly up for a moment before galling back down again. The would use the angle limit with gravity, but no damping or restore forces. Imagine access panels banging open and closed as a giant robot gets slammed with a missile or just walks violently.
- Dangling earrings on a character's head.
- Shutters on a house banging in the wind.
- Antennas...vehicles or bugs.
- Animate tree branches swaying in the wind.
- Streaming smoke from a train, truck or boat
- Animated diving board (using Flexor)
- Pendulum type motions...grandfather clock, metronomes, swings, tails, punching bags.
- Wobbling motions like a drunken human or a boat rocking in some waves.
- Clothes blowing in the wind on a clothesline.
- A hammock swinging between two trees.

Whip Example Scenes

Whip-WhipIt.lws An accelerating car has a vertical rod tilting backwards. As the car steadies itself, the rod restores itself towards the vertical.

Flexor-FlexIt.lws This simple scene shows Whip making a control null vibrate back and forth. This moves the control null making the rod bend back and forth, sort of like a ruler that's been "twanged".

Chapter 4 Displacement Plugins

LightWave's displacement plugins are applied in the Objects panel. JKP features four displacement plugins. Dangle, Diffuse, Flexor and Poke.

4.1 Dangle

Dangle is a displacement tool which automatically produces the hanging shape of ropes and chains, a curve call catenary.

Dangle is simple to use. Build a rope or chain-like object in Modeler. the object should have its origin at 0,0,0 and should extend in the +X direction. You can have parts of the object in the -X direction, but it's not recommended since the plugin won't animate then.

Load the object into Layout. You can position, scale, and rotate your rope object any way you wish. One end of the rope will be anchored at the object's position. The other end will anchor itself to a "goal" object you specify. This could be an object such as a hook, but usually a null object is most convenient. Add the Dangle plugin to the rope object. Dangle's interface really only has one control, allowing you to select this **Anchor Object**

In Layout, you'll see that the rope has "activated" itself, and now dangles in a smooth arc between the rope's position and the position of the anchor. The rope *does not stretch*, so the length remains constant even when you move the rope or its anchor. If the anchor object is too far away for the rope to reach, the rope will straighten out in its best attempt to reach the hook, but it will lease a gap.

Moving either the anchor or the rope object will cause the rope to interactively update itself and form the proper curve. There is no collision detection with the ground or other objects.

If the anchor and rope get very close to each other vertically, your essentially hangs vertically in a "U" shape beneath them. There is no provision for self-collision, so the rope will happily intersect itself if you bring it this close.

You can scale the rope. Scaling it in Y and Z will make the rope wider and thicker respectively. Scaling it in X will make the rope *longer* You can even animate this to simulate a cable being pulled in tighter.

Some Dangle Applications

- Ropes holding a boat to a pier. Imagine the boat rocking back and forth and the ropes tightening and relaxing atomically.
- A chandelier, with the "rope" being glass beads on a string. A dozen dangling strands would be easy to set up. You could tilt the chandelier and have the strands react appropriately. Add the Whip plugin to the chandelier and have an earthquake.
- Power cords! Imagine a battle robot that has a big 6 inch thick power cable attached to its back. Or a spiral phone cord as an animated character is picking up the phone receiver.
- Tower guide wires, especially when the tower is rocking back and forth.
- Bridges! Even as a modeling tool by using LightWave's "Save Transformed" option in Layout.
- Phone lines between poles.
- Chains.
- Handcuffs.
- Leashes.

Dangle Example Scenes

Dangle-DangleIt.lws A very simple scene with a single thick rod. Both ends are animated moving so you can see the catenary effect.

Dangle-Tug.lws Two rectangular blocks are key framed moving across the screen. They're connected with a thick rope with Dangle applied to it. The key frames of the blocks are designed so the separation between the blocks increase and decreased several times to give the impression one block is using the rope to tug the other.

4.2 Diffuse

Diffuse is a displacement plugin designed to work with clouds of points. It's not work use with models; it's a particle system effect that's designed for large groups of single points. Diffuse takes each particle and simulates a natural random gas diffusion, using what is know as a "random walk". this is very similar to the way silt will diffuse through water.

Every particle moves in a random direction every frame. This is not like an explosion where the points all move outwards from the center. It's a *diffusion* and points are just as likely to move back towards their origin as they are away from it.

The diffusion rate is the only control that Diffusion offers. Increase the **Diffusion Rate** values to make the points expand more quickly. You can reframed the expansion rates. This can be useful to make a puff of gas appear quickly but then slowly dissipate.

Over time, a cloud of diffusing points tends to get larger. This is not a linear expansion... if you measure the size of a cloud 10 seconds after you start diffusing, the will not double when compared to a measurement at 20 seconds.

