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

Taft Manual

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

Chapter 1 Introduction

The William Howard Taft Plugin Collection (Taft for short) is designed to extend the power of NewTek's LightWave 3D. Taft has been developed in close association with the largest LightWave studios over a period of two years. 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

Taft is a Layout plugin file for LightWave version 8.3 or later. All of the plugins are contained in the single **taft.p** file. The software uses about one megabyte of hard disk storage. Installation is the same for PC, Mac, and Alpha versions of LightWave.

You can always get the latest version of Taft 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 *taft.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 *taft.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 underpriced 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. Taft uses the dongle you already have for LightWave as its method of protection. Taft 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 Taft 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 Taft, the plugin can load, save and render even if it has not been not licensed yet. However, the *interface for each plugin will not be active until the plugin is licensed.* When you try to open the interface by double-clicking the plugin name, the plugin licensing panel will open. It tells you the plugin name, version, date, 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 generate the code. We also need your plugin serial number, which should 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 license file on your hard disk 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 are permitted to use the plugin on any number of ScreamerNet nodes.

Demonstration Scenes

Taft has a large collection of Download the example scenes.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 Taft 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.

Extracting the example files creates a directory named *WorleyDemos* (The *WorleyDemos* directory makes it convenient to store the example files for our different products in the same place.)), which contains a subdirectory named *Taft* . Set your LightWave content directory to the *WorleyDemos* directory you created in order to view the scenes properly. If you load a scene and LightWave complains about missing objects, you've probably mis-set your content directory.

1.2 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: ewtek\plugins aft.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. 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.

1.3 Legalities

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 of Taft as part of a rendering network via ScreamerNet. Purchase of a single copy of Taft 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!)

1.4 The William Taft Collection

I started to write the plugins in *The William Howard Taft Plugin Collection* (If you're not an American, don't feel slighted by this American-centric choice of name for the collection. You may have never heard the name "William Taft" before, but most Americans have never heard of this guy either.) in January 1998. Its long development has produced a variety of tools designed to help in a Hollywood production environment. The long testing period has polished each individual plugin, so bugs are rare.

The collection is named after William Taft because a name like *MegaPlugin SuperTools Pack #2* is stupid. Our policy for plugin collections is to use the theme of "obscure US presidents" for the titles of packages. We named our first collection after President James K. Polk. It's been wildly successful, and we are now proud to follow it with the Taft Collection.

1.5 William H. Taft

William Howard Taft was the 27th President of the United States, serving from 1909 to 1913. He spent four uncomfortable years((President Taft was a huge man, weighing more than 300 pounds. A special bathtub was installed for him in the White House, big enough to hold four men.)in the White House. Large, jovial, conscientious, he was caught in the intense battles between Progressives and conservatives, and got scant credit for his achievements.(This entire biography is stolen from the official White House biography of Taft. I think they really enjoy using adjectives when describing presidents.)

Born in 1857, the son of a distinguished judge, he was graduated from Yale, and returned to Cincinnati to study and practice law. He rose in politics through Republican judiciary appointments, through his own competence and availability, and because, as he once wrote facetiously, he always had his "plate the right side up when offices were falling."(Elected because of competence and luck, he claims. Today, presidents are elected by money and marketing. Things have changed.)

But Taft much preferred law to politics. He was appointed a Federal circuit judge at 34. He aspired to be a member of the Supreme Court, but his wife, Helen Taft, held other ambitions for him.(Ah, his wife ran the family.)

President Roosevelt made him Secretary of War,(The United States renamed the War Department to the Department of Defense in 1947 to be more politically correct.) and by 1907 had decided that Taft should be his successor. The Republican Convention nominated him the next year.(Shouldn't the American people have decided, not President Roosevelt?)

Taft disliked the campaign--"one of the most uncomfortable four months of my life."(He'd hate campaigning in 2000. It lasts 14 months, not 4!) But he pledged his loyalty to the Roosevelt program, popular in the West, while his brother Charles reassured eastern Republicans. William Jennings Bryan, running on the Democratic ticket for a third time, complained that he was having to oppose two candidates, a western progressive Taft and an eastern conservative Taft.(Too bad Bryan didn't have a brother.)

Progressives were pleased with Taft's election. "Roosevelt has cut enough hay," they said; "Taft is the man to put it into the barn." (This slogan would not work well today.)

Taft recognized that his techniques would differ from those of his predecessor. Unlike Roosevelt, Taft did not believe in the stretching of Presidential powers. He once commented that Roosevelt "ought more often to have admitted the legal way of reaching the same ends."(I guess this means Roosevelt didn't worry too much about obeying the law.)

Taft alienated many liberal Republicans who later formed the Progressive Party, by defending the Payne-Aldrich Act which unexpectedly continued high tariff rates.(Hey, this kind of specific detail is irrelevant. It's not like we *really* care about his history.) A trade agreement with Canada, which Taft pushed through Congress, would have pleased eastern advocates of a low tariff, but the Canadians rejected it. He further antagonized Progressives by upholding his Secretary of the Interior,(I hope he wasn't too heavy.)

accused of failing to carry out Roosevelt's policies.

In the angry onslaught against him, little attention was paid to the fact that his administration initiated 80 antitrust suits and that Congress passed amendments for a Federal income tax and the direct election of Senators.(Yes, we have Taft to blame for income tax.)

Taft, after the Presidency, served as Professor of Law at Yale until President Harding made him Chief Justice of the United States, a position he held until just before his death in 1930. To Taft, the appointment was his greatest honor; he wrote: "I don't remember that I ever was President."(You know, if I had been President, I don't think it would slip my mind so easily.)

This Manual

This manual is the reference guide for each of the seven Taft plugins. Each character is hand-printed by cloistered Tibetan monks,(Errors are punished by head-shaving to keep the writing so perfect it appears machine-printed.) using a snowy goose feather quill and black, llama-oil based ink. The paper is carefully made from hand-sorted Vermont poplar woodpulp, aged for exactly 1107 days. Each sheet is examined for imperfections by x-ray and microscopic inspection.

Chapter 2 LightWave and Plugins

Taft'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 Taft's tools, they are important to understand.

Each plugin has an "About" screen accessible by clicking its titlebar. 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 taft.p is sometimes referred to as a "plugin file," but really it's just a container holding the seven Taft plugins. Try not to get confused when even our own documentation calls Taft 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 website. Check NewTek's website 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 onlne.

2.3 Item Picker

Many of the plugins have controls which refer to items in the scene. For example, the plugin *Hoser* asks you to pick an object to attach a rope or hose to. Taft 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 Numberic 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 keyframed 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

Taft 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 Camera Match

Combining LightWave renderings with real world images can produce extremely realistic effects. It's fun to use LightWave to add giant rocket engines to your car in a photograph! This kind of photoreal 3D compositing is probably the hardest task in LightWave, but it's also the most *common* use of 3D on TV and especially movies.