This plugin really only has one use, for slowing expanding point clouds. It works well with LightWave's glow effect, and also can be used with LightWave's Volumetrics

Diffuse Example Scene

Diffuse-Silt.lws This scene shows a point cloud expanding over time. The object that contains the points is reframed to first rise and then fall, as if gravity were pulling the cloud slowly back.

4.3 Flexor

Flexor is a plugin designed mostly to be used with the Whip plugin. Flexor is a displacement plugin that allows bending of an object in response to the motion of another object, usually a control null. This allows springy objects like a diving board to easily be bent without using a chain of LightWave bones.

Floexor's setup is very similar to Dangle in that it assumes your object is built with its base at 0,0,0 and extending in the +X direction. This is not a limitation, since the object can be rotated (and bent) in any direction.

The degree of bending is determined by the angle between the object's X axis and a specified control null. You can exaggerate the bending amount by using the **Exaggerate** control. The bending object will become more sensitive to the control null's position with large exaggeration values.

The **Bending Center** control affects the shape of the curve. Low values tend to make a "bent elbow" shape and high values tend to make a more crooked shaped. You can get very bizarre shapes at extreme values like 10% or 90%. This parameter is usually left at 50%, which gives smoothest arcs.

This plugin is most often used with Whip. The control null is placed as a child of the Whip axis. The null sweeps an arc as the Whip object rotates back and forth. This moving arc acts as the bending control for the object.

Some Possible Flexor Applications

- Diving boards.
- Flexible antennae.
- Airplane wings bending under stress.
- A fishing pole.

Flexor Example Scene

Flexor-Flexit.lws This scene shows Whip making a control null vibrate back and forth. Flexor makes the rod bend back and forth, sort of like a ruler that's been "twanged".

4.4 Poke

Poke is inspired by the Newtek plugin called "Effector". It allows you to make indentations in an object by using another object as a simple tool. At its simplest, Poke allows you to make a bulge in the shape of a sphere appear in the side of an object. Used with more advanced control, it can automatically add footprints to the ground as a creature walks over it, apply dings from bullets and dents from baseball bats, and all kinds of different "dent" effects. Poke's original design was to simulate the motion of an object under a "skin" of material, like a rat crawling under a rug.

Poke works with one or more "effectors". The shape of an effector is usually a sphere, but you can stretch it or form different ellipse or pancake shapes. You can also change the shape from a spherical to more cubical effect by a curvature control. This is hard to visualize in Layout but invaluable when you need a more hard edged displacement.

The basic use of Poke is pretty simple. It's usually easiest to create an Custom Object in Layout to help visualize the effect. Create a null in Layout and add a Custom Object->Item Shape. Select the "Ball" shape and set the Scale to 2m.

Apply the Poke plugin to the object that needs to be deformed. You need to tell the plugin which object(s) are effectors by simply selecting them with the item requester.

With interactive Layout updates, you can see the effect on your object immediately. You can position the effector and stretch and scale it to cause dents of different shapes and positions. Change the **Curvature** of the effector to make non-spherical effects. Unfortunately the effector itself will still show the original shape which won't exactly correspond with the dent it's making, but it still shows the position and orientation.

Poke's **Sharpness** control changes how the dent behaves. If you reduce the sharpness, you'll tend to get softer effects, as if the effector only pushes points partially. Instead of making a dent in dough with a hard steel marble, it's like making a dent using another piece of dough.

The most useful ability of Poke is a history mode. You can animate an effector moving in, "poking" your geometry and moving away. The poked will stay deformed even after your effector has moved on. This allows you to do things like have your walking characters leave footprints in mud, or plow to leave a furrow behind it.

This history effect is easy to use. Select a positive number of **History Frames** to activate the effect. The Poke plugin will then look that far back in time to see the effect. You can use an outrageous number like 9999 to make the effect start at time 0 regardless of frame number.

The historical computation slows the plugin enough that it can be much slower. You can speed it up by selecting a lower **Curve Quality** Quality settings which are too low will cause a loss of detail when the effector follows a curved path.

You can make the displaced object respond as if it's made of rubber by using the **Restore** control. This makes any effector spots slowly "heal" over time. It's a cool effect, sort of like watching bread dough spring back slowly after you poke it with a finger.

The **Strength** control moderates the total amount of displacement Poke adds. A strength of 50% will only displace the surface half as much as the default 100%. This option does not work well when History is used.

The effectors are often parented to the object they're poking, so that the effect travels with the object as it moves and rotates. You can use as many effectors as you like simultaneously by picking more than one item in the item picker.

If you're using the history feature and you move an effector *through* an object like a sheet of paper, you'll get the cool effect of the paper bulging and following the effector even after it has passed through the plane of the paper. Be careful in these extreme cases since you're deforming the object so much that it's easy to run out of polygon "detail" since the vertices thing out as the skin is stretched. Use more polygons in the areas you expect to be heavily stretched.