It's very difficult to seamlessly incorporate a rendering into a real photographed image. You have to worry about matching lighting, shadowing,(Our *G2* software solves the problem of matching real world lighting and shadowing.) and position. Sometimes matching the position is easy; it's simple to place a LightWave airplane into an image of a cloudy sky, since nearly any position and angle will look good. It's much harder to take a photograph of your one-story house and use LightWave to add a fake second story with giant gun turrets. (It may be easier simply to rebuild your house using real gun turrets, photograph it, and ignore the LightWave step altogether.)

Matching becomes difficult in situations when any small error in position or angle is easy to notice. If you move the composited airplane five pixels, it still looks like an airplane in clouds. If you move your house's new top story five pixels, you see only a badly manipulated photograph with an obvious problem. Often a position error of only a single pixel will ruin the effect.

When you try to match the position of your LightWave objects by hand, you have to use good judgment and frequent rendering tests to ensure your placement is correct. Most image alignment setups are deceptively difficult. It's easy to align one part of your object with your background image, but it's harder to align two points simultaneously. With something like your house, you have to align the whole building at once!

3.1 Matching Feature Points

The idea of matching points is the key to understanding how to properly position your object and camera for compositing. Compositing an airplane into a cloudy sky is easy, since you don't have to align the airplane with anything. Compositing an airplane resting on an empty runway is only a little harder. When it's on the ground, you only have to match one constraint: that it's not floating above the ground or penetrating below it.

It's significantly more difficult to match more than one point simultaneously. Using the airplane example again, imagine that you have a photograph of a real airplane on the ground and you want to *replace* the real airplane in the image with your LightWave airplane. This will allow you to do fun things like change the airplane's paint job, add exotic weapons, or crossfade the image into a wireframe blueprint.

Matching your LightWave airplane to the real airplane is *very difficult.* Even if your model is perfect, it's frustrating to find the exact position, size, and scale of the object to make it match your image. A small shift to align the landing gear may move the wing too, causing it to be misaligned. Rotate the model a few degrees to re-align the wing, and the tail will slide off center. Fix the tail, and your landing gear will be broken again.

3.2 Matching Strategies and Limitations

There are two methods to find the best alignment. The first is a combination of skill, luck, and patience, simply manually adjusting the object carefully until it matches your image. It is not an exaggeration to say that a difficult match of a single image, even without animation, might take a full day.

The second method for perfect matching is to change strategies. Instead of using your eyes and skill, you use measurement and a tool: Taft's Camera Match plugin.

Camera Match uses *mathematics* to automatically determine what camera settings (position, rotation, zoom, and pixel aspect ratio) will match your image best. The disadvantage of this technique is that you have to tell the plugin about what you want to match (which points on your object) and where they should match to (which pixels in your image.) The quality of the match is only as good as the quality of the data you give it. Inaccurate or insufficient data will make the match quality suffer.

Taft's Camera Match plugin is still limited in the sense that it is not designed for camera *tracking* , which would analyze 2D image sequences, automatically follow the feature points as they move, and produce an *animated* camera path. This kind of camera tracking tool is more useful than simple camera matching. But it's

also a *lot* more effort to write and use properly; the commercial packages which do it well cost thousands of dollars.

Even if your data is perfect, matching some kinds of photographs is still impossible. Specifically, LightWave uses a simple camera lens design which can't match all real camera lenses, no matter what settings you select. (The *Lens* plugin in our James K. Polk plugin collection was written to solve this problem.) Luckily, most cameras use lenses which LightWave can easily approximate.

3.3 Using Camera Match

Camera Match uses the idea of *feature points:* easy to measure representative locations on your object. These feature points need to be easy to identify so you can see where they appear in your image. Camera Match determines the settings that best move these representative points into place, which should properly position the rest of your model.

For example, if you have a photograph and a LightWave model of a car, you might pick feature points like the center of each headlight, the corners of each window, the outer point of the side view mirror, the front corner of the hood, and the center of the door handle.

You can choose up to 20 feature points. Camera Match will always do a better job when you use more points, so use as many as you can! You must use at least 4.

The feature points should be easy to identify in your image so you can determine their position accurately. The center of a featureless door is a bad choice because there's no detail in your image to show exactly where that point is. Corners, edges, and joints are usually the best choices because it's easy to locate their exact pixel location in your image.

You also need to identify the same locations, in 3D, in your scene. Very often you'll need to take careful measurements of your object in order to locate the 3D points accurately. Sometimes this measurement step is easy, especially if you have blueprints or your object is something like a brick tiled walkway which are easy to count and measure.

It's more difficult to measure a complex or organic object, so you have to be more careful and use more points to compensate. A tape measure and a notepad are invaluable for measuring. (We like using a *metric* tape measure since it's much easier to add centimeters than feet, inches, and fractions.) If you have a good 3D LightWave model of your photographed object already, measurement isn't as important since the model itself already shows the proper locations.

Next, find the exact pixel location of each of the feature points in the image. You can do this with a tool like Photoshop by zooming into the image and using the cursor position readout. You can also do the same thing using LightWave by using your photograph as a background image, and using QV4's zoom and cursor location display. In the end, you'll end up with a list of X,Y pixel locations of each of your feature points. (If you're not using *QV* to display LightWave's renders, you should. You can get this plugin from Ernie Wright's webpage. It's vastly superior to LightWave's image display.)

Now set up your LightWave scene. Each feature point needs to be marked for the plugin to find it. The easiest way to do this is with null objects. Add one for each of your feature points. It's a good idea to name them descriptively, like "Top Window Corner" or "Doorknob" since it's confusing to have dozens of points named similarly to "Null (17)."

If your object is movable, like a car, you should parent the nulls to the object and move them into their proper 3D positions. If you're fitting something that will never move, (like the ground or a building) you don't have to parent the nulls since you'll never move them as a group.

If you've measured your object, then it's easy to use LightWave's numeric position control to specify the exact location of each null. If you don't have measurements, you'll have to use your best judgment in placing them, moving each null to the proper location on your model. (This is much easier and accurate with a good model.) Without measurements, you'll need more feature points to get a good fit.

It's not necessary to actually build a model of your object. You only need to be able to place the null objects in the appropriate locations in Layout.

3.4 Entering Your Data

Camera Match is a motion plugin which is applied to the camera in the Camera Motion Options.

Most of Camera Match's interface is a long list labelled **Feature Points.** This is where you enter the data you've measured. For each feature point, use the item picker on the left to select the null object defining that point.

Next, enter the pixel location of the feature point to the right, in the columns labelled **At X Pixel** and **At Y Pixel.** It's OK to use fractional pixels like 135.4 to get extra accuracy if you can guess the position of each point that accurately.

Finally, enter the size of your image in the area labelled **Original Reference Image Size** if necessary. The default values match your current LightWave image size, so these are usually already correct.

After you've entered the feature points and pixel positions, Camera Match displays a lot of interesting information. The box on the bottom left of the panel, labelled **Compute Best Values For**, lists the camera settings that produce the best match. This includes the camera position, rotation, and zoom, as well as the image pixel aspect ratio.

Camera Match also gives you two error numbers labelled **Original Error** and **Final Error.** A perfect fit (0.0 pixel error) means that every feature point is placed exactly at the proper image point when rendered. Worse fits have larger errors. The Original Error shows the original camera setting quality before fitting. (You can try your manual camera matching skill by matching an object then using Camera Match to tell you how large your error was.) A good fit will have an error of less than two pixels.