Some Poke Applications

- Leaving footprints. Stretch an effector to be the same shape and size as your character's foot. Use multiple effectors grouped together to form toes. Parent them all to the character's foot. (Remember the effectors won't render). Build a flat plane with many subdivisions. Apply Poke to it. As your character walks over the flat plane, it will leave footprints behind him. Use the Restore control to have them slowly fill back up with ooze.
- Bullet holes. Key frame an effector appearing next to your object for a single frame and moving away. After a very short pause, have it reappear next to your object, offset from the first location. Repeat several times. Animate with history, and you'll see a string of dents appear. Alternately, use one effector per dent, and move them in one at a time and leave them pressed against the surface. This second method doesn't use the history frames, so it will be faster to compute.
- Organic ickyness. Show the movement of the aliens larva underneath the skin of your hero.
- A robotic arm which stretches beneath it's rubber sheath.

Poke Example Scene

Poke-Bulge.lws A rodent makes a moving bulge under a carpet.

Poke-Furrow.lws A plow leaves a trench behind its path.

Chapter 5 Shader Plugins

Shader plugins are applied in LightWave's Surfaces panel.

The word "shader" is a misnomer, since neither JKP's two shader plugins perform shading calculations. They are more properly termed *textures* since all they do is manipulate the surface attributes which LightWave then uses for shading. A plugin like G2 is a true shader.

5.1 Acid

Acid is a surface texture plugin. It allows you to place irregular texured spots on your object surface.

The basic idea is that you can use a separate control object known as an "effector" to place a splotch of color onto the surface of an object. The control object is often a null. The applied color splotch is a circular path, really just showing the intersection of the "sphere of influence" of the null with the surface of your object.

You can position , rotate, and scale the effector to make different patches at at different areas. This includes stretching the effector out into an ellipsoid to form long thin marks. The effector is usually parented to the object you're effecting, so as you move the base object it follows along. You can also use plugins like JKP's Parent to make this tracking easier, or even leave the effector parentless and keyframe it at will.

To help visualize the effect, you can add a Custom Object to the null. Use Item Shape->Ball and set the Scale property to 2m.

The spot of color the effector leaves behind can be modified with fractal noise to make ragged edges. The "Ragged" controls are easy to play with since you see the effect in the preview. When the **Strength** is 0%, the effect will render faster because no noise needs to be computed. The **Ragged Scale Size** makes the edge rags either fine streaks or lumpy blobs. The **Ragged Small Detail** changes the ratio of fine details to larger details. The **Ragged Number of Scales** allows more complexity in the ragged pattern.

The "Spot" controls are similar in that they use fractal noise to make the color spot irregular. Instead of making the edge streaky, this tends to add more of a random "dirt" effect to break up the mathematical perfection of the central part. The four controls are extremely similar to the "Ragged" controls, and with the preview are not hard to understand

The central preview draws white wherever the effect is added. This helps show the effect's behavior, especially with the Radius and Reverse options (described later).

The right hand section defines the effector objects themselves. You can use the item picker to choose one or more items to use as effectors.

Effectors allows you to pick one or more effectors which apply the effect to the surface. It uses JKP's object picker. You can select as many effectors as you wish.

Curvature allows you to make the efect non-spherical. A curvature of 0% makes the effector apply a cubical effect. 100% curvature makes a diamond-shape effect. 50% curvature (the default) is a sphere. You can see these effects in the preview. Remember that the effector can be stretched and rotated to make ellipsoids or rectangular blocks. Curvatures that are not exactly 50% will render more slowly.

Global Strength is something like an alpha channel. If your effector lays down red paint, you can set the Global Strength control to 50% and it will only tint the surface halfway to red, allowing some of the original surface to show through. Note tat this works *along with* the individual strength controls in each channel, discussed below. It's like a "ganged" control that modifies them all at once.

Global Radius allows you to change the effective radius of the effect without manually scaling the effector. This is useful when you just want a biffer splotch and you don't want to keyframe your effector's size channel again. It's also a "ganged" control, scaling the effect radius for all channels.

Use Undisplaced Coordinates is useful but non-intuitive option. Imagine you have a running man and you want to add a red splot on his hand. Your man is animated with bones or any other displacement tool. You would have to manually keyframe the effector following his hand through its movements. This is annoying, even with the use of plugins like JKP's Parent since these aren't exactly accurate in folowing the exact path of the polygons that the bone is displacing. The "Undisplaced Coordinates" option lets you place an effector in the place where the hand would be if there was *no* boning or displacement effect, and the plugin will use UV coordinates to "stay stuck" even as the surface gets moved via bones or displacements. Note that there may not even be a real surface where the effector is; it's in the location the surface *would* be if there was no boning! Artists have sometimes loaded two copies of their figure, setting one to be non-rendering (using the polygon size of 0% trick) with no bone action to give them visual feedback of where to place the effector.

This is a confusing option. It is still very important for the very few cases where you need to do this kind of texturing. It would simply look stupid and wrong if the woulnd bounced and jiggled because your tracking of the effector doesn't match the object's motion perfectly.

The lower part of the interface defines the attributes that are applied by the effector. The first two channels are easiest to understand; they are colors applied by the effector. The remaining channels apply other surface properties like diffuse and specular values.

Strength is an alpha channel. When it's at 100%, the "paint" it's applying will be fully opaque. It's like the airflow control on an airbrush. High strength means lots of paint, and it will coverup anything it effects.

Apply is the actual attribute you're applying. In the color channel, it's the color of the paint you're laying downl. the preview area will show the applied colors for the Color options, but the other channels (suc as Diffuse and Gloass) do not show the applied values. In these cases the preview efectively shows you the appplication map; where the applied material is being adde, not the final effect.

Sharpness determines the contrast of the edge of the effect. A high sharpness will make the edge very sharp. Low sharpness makes a smooth edge transition.

Radius lets you blow the size of the effect up or downl. Each channel has its own radius. Perhaps your blast scar applies a blackened circle, but the very center also needs to have the specular channel cur down to make it look like the metal is charred as well. by using a larger radius for the black color and a smaller radius for the specular channel, you can have the effects have different ranges.

Reverse lets you reverese the area of appllication of any channel. This leaves the center of the effect *unchanged* but applies the current attribute to everything outside the center. You could use this on a textured object to "reveal" a clear polished spot, but the rest of you object has obsuring paint over the rest of it.

Acid FAQ

Question: I don't understand the difference between "Strength" and "Apply" for attribute channels like diffuse and specular.

Answer: These channels are totally independent. A common mistake is to want to apply 100% transparency at the center of the effecotr to make a hole, so the user set the Strength to 100% under the "Transparency" row and renders. This doesn't work! the Strength channel defines how much of the attribute listed under the *next* column, "Apply," to add. So you really want to set two values: the Strength to 100% and the Apply value to 100%. It's pretty clear with the color channel, since it's like the airbrush strength and the paint color. The other channels work *exactly* the same, but it's easy to forget because both values are percentages.

Question: How do I choose which channel is displayed in the Preview window?

Answer: The fractal noise pattern varies all over space. You can never count on exact knobs and blobs you see in the preview, only the general type of appearance.

Question: I've used Transparency and Reverse on a surface but some of the surface is still visible.

Answer: There are two possible causes for this. One is that you're seeing specular or reflective effects. Another is that you're using fractal Breakup and there's "holes' in the applied efect due to the overlaid noise patterns.

Question: If I use any other value other than 50% ofr Curvature, my rendering slows down.

Answer: There is an internal optimization that works well for effectors with a circular curvature. Any other shape has to use a more complex algorith, and you pay the price in slower rendering.

Question: When I load previously used settings, the effector doesn't work.

Answer: Saved settings include a reference to the effector. If you load older settings into a new scene, you need to define the new effectors which would be applicable for this new scene.

Question: How do I animate size and shape with the History option? Animating the Global Radius doesn't seem to work.

Answer: Global Radius enlarges the area of effect even for attributes applied in the past. To make the size of the currently applied are change over time, animate the size and shape of the effector itself.

Some Possible Acid Applications

- On a human you can do cuts, wounds, bruises, sweat on clothes, burns, tears, and blushes.
- Tracks from vehicles, humans, or animals.
- Great for surfaces destroyed by fires and explosions.
- Cleaning and polishing dirty surfaces using Reverse.
- Animating blinkg lights on an airplane or control panel.
- Animating burning objecs like dynamite fuse, cigarette, or a burning log in the fireplace. Also for burnt ash and charred wood.
- Making terrain; Moonlike surfaces and bomb craters.
- Animating waves crashing on a sandy beach.
- Animated writing, simulating many materials. Paint, pencil, crayons, chalk, charcoal.
- Animating paint spill from cans, brushing on a piece of furniture, rolling paint onto a wall.
- Together with JKP's Diffuse plugin, spray paint graffiti on a wall.
- Hammer dents into a piece of wood.
- Great for adding dirt, grime and rust.
- Disgusting effects like pigeon poop and cow pies.
- Animating acid eating away metallic surfaces.
- Making webpage buttons.
- Adding mug to soldiers boots as the walk through swampy terrain.
- Make a surface gradually transparent; could be used for making transition wipes.
- Cleaning a dirty floor or window, using the Reverse application button to "remove" the dirt with an effector.
- Concrete, asphalt, and roadway grime and bumps.