The Final Error shows the quality of Camera Match's computation. If the error is high (over five pixels or so), your input data probably has errors. These may be from bad feature point measurement, an inaccurately placed null object, or a poor 2D pixel position.

The right hand column labelled **XY Match Error** shows the *individual* X and Y pixel error for each feature point. This shows how far away each feature point is from its desired position in the reference image, which may give you a hint about why the point doesn't match well. The points which have the highest error are flagged with a red color to make them easy to spot.

You have control over which camera settings are fitted. For example, if you want to force Camera Match to use your keyframed camera zoom during the fit, click off the **Zoom** button and Camera Match won't fit that value.

This ability to restrict what channels are fit is sometimes useful, but it will never improve the quality of your fit. Any restriction gives Camera Match less control, so it will have a harder time fitting your data.

The most common restriction is **Aspect**, LightWave's pixel aspect ratio. Camera Match will tell you the best aspect to use, but you may want to override it since the aspect is not often changed in LightWave and is never animated.

Restrictions on position **XYZ** and rotation **HPB** aren't recommended, but are available if you need them.

Restricted parameters display the keyframed value in grey text. Values computed by Camera Match are displayed in bright yellow.

You can view the fitted camera parameters for any frame of an animation by changing the **View Frame** control. This tells you the camera parameters that best match your feature point positions for that frame. (This feature doesn't help for animated camera tracking.) You'll nearly always leave this set at the default, frame 1.

3.5 Using the Matched Camera Parameters

After Camera Match has finished its fit, you're probably eager to see how it looks! The plugin can move and rotate the camera when you activate the **Move Camera** button on the bottom right. Unfortunately, LightWave doesn't allow plugins to set all of the camera settings automatically. To set the zoom and pixel aspect ratio, use a piece of paper to write down the Zoom and Aspect values displayed in the bottom left box of the interface.

Visit the camera properties panel. Enter the values for the Camera Zoom and Pixel Aspect Ratio which you wrote down.

Now you may want to make a test rendering to make sure you're happy with the fit. When you're satisfied, you can make your final renderings. However, the Camera Match plugin is usually used *temporarily* to determine your needed camera settings, then removed. After you've computed your camera parameters, you won't need to keep Camera Match in your scene any longer.

Camera Match takes about one second of computation (This was written in January 2000 using my cheesy 450 MHz x86 PC. If you're reading this note in 2009 and you're laughing at my long "one second" time estimate because of your 4 GHz quad Athlon PC, I hate you.) to compute a fit. If you leave the Camera Match plugin active, it will slow LightWave down every time it redraws the screen. To avoid this annoying slowdown, you could write down the camera position and rotation values and keyframe them manually. But it's easier to export them into a LightWave motion envelope which you can simply load into the camera motion. In the Camera Match interface, make sure Move Camera is selected, then click the **Export Motion** button. This will save a motion envelope containing your camera position and rotation. You can then remove Camera Match completely, and use LightWave's Load Motion button to apply the settings to the camera.

You might still want to keep Camera Match applied, but disabled. This preserves your fit setup if you want to modify it later. To keep Camera Match disabled, simply make sure that the Move Camera button is off.

3.6 Camera Match FAQ

Question: One of my feature points is flagged with red, showing a high error. But I've checked, double checked, and triple checked the point and it's very accurate. Why does Camera Match say it's wrong?

Answer: A point with high error (shown in red text) means that that point's information is *inconsistent* with all the other points. If you know the point is accurate, then *all of the other points* must be wrong. *This is quite possible.* It's easy to make a measurement error which offsets a whole group of points by accident. This happens because it's often convenient to measure the position of one feature point relative to another feature point. This means that any error in one point is shared with the points that you measured relative to it.

For example, if you define point A to be at 0, and measure B to be 10 cm to the right of A, and C 10 cm to the right of B, then obviously C is 20 cm from A. But this means that C depends on *two* measurements, so it's twice as likely to be wrong. If you make an error with measuring B's position (perhaps measuring it as 9 cm, not 10,) then that error will be shared with C. You'll think that C is 10+9=19 centimeters away from A, even though your C measurement of 10 cm is correct. In this case, B and C are consistent with each other, but A *looks* like it's wrong, since it doesn't match either B or C.

Moral: Double check *all* of your measurements when you have a poor fit.

Question: LightWave becomes sluggish when I use the Move Camera option.

Answer: This option isn't meant to be left on. It's just a quick way to view the applied camera position and rotation to see its effects before you export them with the Export Motion button.

The slow speed can be a major problem if LightWave's "Show Motion Paths" option is on, since LightWave will call Camera Match literally hundreds of times for each screen redraw. This can slow LightWave down enormously, in some cases taking 20 seconds or more to redraw. Don't use LightWave's "Show Motion Paths" if you're using Camera Match's "Move Camera" option.

Question: How do I match a scene when I didn't measure anything, or the points aren't very distinct so I can't identify them easily?

Answer: It may not be possible. If there are no reference points, there's no way for the plugin to know what to do. You may be forced to use your best guesses for missing data, which will probably give a poor fit. In that case, you may want to use that fit as a starting point for a manual matching; without information from you, the plugin can't help much.

Question: I have a low fit error, but it produced crazy settings like Zoom=100. Why?

Answer: When you have a very flat object, you get a nearly identical image if you move the camera backwards and zoom in. With no depth, the flat object doesn't give any clues about what kind of zoom your

real camera used. You can force the zoom to be any value you like using the zoom restriction button. With flat objects, this often doesn't hurt the fit quality much.

Question: I cropped a photo, and Camera Match can't fit the points as accurately as it did when I used the original full image. Why?

Answer A cropped photograph can't be accurately rendered with LightWave's rendering model. A real photograph's vanishing point is in the center of the image. If you crop the image, that vanishing point will no longer be centered. Neither real cameras or LightWave make photographs with offset vanishing points. Camera Match will still *try* to fit the image, but the fit will probably be degraded.

Question: My fit is obviously wrong. Why?

Answer: There can be many reasons. You may not have enough data, so you should use more feature points. Your data quality may be poor; check your measurements. Your data may be badly chosen; make sure that you choose points as widely spaced as possible, covering the entire image if you can.

Camera Match's fit should be optimal; *no* other camera settings will give you lower mathematical error. When there's a problem, it's usually with the quality and quantity of the input data.

Question: If I use any restrictions on the camera settings, the match quality sucks.

Answer: Restrictions are usually not a good idea unless you're very sure you need them. For example, you might *think* that your camera has no bank rotation, so you want to restrict the camera Bank to 0~degrees. But even a 1~degree bank error can prevent a camera match from being successful.

Question: What is the difference between camera *matching* and camera {\em tracking?)

Answer: Camera matching finds the camera settings required to recreate a view of an object in a photograph.

Camera tracking finds *animated* camera settings for an entire animation. Additionally, a tracker will usually automate the process of identifying the motion of the 2D pixel locations of moving feature points.

Taft's Camera Match plugin is designed only for *matching* . It can be used for tracking with significantly more effort, but it's not much fun.

Question: So how do I do camera tracking for my animation anyway?

Answer: This is a lot harder than matching a single frame! Camera Match can help you, but it's not what the software is designed for. But if you do have to track something, you can still use Camera Match to make it easier.

To track an animation, start by using Camera Match normally on the first frame to get good initial settings. Then, go to a later frame (perhaps 30 frames later) and fit *that* frame. This second fit will be easier, since you only need to update the 2D image pixel locations, not the 3D null object locations. Repeat this for the length of your animation, matching perhaps every 30th frame.

Now watch your animation in LightWave. Since you didn't match every frame, it's likely that the match will be poor during some of the interpolated frames, especially if your object motion is fast or wild. Find the frame with the worst error, use Camera Match to find the best settings for that frame, and use those settings to make the frame into a keyframe. Keep repeating this process of finding and matching the worst matched frames, and eventually the whole animation should be tracked with low error.

This process is tedious, and you have to do a lot of cut and pasting of camera locations. But it does work, and it's still easier than manual matching. If you do this kind of tracking a lot, you may want to spend the money to invest in a camera tracking program.{Digital Domain is one studio that wrote their own in-house tools for tracking. A couple other studios have their own private tools, but I don't know a commercial LightWave tracking program.)

3.7 History

I visited Amblin Imaging{Now defunct.) in 1994 when it was the largest LightWave studio. They pioneered the use of LightWave in a TV series with the show *SeaQuest.* They were doing extremely well and bidding on larger projects like movies.

They were excited about the idea of seamlessly compositing LightWave characters into real, filmed sequences. In the days of 25 MHz Amigas, this was an ambitious goal! The hardest part of compositing wasn't making photoreal characters, but making their motion match the filmed background.

They were bidding on a movie called *Small Soldiers* , which featured toy dolls running across the floor, climbing on furniture, and interacting with the real world. Integrating these characters into filmed sequences was very difficult, especially finding the proper LightWave camera motion to keep the characters from "sliding" across the floor. The artists had no tools to match the real-world motion except their eyes and experience, which simply weren't enough.

When I visited the studio, they were very eager to find any alternative method for camera tracking. I wrote a primitive, GUI-less tool which could do most of what the Camera Match plugin does now. It was cumbersome, but it did find the proper camera settings accurately.

They wrote some inhouse tools (in Visual BASIC if I remember correctly) to help make my program easier to use. The program I wrote read and wrote raw text files, and these had to be manually moved to and from LightWave.

Amblin didn't get the *Small Soldiers* job, but they did use my primitive program for *SeaQuest.*

My experimental camera match program sat unused for a couple of years until a small studio heard about it and begged for a copy. They used it successfully but it was still awkward to use.

In 1999 we were testing Taft's Sticky~FP plugin at different studios. Several artists realized StickyFP would be much more useful if you could accurately match LightWave's scene to the camera used to take the sticky image map.

I dusted off my old camera matching software and made a LightWave GUI for it. I become unsatisfied with the quality of the matching, so I kept the new GUI but re-wrote the matching algorithm completely. The version of Camera Match in Taft is faster, more robust, and more accurate than my five-year-old test matching tool.

There's a lot more that could be added to Camera Match. It would be nice to make a GUI for selecting image points directly instead of having you type them in. It would be especially great to extend Camera Match into a camera tracking tool, though that's an extremely difficult job to do properly, mostly because of the extensive GUI which is needed to do it right.{The math is also difficult, but I *like* that part!)

Chapter 4 Hoser

The world is filled with tubes, ropes, tails, cords, tentacles, antennae,{The English language is stupid. Radios have *antennas* but insects have *antennae.* } strings, ducts, pipes, vines, cables, and hoses. There are countless long, flexible cylinders in the world.

It's annoying to make tubes which deform smoothly in LightWave. The most common method uses LightWave's bones. To bend a tube, you make a long chain of bones going down the tube's center. You control the bone chain using LightWave's inverse kinematics (IK) and the tube bends reasonably smoothly as long as you use enough bones. This technique works in LightWave without any plugins needed. But it's a pain to set up! And LightWave bones and IK are not fast, so it makes Layout sluggish.

The Hoser plugin is designed to replace the tedious bone-and-IK method with a quick alternative which just "does the right thing."{Sometimes I feel that plugins that just "do the right thing" are the best kind. Of course usually "the right thing" really means "whatever will make my client happy" or "whatever will let me finish this shot so I can go to sleep soon."} Hoser takes only a few seconds to set up, and it provides faster and smoother results than bones.

4.1 Setup

Hoser is a displacement plugin you apply to an object (usually a tube) in Layout's Object panel (LW 5.x) or Object Properties panel (LW 6.x). The plugin has a simple interface, so you can set it up extremely quickly. Hoser's main ability allows you to move and rotate *either end* of a tube and have the entire tube bend and stretch realistically.

To use Hoser, first model your LightWave object with the longest direction along the X~axis. Hoser will bend and move the part of the object to the right of the X=0 centerline. Most objects are just hoses, so you'd make their left end at X=0, with the tube laid out to the right. You should model the hose in its straightest orientation, centered on the axis lines. For a tube connected to a solid object, like a cat's tail, leave the cat's body to the left of X=0 and the tail stretching to the right.

Your hose should be made of triangles, not quads. This is because any deformation of an object in Layout (by plugins, bones, or morphing) often turns quads into non-planar polygons, which LightWave doesn't render well. Triangles don't suffer from this LightWave problem. Using more triangles will make the bending smoother, since the smaller polygons are less visible.

The tube can be textured, and even have polygon details like an accordioned zig-zag, or a helical spring shape. Most of the time you'll just extrude a simple tube, but you're not limited to simple designs. You can model an entire arm, a treebranch, a spring, a tentacle, or a chain.

4.2 Hosing

In Layout, load your tube object and apply Hoser. The base of the tube is controlled by the tube object itself, but you'll need a second, "anchor," item (usually a null object) to control the end of the tube. Add that item to your scene if necessary.

Hoser's interface is quite simple and setup only takes a few seconds. Use **Anchor Item** to pick the item which will control the end of the tube. Your tube will now bend and react to the motion and rotation of both the tube object and the anchor item you defined.

The behavior of the tube can be further modified with Hoser's **Base Stiffness** and **Anchor Stiffness** controls. A higher Stiffness value makes that end of the tube less flexible. A stiff tube makes stiffer, larger, arcs in the tube when it's bent. Low stiffness is like a rubber band which simply stretches and contracts instead of arcing.

A setting of 0% for both Base and Anchor Stiffness is especially useful. This makes the tube behave like a taut rubber band, always stretching in a straight line between the base and anchor points.

It's not obvious that stiffness is correct

This rubber-band mode is particularly useful for connecting seperate items with ropes and springs. No matter how you move the objects, the tube will simply grow or shrink to fit. This has many applications! The springs in a car's suspension will automatically stretch-to-fit. A sailboat's rigging is easy to set up and animate because you don't need to keyframe direction and scaling of a tube; just move the anchor and base.