Acid Example Scenes

Aicd-Buttonmaker.lws This scene renders a simple single still frame. It shows a simple flat plane with a single, centered large effector. This geometry makes it convenient to experiment with the effects possible using Acid's abilities. In particular, the Acid settings used in this scene use a reversed transparency application to make the background plane invisible outside the range of the effector. This mkaes a black background surrounding the central textured area. The rendered image and alpha chnanel is convenient for making Web page buttons.

Acid-Slotmaker.lws This simple scene shows Acid "carving" a slot into an object. The effector adds a central fully=transparent spot, making the surface disappear. A bump-mapped ridge surrounds the transparency. The history effect of Acid makes the moving effector leave a simple but dramatic groove in its wake. Try rendering s single frame near the end of the frame range to see the best view of the effect.

Acid-Stroke.lws This animation shows how the applied attributes can change over time to make dynamic effects. This example draws a thick shiny line of paint which quickly diffuses into the paper, leaving a broader, ragge, but flat painted area. All four attributes change smoothly over time and are contolled by animated parameters.

In the Acid interface, you can view the bump height's animation panel to see how the applied bump height decays over time, simulating the thick paint being absorbed int othe paper and flattening over time. The specularity is also animated to make the wet paint initially show a very glistening appearance but become fully matte by the end of the animation.

The color application is clever. Both color channels apply the same shade, but the first channel applies a smaller spot at full application strength. It is time-animated to fade in strength over time, decreasing the strength of the colored paint to simulate the paint's absorption into the paper. The seond color channel appplies a weaker but broader swath of color which does not fade.

The painted spot broadens over time to simulate the paint diffusing into the unpainted part of the paper. This is done by animating the global radius of the effector over time. This makes the applied spot size enlarge smoothly, even in the areas which have already been applied. If the animation were much longer this would start to cause problems because the newest paint would be as broad as the oldest paint, but this could be cured by having the effector itself scale downl insize over time. This problem is not apparent in this short animation so no scaling was done.

Since the effector's strok is a simple straight line, the history curve qualtiy is set very low so the texture renders very quickly. If the path were curved, this value would need to be higher and rendering would be slower.

5.2 HSVBoost

HSVBoost allows you to boost, reduce, or shift a surface color's hue, saturation or value. More importantly, these color attributes can be modified based on the angle of the surface to the camera.

This is not an effect that is needed very often, but for some real-life surfaces it can be very appropriate. One of the best examples are tropical fish, which have colorful skins that are covered by nearly-transparent scales. When you view the fish from the side, the brightly colored skins show through the outer scales and you see very colorful fish skin. However, if you view the fish at an angle, the outer tanslucent scales start adding a milky whiteness that robs the colors of their richness. The HSVBoost tool can emulate this effect by using a different boost of th esurface color's saturation based on the angle of the surface to the camera.

The HSVBoost tool has simpe ocntrols. You can define a **Perpendicular Saturation** and a **Glancing Saturation**. These are the boosts (or reductions) to the suface color when it's views at a "flat-on" or "glancing" angle respectively. If you have tropical fish with fully vibrant texturing, you might set the Perpendicular Saturation to 100% (to get all the vibrancy) but the glancing saturation to 20% (to dilute the color to a pastel shade).

The rapidity of this change can be controlled by the **Skew** and **Sharpness** controls. These allow you to shift the transition zone's angle and change the rapidity of th echange. The small preview sphere to the right of the controls helps demonstrate this change. This display can be thought of as a depiction of what boost would be applied to different parts of a sphere. A fully white pixel is judged as being "perpendicular" and a fully black pixel is "glancing". The transition between these is modified with the skew and sharpness controls, allowing you to see how the effect changes with angles.

Hue and brightness can be modified in the same way as saturation. In the case of hue, the color can be shifted around the color wheel to new shades. This can be useful for optical effects like a CD-ROM, which shifts colors based on angles.

I'm not sure why you would want to vary surface brightness base on viewing angle, but Worley Labs, in its ultimate generosity, allows you to do it anyway.

A final obscure control activates a special modulation of this too. By applying a texture to the *luminosity* channel of the object, the HSVBoos shader can apply itself more or less strongly based on that luminosity. the luminosity channel is set to 0.0 afterwards. This is an abuse of LightWave's surface texturing (luminosity is being "kidnapped" in order to pass information of HSVBoost) but this allows you to use fractal noise or image maps to break up the HSV effect. This can make the color shift look much more random and natural.

HSVBoost Example Scene

HSVBoost-Tilt.lws This scene shows five copies of a rectangular checkerboard box. The sides of the box are tilted to show different viewing angles. The applied HSVBoost plugin rotates the yellow color of the box towards green and also decrease the saturation in these areas.

Chapter 6 Pixel Plugins

Pixel plugins allow effects to be added to LightWave's imagery during each antialiasing pass. This distinguishes them from Image plugins that are applied to the final image LightWave has rendered. Pixel plugins are applied in LightWave's Effects panel.

JKP features two pixel plugins, Enviro and VFog.

6.1 Enviro

Enviro is a plugin designed to build 360 degree maps showing your entire scene from a central viewpoint. This can be useful for making environment maps for later use as reflection maps. It can also be used to make Quicktime VR (QTVR) and Microsoft Surround Video (MSSV) files, which allow a user on a web page to interactively pan around a 360 degree view of a scene.

Enviro is a pixel filter. It completely overwrites LightWave's rendered image. Enviro traces rays in all directions and puts the results into the output image.

Enviro supports several projection methods. All of them are based on the camera's position and orientation.

Spherical and **Cylindrical** are the most common projections. These are "classic" environment maps that show the full view around the camera. LightWave's reflection maps use a spherical map. The second type, cylindrical, has mathematical advantages and is used in QTVR and MSSV. The Cylindrical projection map doe the whole environment, like the spherical map, but has a different horizontal distortion. The "natural" aspect ratio for these images is 2:1, so rendering at a resolution similar to 640 by 320 is usually optimal.

Some renderers commonly use cubic environment maps. You can form out a **Cubic Map** one face at a time. It's usually best to render with a square resolution like 512 by 512.

the final option is an **Orthographic** view. This allows you to render a view with no perspective distortion, like the front, side, and top views in blueprints. You need to specify the width and height of the "projected" space. The projection is centered along the direction the camera is facing.

Generating high resolution maps can be slow, since every pixel is raytraced.

Since the environment is sampled by raytracing , many effects can't be included in Enviro's output. LightWave's lens flares, glow, background images, and the effect of other pixel and image plugins will not be show by Enviro. Background gradient colors *are* part of the world and will appear in Enviro's output.

A neat trick is to make your own interactive "VR" view in LightWave itself. Use Enviro to generate spherical map of any complex scene, then apply it to a LightWave sphere. Put the camera in the center of the sphere and activate LightWave's OpengGL view with texture maps. Rotate the camera around to see the different parts of your image.

Enviro Example Scene

Enviro-Room.lws A cubical room has each of it's six sides labeled.

6.2 VFog

Vfog is a very simple and fast tool for adding a ground fog to a scene. VFog isn't a true volumetric effect; it operates by compositing the fog's effect onto your rendered imagery. This method has limitations, including not showing up behind transparent objects or reflections and not appearing in tools like Enviro which do ray tracing to sample the scene's appearance.

Minimum/Maximum Amount restricts the opacity of the fog, useful when you never want extremely distant objects to be completely obscured.

Clear Camera Distance is a fog-free sphere surrounding the camera. This is useful to keep foreground objects completely unobscured.

Fog Falloff Distance sets the fog density. Denser fog has a shorter falloff distance. This is the most-tweaked control.

Fog Layer Altitude sets the Y height of the start of the fog layer, in world coordinates.

Thickness controls the height of the transition from solid ground fog to the region above which has no fog at all.

Top/Bottom Sharp are subtle controls which change the exact rate that the fog falls off vertically.

If you're compositing, you may wish VFog to **Multiply Alpha**, which will add the fog's effect into the alpha channel. If you want to make the alpha channel show *only* the fog's density, you can use **Replace Alpha**.

VFog Example Scene

VFog-Range.lws Placards at different depths show VFog's effect.

Chapter 7 Image Plugins

JKP features three image plugins, Confusion, DropShadow, and Lens. Image plugins are applied in LightWave's Effects panel.

7.1 Confusion

Depth of Field (DOF) is a common artifact of real life imagery. The Confusion plugin is an attempt to make depth of field effects using a simple algorithm based on 2D image blurring. This method is not as accurate as LightWave's, but it can produce results that are sometimes faster or more pleasing than LightWave's method. No tool like Confusion will produce fully accurate output, since the depth of field is actually *not* a blur, it only has some blur-like properties. Don't depend on any blur-based DOF tool for photoreal effects!

In photography, the "Circle of Confusion" is a measurement of the depth of field effect at any point in an image. The parts of an image which are well-focused are said to have a small, or nonexistent, circle of confusion. The parts of the image which are blurry due to depth of field effects have a large circle of confusion depending on the amount of blurring. The circle of confusion is simply the disk over which the pixels get smeared due to the depth of field effect.

This information is important when compositing. A better depth of field effect can be produced by using blurs with variable blur radii based on a circle of confusion map.