Hoser's final option, **Tip Twist**, allows you to twist the ends of the tube. You can't twist the tube more than 180 degrees in either direction, since your control items don't show the difference between (for example) a full 360 degree twist and no twist at all.

Hoser does *not* work when other displacement, morphs, or bones are applied to the hose. Hoser needs to know the shape of the hose in order to properly position it, and other displacements interfere with this computation.

Hoser will show its effects interactively in Layout if you leave the **Interactive in Layout** button selected.

A bug in LightWave model import/export can occur when you modify an object in Modeler and use the "Put" command in LightWave~5, or Synchronize Layout in LightWave~6. Hoser won't update its hose properly because it thinks the hose is still the old shape. A workaround for this problem is simply to use LightWave's "Save Object" button after you export from Modeler.

4.3 Hoser Application Ideas

- Tornados! Apply hoser to the funnel cloud and use a null for an anchor to move the tip of it in an erratic path of destruction. By using a null to control the anchor/base settings you can easily have the funnel cloud bend and twist. Some fractal noise tied to a spinning reference null should handle the texture for the fast moving clouds.
- Scuba diver air hoses.
- Springs on the suspension of a car.
- An elephant's trunk, octopus tentacle, or snake.
- A mouse cord, balloon string, or telephone handset cord.
- Hydraulic hoses on a robot or heavy machinery.
- Church bell ropes, a bungee jumper, or parachute cords.
- Flexible hoses connecting a truck with its trailer.
- Prehensile tails.
- Spaceship fueling lines.
- Control wires joining mysterious electrical boxes.
- A diagram's caption arrows which re-adjust when you move the caption or object. (Use the rubber-band settings of 0% Base and Anchor Stiffness.)
- The cable of a winch, crane, or elevator. Use the rubber-band settings of 0% again to make the cable always stay taut as you raise or lower the load.
- A spider weaving a web. Use Hoser's rubber-band settings to have the strand stretch out from the rear of the spider as she's spinning.
- Make a "tube" which is simply flat square. Attach one end to a moving object, and it will be stretched out behind you. Mowing the lawn.. A kind a geometric, cheesy, Acid, with no curves

4.4 Example Hoser Scenes

Hoser_BaseStiffness.lws

The Stiffness value controls the shape of the bent tube. This demo shows the effect of Base Stiffness on four tentacles. The leftmost tentacle has Base stiffness of 0%, and by scrubbing through the animation frames, you can see how the base of the object bends easily. The next three tentacles have progressively increasing

Stiffnesses of 33%, 66%, and 100%. You can see how the base becomes more and more resistant to bending as the Stiffness increases. You can control the Anchor Stiffness similarly.

Hoser_Crane.lws

Hoser can be useful to make ropes or cables that are being "wound up." This scene shows a crane that's lifting a box. The cable is really just a thin stretching cylinder. The "rubber band" option of Hoser keeps the box attached to the crane no matter how the box or crane is animated.

Hoser_HoseBar.lws

Hoser works well with parented objects. This demo shows four tubes with both ends parented to different objects. You can grab and move the parents and Hoser will still update interactively.

Hoser_Multihose.lws

You can chain two hose objects together by making the second hose object the anchor of the first. This gives you a hose which has *three* control points for even more precise manipulation. This can be extended to make hoses or ropes of any complexity.

This scene shows a tightrope weighted down by a heavy rolling wheel. The rope is made in two parts, joined at the dip. This join is animated to follow the moving wheel, keeping the dip positioned directly under it.

Hoser_Pulsing.lws

Hose stiffness can be animated, producing interesting results. In this demo, four hoses show the effect of animating their Base and Anchor Stiffness over time, causing the hoses to bulge out in an arc.{This may not be terribly useful, but it's fun to watch.}

Hoser_Springs.lws

The rubber-band technique works well with springs. This demo shows how spings will move, pivot, and stretch to keep two objects connected when using the "rubber band" setting of 0% stiffness.

Hoser_Suspension.lws

Similar to the Hoser_Springs.lws demo, but with a car's suspension.{OK, it's really a big square, not a car, but this is just a demo, not a photoreal movie rendering.}

Hoser_Tail.lws

Only the part of the object that extends in the +X direction will be affected by Hoser. This is convienient when you have a tail or arm that you want to manipulate, but you don't want the main body to move.

Hoser_Tentacle.lws

Hoser is great at controlling tentacles or tails since you can move the tip around without affecting the base. This demo shows how an alien octopus tentacle easily moves just by animating the anchor. Grab the Anchor null or Tentacle object and move it around to see Hoser's interactivity.

Hoser_Twist.lws

Hoses usually don't twist, but a tentacle might. Hoser's Tip Twist option allows you to twist the hose in addition to moving and rotating the ends.

4.5 History

One of the last plugins I wrote for The James K. Polk Collection was "Dangle," a tool which made drooping ropes. I don't remember what studio needed this tool, but its interactive speed made it fun to play with. I was demonstrating it to Stuart Ferguson, the author of LightWave Modeler, and we started talking about possible improvements to it. The most obvious extension would allow complete control over the tube, as if it were stiff. Stuart then exclaimed "And you could call it *Hoser* !" which was extremely funny.

As a test, I enhanced Dangle with extra control, but that version was slow and made Layout's update sluggish, and I knew I had to redesign it. A few months later (after stewing on the mathematics during idle moments), I wrote the first version of Hoser. It was also slow, so I spent a weekend improving its speed until it was fully interactive.

Unlike most of my tools, I didn't test Hoser at any studios since I still felt it was an experiment. In April 1999, I spent a week working at Foundation Imaging, and Dave,{Argh, Dave, I don't remember your last name, sorry!} one of the animators, asked about a tool to control hoses on a robot he was animating. I grinned because I had Hoser ready and waiting. I spent an hour making sure it worked right and gave it to him.

In November 1999 I enhanced Hoser with the tip twisting ability and some final interactivity improvements. The tip control was needed to allow one studio to animate a tree branch that "came alive" and was grabbing people.

Chapter 5 WhirleyPoints

Many objects, like a car, are very rigid and don't change shape easily. But a lot of real objects are somewhat soft, and they deform when you move them quickly. If I grab a flower and wave it around, it's going to bend and wobble. A pencil is rigid, but a thin ruler will bend slightly.

Graphics tools which simulate this kind of soft shape distortion are called "soft-body dynamics" tools. WhirleyPoints is a displacement plugin which adds effective, fast soft-body dynamics to LightWave.

5.1 Take it for a Whirl

WhirleyPoints is a displacement plugin which can be applied to any object. It loads the object geometry from disk and analyzes its structure. It uses this precomputation to quickly simulate the surface dynamics when the object moves.

To use WhirleyPoints, load an object and keyframe its motion. It's OK to parent your object; it's common to parent hair onto a head, a tail onto a lizard, or a scarf onto a neck.

Add WhirleyPoints to your object in the Object panel (LW 5.x) or Object Properties panel (LW 6.x). The default WhirleyPoints settings make your entire object "springy" and you may enjoy watching the default motion in Layout's window before you start customizing WhirleyPoints's behavior.