The Confusion plugin can be use in two ways. If you use an external compositing package, Confusion can output a grey-level image map depicting the radius of the circle of confusion at each pixel. This is an image which is black in the "in focus" area and white in the fuzzy areas. This map can then be used in the compositing program as an input to a blurring tool. High-end studios with good compositing tools can get good results with this technique, especially since most high end compositors give a lot of control over the final effect. You can choose to replace the RDB colors with confusion map (useful for visualizing it right in Layout) or replace the alpha channel (useful if you're trying to save the RGB image at the same time).

If you don't have an external program, confusion can apply a very simple blur to the image. This is a simple blur of each pixel based on its circle of confusion.

Three are very few controls to the Confusion tool. The first is the **Lens F-stop**. A larger F-stop will reduce the depth of field effect; all the circles of confusion will get smaller.

The **Focal Length** of the camera can be defined by using the position of an object in the scene, typically a null. This is done for extra flexibility in changing the camera focus, especially over time. You also have the option to enter the focal length numerically. When you use a numeric focal length, remember that this is the distance as measured from the camera. An object located 10 meters away from the origin may be more or less distant from the camera depending on the camera's placement.

You must choose the effect you want applied. You can either output a confusion map in the RGB or alpha channels, or you can apply a blur base on that map. If you decide to blur using the Confusion tool, you can also use the **Blur Background** button to indicate whether the background should also be blurred. since the background is usually at infinity, it tends to be completely out of focus, which may not be appropriate.

Note that there are several free and commercial plugins very much like Confusion. All of these plugins, including Confusion, are fatally flawed because of the incorrect substitution of a 2D blur for the true depth of field effect. Nonetheless, they can be useful in many cases until new algorithms add high quality true depth of field effects to renderings.

Confusion Example Scene

Confusion-Focus.lws This scene shows eight signs imprinted with their distance from the camera. The focal length of the camera is controlled by a null object which is positioned to be at the same distance as the sign labeled "4 meter."

7.2 DropShadow

The Dropshadow plugin recreates a very common effect often done by hand in a paint or compositing program. It uses the LightWave images' alpha channel as a mask to lay a blurred, darkened area onto the background. This is a quick and cheesy way to emulate a true drop shadow.

The "shadow" is just a simple 2D effect, an offset of the shape of the original object onto the background. This is often appropriate for logos and 3D text. Worley Labs often uses this plugin for making web page images, since any graphic element can easily be given a quick soft shadow.

X, Y Offset defines how much the shadow is displaced from the original object position. The distance is measured as a percentage of the image size, so the effect renders properly at all resolutions. A small value of 10% or less tends to work best.

Shadow Softness determines how much blur is applied to the shadow. Higher blur increases rendering times.

Shadow Color is the applied shadow color.

Shadow Opacity is the strength of the shadow. A small value makes the shadow lightly tint the background with the applied color. A value of 100% will completely replace the background with the shadow color.

Add to Alpha instructs Dropshadow to apply the shadow to the image's alpha channel. This is useful if you later composite the rendered image onto another one.

DropShadow Example Scene

DropShadow-Racer.lws A simple scene which renders a wheeled car and applies Dropshadow.

7.3 Lens

The Lens tool is a powerful 2D image manipulation tool that allows you to compensate for most kinds of camera lens distortions. The basic idea is that pincushioning or other real-world lens effects can be removed by a 2D distortion; the Lens plugins can stretch the image so it's "flat" again. This ability is critical for studios doing advanced composition and motion tracking work. It can also be used just for fun or special effects.

LightWave renders a mathematically perfect "pinhole camera' image. Real cameras don't. There are usually significant distortions in real filmed imagery, the most common being pincushion "bowing" effects, especially at the top and bottom of the image. This is most pronounced when wide angle and anamorphic lenses are used. Image offsets (simple shifts) are also common, mostly from the transfer from recorded film to a digital image. Digitizing from video capture hardware such as a PVR often adds a shift of a few pixels, and film recorders sometimes offset the image by as much as 50 pixels. This means the camera's vanishing point is no longer at the center of your screen, but moved slightly to one side.

If you want to composite some LightWave objects into imagery with this kinds of distortions, what can you do? A bend of 10 pixels is deadly, since the view will clearly see that the two components don't merge.

Most studios do this kind of "un-distortion" with a compositing tool. However, it's much easier to do it directly in LightWave itself.

To use Lens, just load your distorted background imagery and use the interactive display in the Lens tool to stretch and manipulate the image so it's "flat". You can then use LightWave as a tool to "un stretch" all your distorted imagery, and you're ready to composite in LightWave without worrying about fisheye or other lens effects. Lens can apply corrections for image **pincushioning, shears, keystoning, scaling, rolling** and **offsets**

Lens has been designed so that it can be used *reversibly*. What if you really need your final output imagery to *match* the distorted imagery? It would look strange to a viewer to see the camera's fisheye behavior pop in and out depending on the cut they view.

Lens has an **Inverse Effect** option which allows you to do this. If you have perfected your settings to "flatten" some background imagery, you can use the Inverse Effect button to make the Lens tool apply the deformations to a *flat* rendered image, forcing it to match the original distortion. Thus, you can use Lens to "flatten" your backgrounds to composite immediately in LightWave, or use Lens to distort your LightWave imagery to match your backgrounds.

Any optical distortion loses information, especially when parts of the image are forced off screen. For example, if a pincushion effect bows the image out on the right and left sides, when you remove the effect, there's an area on the right and left sides that wants "offscreen" pixels to fill itself. We don't have these, so it just fills them with a solid color (which you can choose).

To test the "invertibility" of the Lens tool, you can add two copies of the plugin, each with the same settings. Click the :inverse transform" on one of them, and the second one will "cancel out" the first. If you have a large distortion, this cancellation will never be perfect; you lose image detail when you really squish pixels around. You can use the **Filter** option to remove the pixel-sized jaggy artifacts. The artifacts are minor for any real lens distortion, but it's a lot of fun to use giant distortions to play with.

While Lens was really designed for composite matching, you can use it for fun too. At very high distortion levels you could make a weird space warp effects. It may be useful to have LightWave's output shrunk to 1/3 size and offset in the corner, sort of like an over-the-shoulder infographic on a TV news show.

Lens Example Scene

Lens-FishFrame.lws This scene renders a rectangular frame and applies the Lens post-process to the image. The long straight edges of the frame help show the effect of Lens when applying pincushion and other effects.

Chapter 8 Help

Things can go wrong. No, things *will* go wrong. 3D rendering is a complex process and there are a lot of opportunities for problems to appear. When you do have a problem, such as an object rendering incorrectly or a scene rendering too slowly, you have to learn how to diagnose and fix the problem, or at least find a way around it.

You can find a list of known software bugs and patches on our web site at http://www.worley.com/support.html. NewTek posts LightWave patches on its web site at http://www.newtek.com. You should periodically visit both of these sites to make sure you're running the latest versions.

No matter what the problem, the best way to solve it is to accurately diagnose the cause. This requires a good understanding of the programs you're using so you know what you *should* see. If you understand your tools, you'll know when they really aren't working properly, and usually why.

To diagnose problems, you'll have to understand not only our plugins, but LightWave as well. This manual contains everything you need to use our software, so please read it! If you don't understand what a plugin is meant to do, you can easily misinterpret its proper behavior as an error.

Understanding the software is important when using LightWave in general. It is a complex program and difficult to master. Read its documentation too.

If you don't understand your tools, you will never be able to use them to their full potential.

8.1 Common Problems

There are a lot of common problems that can occur. The list of frequent questions below may help you. Make sure to read the full documentation of the specific plugins you're using as well! Most of them have their own FAQs which address problems and questions specific to that individual tool.

Frames rendered via ScreamerNet don't show any plugin effects.

Our plugins fully work with ScreamerNet, but LightWave can be *very* picky about proper ScreamerNet installation. You must ensure that your LW.cfg file has the plugin properly installed. This often does not happen automatically when you are running over a network, since you may have installed the plugins using an *absolute* path, like C:\newtek\plugins\taft.p , which hardwires the plugin file location to be on a specific drive on the local machine. You should usually use a *network* path

When ScreamerNet hits a plugin error, it continues rendering anyway. This makes it appear as if the plugin has simply disappeared from the frames rendered by that ScreamerNet node. **ScreamerNet will not print any warning or error message if it can't find missing plugins.**

Viewing the LW.cfg file manually is often useful when troubleshooting Screamernet problems. Consult your LightWave manual (and NewTek customer support) if you are having difficulty using plugins via ScreamerNet on remote machines.

How can I automatically apply a plugin to all the surfaces in my scene, or all the objects?

LightWave doesn't automatically add plugins to surfaces, which means you must manually add the plugin to each surface you wish to have affected. It is often easiest to use the ``L'' and ``S'' buttons at the bottom of the plugin's interface to apply a starting template of values to your surface. By holding ``shift'' when you click the ``L'' and ``S'' buttons, you can cut and paste without using a disk file.

8.2 Resources

We release patches to our software when we find bugs. You should periodically check our web site at http://www.worley.com/support.html to ensure you have the latest version of the plugins.

We're always pleased to hear from you, and if you've made anything interesting with our software or have plugin ideas or suggestions, we're interested! We also want to correct any spelling or factual errors in this manual or on our web site. This is especially true for bugs in our software; we try very hard to have no bugs at all in our tools.

You can get in touch with us a number of ways.

Snail mail: Worley Laboratories, 405 El Camino Real #121, Menlo Park, CA 94025

Web site: http://www.worley.com

Email: support@worley.com