Activating WhirleyPoints's interface will open two windows. One is a normal plugin panel with the WhirleyPoints controls. The other is an animated OpenGL wireframe preview of your object's motion.

The preview window shows the WhirleyPoints effects in *realtime*, unlike LightWave's Layout. The preview will drop frames if necessary to play at realtime speed (which is 30 FPS by default in LightWave), so don't be surprised when its animation is faster than Layout's main window. On the PC, OpenGL is shown in a seperate movable and resizable window. On the Mac, the OpenGL preview is integrated into the WhirleyPoints panel itself.

5.2 Dynamics Controls

The most important controls are labelled **Acceleration Sensitivity** and **Centrifugal Sensitivity.** A surface with no sensitivity will not react to acceleration; it's not springy or soft, so it moves rigidly with the object. Higher sensitivity will make the surface "springier," more able to wobble and deform. Very high Sensitivity values will usually make your surface wildly distort, so mid-level settings of 20% to 60% are most useful.

Acceleration and Centrifugal Sensitivity differ by the *kind* of motion they react to. "Centrifugal" is just a fancy word used for spinning, twirling and rotating forces, caused by object *rotation.* Acceleration describes object *motion.*

When I catch a water balloon,{And if it doesn't break...} Acceleration Sensitivity makes it quiver when it stops in my hand. When I twist my neck, Centrifugal Sensitivity makes my hair fly outwards away from my head.

You can control Acceleration and Centrifugal settings independently. This lets you choose the best blended motion behavior. The greater the Sensitivity, the greater the displacement and distortion which occurs.

Restoration Force controls the *speed* of the springy dynamics restoration. High values make the surface quiver, quickly vibrating back and forth. Lower values will make slow lazy wobbles.

Damping controls how long the vibrations continue. Large values make any disturbances quickly die out, low values make your surface continue moving for many seconds after it has been disturbed.

Together, Restoration Force and Damping allow you to make a variety of effects from the fast twanging of a ruler to the slow glue-like motion of honey.

5.3 Dripping, Drooling, and Stretching

Motion Lag is an interesting alternative to the dynamics engine. This adds a kind of "delayed reaction" motion to the surface.{This is similar to LightWave's Lazy Point plugin, but with a lot more control.}

Motion Lag can be used by itself or in combination with dynamics. It has the effect of delaying the motion of parts of your object, leaving them in place as the rest of the object moves. This effect isn't used often, but it does give objects an interesting stretching appearance.

5.4 What Gets Wobbled?

By default, your entire object's surface deforms. You can restrict which parts of your object respond in several ways. The default behavior of **Apply to Entire Object** moves all of your object's points. Distant outlying points move more than central points.

The alternative option, **Apply to Named Surfaces**, allows you to pick the individual surface(s) you want to restrict the effect to. Type in the name of the surface you want to restrict the motion to in the **Surface Name(s)** area. You can enter multiple names, separated by commas. Capitalization does matter. Don't use spaces unless they're part of the surface name. You can also use the '*' wildcard, so "S*,Dup*" will match surfaces named *Super*, *Sugar*, *Sam*, *Duper*, *Duplicate*, and *DupDup* , but not *Aardvark*, *zebra*, *sam*, *Dummy* or *Dusty*. The number of selected surfaces is shown to the right, so it will report "2/5" if two surfaces are selected from an object with five surfaces.

The OpenGL preview shows what parts of your object are affected by coloring them yellow. Unaffected areas are blue. You can shift the effect towards the outlying edges of your surface by using the **Effect Bias** control. The **Sharpness** control will sharpen or soften the blend between the affected and unaffected parts of your object.

Surface restriction, Bias, and Sharpness work with both WhirleyPoints dynamics (controlled by Sensitivity) and Motion Lag.

5.5 Stretching Can Suck

Sometimes your object's distortions will be too large. You can reduce the Sensitivity controls to lower the distortion magnitude, but sometimes this hides small, pleasant, subtle motions. In this situation, you may want to adjust the **Maximum Motion** control, which prevents points from being distorted too far from their original location. Reducing this value will eliminate the wildest stretching but leave small motions unaffected.

Occasionally, there are situations when you want to limit distortions along just one direction of your object. A common example is with cloth or hair which you want to sway from side to side, but not stretch up and down like rubber.

Local Coordinate Application Strength helps in these situations. By reducing the strength in one (or two) directions, the surface will be inhibited from moving in that direction. This restriction is done in *the object's* coordinates, known as "local coordinates." This means that the restriction direction won't depend on the orientation of your object. Hair that's restricted in the Y direction to prevent it from stretching longer will still be restricted properly even when the head looks downwards or turns around.

5.6 Limitations

WhirleyPoints has many limitations. It does *not* do any collision detection, so your deformed surfaces may penetrate other objects.

You should build your object out of triangles, since deformations can make other polygons non-planar. A LightWave bug often makes non-planar polygons render incorrectly. Triangles don't have this problem.{This advice is useful in general for all displacement plugins, bones, and morphs.}

WhirleyPoints does not react to displacements, only object and bone motion. Any displacements (or morph) effects will still be shown, but their effects won't be "Whirley-ized". Bone motion *is* Whirley-ized. The WhirleyPoints panel OpenGL preview won't show any displacements (just object and bone motion) but they

will be in your final render.

WhirleyPoints will show its effects interactively in Layout if you leave the **Interactive in Layout** button selected. You can disable WhirleyPoints temporarily with the **Disable All Effects** button, which is especially useful in LightWave 5.6 which has no method for disabling plugins without removing them.

5.7 Possible WhirleyPoints Applications

- Loose, wobbling fat! Think of double chins, Sumo wrestler stomachs, bulldog jowls, pot bellies, and thunder thighs.
- Clothes. Motion Designer is the best solution for most clothes, but WhirleyPoints is faster and more interactive for many kinds of clothes like skirts or scarves.
- An alien creature covered with soft spiky "feelers," bouncing as he walks.
- Breasts! Boingy, boingy, boingy.{Obscure anime reference, I'm sorry... (Golden Boy, Episode 1)}
- Car antennas.
- Bouncing springs.
- Animal and insect tails, antennae, feelers, or whiskers.
- Particle turbulence. A point cloud object with WhirleyPoints applied will flow and sway as it moves, rotates, or scales.
- Loose threads hanging off clothes.
- Space ship contortions as it cloaks and uncloaks.
- The flagella of bacteria and paramecia.
- Smoke rings wobbling through the air. (HyperVoxels makes great smoke.)
- Rubbery flowing flying logos.
- Shockwaves traveling through a building or warship.
- Butterfly wings flapping.

5.8 WhirleyPoints Example Scenes

Whirley_Simple.lws

This is the scene I use most for testing. It features slightly jerky object motion, recovery pauses, and a final rotation. This makes it easy to experiment with both Acceleration and Centrifugal Sensitivity. It's a good scene to use just to familiarize yourself with WhirleyPoints's controls.

Whirley_Fall.lws

A simple scene showing a disk reacting to acceleration.

Whirley_Flow.lws

A scene that shows WhirleyPoints applied to a named surface only. The honey-like flowing comes from the use of Motion Lag. Experiment with setting the Acceleration Sensitivity to 0% to see how a small amount of dynamics enhances the stretching lag effect.

Whirley_MaybeNot.lws

Clothes or hair on a person flop and sway as they move. This scene shows a simple head animated with a bone. The hair{For this demo, the hair is just simple polygonal lines. Photoreal LightWave hair requires other techniques.} is a separate object with keyframes that follow the head motion. WhirleyPoints makes the hair sway realistically.

This example uses a low Local Y Strength to reduce vertical stretching. The X and Z motions are important for swaying, but too much Y makes the hair look like stretching rubber.

The effect is concentrated at the hair's tips using Bias and Sharpness.

You can make much better hair and clothes motion using Motion Designer, especially when collisions are important. However, WhirleyPoints is still often a good alternative because it's quick, robust and interactive.

Whirley_Sidewinder.lws

The Motion Lag option will make objects bend as they go around curves. This gives a cute, flowing motion. This scene shows a simple sphere flowing as it moves along a zigzag path.

Whirley_Tail.lws

Tails or any rubbery kind of tentacles can be animated with WhirleyPoints. Hoser can give you exact control of a tail, but WhirleyPoints can make one which loosely flops automatically.

In this scene, a tail{Well, it's really a tentacle, used for Hoser's example scene. Just pretend it's a tail.} lazily waves. The interesting motion comes from Centrifugal Sensitivity.

The main length of the tail is modeled along the X axis. A low Local X Application Strength kept the tail from overstretching. If you set it back to 100%, you'll see the tail stretch when the object rotation flings it quickly.

Whirley_Twister.lws

This is a very pretty example showing the effects of Motion Lag and some dynamics. A simple square can make some elegant shapes!

Whirley_Windy.lws

Motion Lag alone can make an object twist and fold as it moves. This scene makes a simple square look like it's being blown by the wind.

5.9 WhirleyPoints FAQ

Question: My surfaces aren't being moved by WhirleyPoints no matter what settings I use!

Answer: WhirleyPoints is very smart about surfaces and doesn't move the parts of your object at the join of applied and unapplied surfaces. The strength of the motion increases as you move away from this seam. This allows the tip of a tail to wag freely, but keeps the base connected to the body. If your object is malformed, you won't get the proper effect since there's no seam at all. Common problems are unmerged points, or multiple copies of polygons on top of one another. Clean up your object with Modeler's Merge Points function and you'll probably fix any odd motion problems.

Question: Whirley points doesn't work when I apply a motion plugin to the parent of a Whirley object. Why not?

Answer: This is a LightWave limitation. LightWave's FSPE mode needs to be active. Unfortunately, FSPE is slow and unreliable.

Question: On the PC, how do I keep the OpenGL preview from opening on top of the WhirleyPoints interface?}

Answer: The main WhirleyPoints panel isn't movable after it's opened. If you move the LightWave window to one side of your screen before opening it, the WhirleyPoints panel will also open to the side of the screen. This will give you more room to move and resize the WhirleyPoints OpenGL window. The Mac has the OpenGL preview integrated into its panel, so it does not suffer from this problem.

Question: Why can't I see all of my objects in the OpenGL preview?

Answer: Taft only shows the single object it's applied to. It would be great if it showed more, but this slows down the preview considerably, and it's also technically difficult for the plugin to "fetch" all of the object data necessary to draw the whole scene (especially in LightWave 5.6.)

Question: When I change my object in Modeler and export it back to LightWave, strange things happen. Why?

Answer: A LightWave bug causes plugins to be told the wrong information about an object after it's been exported from Modeler.{Specifically, LightWave doesn't report the current object filename, only the previous one.} The easiest workaround is to use LightWave's Save Object button, which will correct LightWave's confusion.

Question: How do I apply a swaying effect to my whole object without stretching and deforming it?

Answer: Use the Apply to Entire Object option with 0% Effect Bias and 100% Effect Sharpness. This will make your object move and twist as a solid unit.

Question: Why aren't one point polygons affected by WhirleyPoints?

Answer: You can't use individual surfaces to select the points. When WhirleyPoints is in the surface-selection mode, it applies the effect to points more strongly when they're farther from the join of that surface with the unaffected surface. This makes the base of an animal's tail unaffected, even though the tip sways a lot. Isolated points, *have no connection* between the surface and the rest of the object, so WhirleyPoints doesn't know how strongly to move those points. However, isolated points work fine as long as you use "Apply to Entire Object" mode.

5.10 History

WhirleyPoints began as an experiment. In early 1999, I was designing fast fur dynamics algorithms for my *Sasquatch* plugin. The dynamics had to behave smoothly and realistically, but computation speed was also important because it would be applied to millions of hairs simultaneously.

I implemented several different approaches using IK chains. It wasn't too hard to design a tool that looked great. It was also fast, but still not fast enough; with 10 million moving hairs, the computation time was several seconds, destroying interactivity.

I read a research paper from SIGGRAPH 1999 by Doug James about deformable objects, and I implemented his technique. It wasn't designed for hair and it was too slow to use on millions of strands, but its effects were still promising. I kept thinking about some of his interesting ideas.

I realized that I could modify his technique in a simple way which would increase computation speed *and* add full dynamics. This trick uses an analysis of the object's shape and determines how points will respond to accelerations beforehand. In the real-time simulation, the results of this precomputation can be applied in real time.

To test the new technique, I decided to make a quick LightWave tool to preview the motion by simply loading models and watching how points moved. The first version of my algorithm worked better than I had hoped, and I spent a week or so tweaking and enhancing it.

Steve Hurley,{No relation to me. His name is so similar to mine that we often have confused users.} our inhouse artist, saw these cool tests and asked a question that had never occurred to me. "This test tool is useful just by itself! Why don't you make it a plugin?" He was obviously right.

The plugin originally had the clever name *Test* , so I renamed it to *Dynamation* until I remembered a few hours later that *Dynamation* was the name of an old particle system for Wavefront. Oops! I decided on *DynaPoints* instead, but Mr. Hurley liked *TwirlyPoints* better. I was about to change the name one last time until he yelled *"WhirleyPoints!"* and we both started laughing.{Hoser's naming has a similar story.)

The biggest change to WhirleyPoints after its baptism was the addition of the OpenGL preview. Once the preview was added, the tool became much more useful and extremely fun to play with. Even I was surprised at the interesting motion you could make an object produce. Without the preview, it's much more difficult to experiment.

I experimented with collision detection, but this slows the algorithm down immensely, ruining the real-time feedback. I then learned that Motion Designer by Diasuke Ino{Ino-san is one of the most talented graphics programmers I know.} was going to be included in LightWave~6, and I was happy. This gives WhirleyPoints a nice "niche" as a quick, fast, interactive dynamics tool, with Motion Designer being more powerful but more awkward to use.

When we were making demonstration scenes for this manual, we wanted to make a "stretching taffy" effect for the flying logo example. I added the Motion Lag control to make a simple way to stretch and flow an object in addition to the dynamics simulation.

I gave four or five studios a copy of WhirleyPoints, and they all immediately loved it. I didn't make many changes even after the studios gave their feedback; the tool seems to have a good balance of ease-of-use and capabilities. It's a tool which can add a lot of small interesting motion to an object without much effort.

Chapter 6 Tracer

Tracer is a tool designed for a single purpose: automatic weapons fire.

It's not difficult to make a gun or laser fire in LightWave. You can model a projectile and keep it hidden inside the gun until you need it to fire. Then you keyframe the shot moving outwards, possibly bringing it into view with a dissolve envelope. It can be a little tricky to set the rotation to make the shot align with its motion, but it's not too hard. Once you've made the model, you can make it fire any way you wish in two or three minutes of straightforward setup in Layout... And that's the problem.

It takes two or three minutes to shoot an object. But what if you have a space fighter that's firing 100 laser blasts? Ugh! Even if you cut the time for each shot down to a minute of setup, it's still almost two hours of tedious setup! And then if you change your scene... you have to redo it all over again.

Tracer solves this problem. You can spit out thousands of projectiles automatically. It doesn't do anything you couldn't do by hand, but it automates the work for you, so it's trivial to make gun battles. Even better, you can change your scene and the gunshots will automatically adjust themselves.{Most animators *haven't* had to make gun/laser/missile shots. But the ones that have know just how big a problem this really is, and will be dancing happily when they start using Tracer.}

Tracer adds laser/bullet/streaky blob projectiles to your rendering automatically. It renders the shots as volumetric glowing projectiles. The color and shape of these shots is extremely customizable.

Tracer is a pixel filter which adds moving gunshots to the rendered LightWave image. This has the big advantage of not needing any extra LightWave geometry and allowing unlimited numbers of shots.

6.1 Open Fire!

To add Tracer, first choose an object in your scene to be the source of the shots. This can be a gun object, but often it's convenient to make a null object and parent it to your gun. This lets you specify the exact origin of all of the shots, which is usually the tip of the gun, at the muzzle. The shots will be launched starting from this object and move in its +Z direction. Add Tracer and open its interface.

The first control is **Gunbarrel Item** which identifies the source of the shots. By default, the shots will be launched based on the gun's +Z direction. This lets you animate a gun's direction and have its shots fire appropriately. However, sometimes you'll want to make sure the shots hit a specific target. It's hard to exactly aim a gun to hit something, but you can tell Tracer to force the shots to fire towards a specific target by using the **Aim Towards** control. The shots won't lead a moving target, but you can use an animated null as a target to manually lead them, or make near misses.

The accuracy of the gun is set by **Aim Accuracy.** A lower value will make the shots fire in less precise directions. This gives natural, imperfect aim to the gun. You can use extremely low accuracy to make shots fire in all directions, even backwards.

The **Fire Rate (rounds/sec)** control specifies how often the gun fires. This can also be animated, allowing you to fire bursts of shots. The most common use of Tracer is to fire a single burst of shots. You could animate the fire rate, but for this common case, you can use a simpler method which doesn't require animating the rate. The **Absolute Start Fire Frame** and **Absolute End Fire Frame** allows you to restrict when the gun will fire. If you wanted to fire a single burst of three shots starting at frame 10, you might set the Absolute Start Fire Frame to 10, and the Absolute End Fire Frame to 13. With the default fire rate of 30 rounds per second (one round per frame), three shots will be fired.

Speed (m/s) defines how fast each shot moves in your scene. **Maximum Distance** specifies how far each shot will travel before it disappears. You can use a very large distance to prevent the shots from ever disappearing. However, it's often good to limit the range just so that Tracer doesn't have to compute many offscreen shots, which slow down its rendering. If you use the Aim Towards control to specify a specific target, the shots will disappear when they reach that target.

Shot Size (m) specifies the *width* of the shots.

6.2 Appearance Controls

The controls in the center of Tracer's interface defines the *appearance* of the fired shots. The number of controls may be intimidating at first, but actually they're quite easy to use especially since there's a realtime preview.

Each shot's appearance is defined by up to four blobs of color. You design the bullet/laser/streak/shot appearance by stretching, shifting, and coloring these blobs. This gives you a huge variety of appearances, from glowing spheres to laser bolts to flamethrowers.

Most shots you make will be long streaks. Each of the blobs that form the streak has its own settings. The first is **Size (as %of beam)** , which lets you make the blob bigger or smaller. The **Position Shift** control lets you move the blob to the left or right. The **Aspect Ratio** control makes the blobs circular or elliptical. By combining these three controls, you can move, size, and position the blob to build the streak's shape.

Each blob has its own set of colors. The center and edges of the blobs can be colored independently with the **Center, Edge Color** controls. This makes a shaded gradient, which can be changed with the **Color Transition Skew** value. Finally, you can make the blobs partially transparent using the **Center Opacity** and **Edge Opacity** controls.

These controls may initially seem overly complex, but they aren't very difficult. Use the preview to adjust the values and you'll quickly see the effect of each control. A final hint; if you want to eliminate a blob completely, just set its size value to 0%.

6.3 Additional Options

At the bottom of the Tracer panel are a final set of miscellaneous features. **Add to Alpha Channel** will make Tracer's shots appear in your rendered alpha channel if you're compositing later.

Tracer shots do *not* motion blur unless you activate **Motion Blur Shots**. Most shots look better unblurred since LightWave's motion blur algorithm doesn't work well with fast moving objects. This option removes the blur from *shot* motion, but retains any *camera* motion blur.

Shoot Only at a Frame Start prevents the gun from firing between frames. This changes the appearance of the fired rounds, since every shot will appear right at the gun muzzle the first time it's visible. This is especially important when you don't use motion blur. Without this option, the shots may appear to be launching from a point in front of the gun, not from the gun itself.

Export Light Envelope allows you to make muzzle blasts using LightWave lensflares. This option allows you to specify the light's intensity behavior. Plugins can't affect a light's intensity directly, but Tracer can export an envelope which you can load into a light manually. If you parent the light to your gun object, the light will flare each time a shot is launched.

The export button will open a small panel where you define the light's behavior. When a shot is fired, the light will increase to a maximum intensity, and decay back down. **Ramp Up Frames** and **Ramp Down Frames** specify how quickly the light grows and fades in intensity. The **Time Offset** control will let you advance or delay the *time* of the light's flare. The effect usually looks best when the light flares just *before* a shot is fired, using a small negative offset value. The **Shot Light Intensity** is the brightness of the light when the gun is firing, and **Base Intensity** is the brightness when the gun is *not* firing. The **Additive Shot Lighting** button will make brighter flares when you're firing rapidly. Finally, you can choose how many envelope frames to save using the **Maximum Frame** control. When you're happy with the settings, the **Save Envelope** button will bring up a file requester to chose a name for your saved envelope file. Once you have done this, enter the light's intensity envelope and use LightWave's Load Envelope button to load the file you've created.

Finally, you can temporarily disable Tracer in LightWave by using the **Disable all Effects** button.

6.4 Limitations of Tracer

Tracer isn't perfect. Since it's a pixel filter, it doesn't add LightWave geometry. But this also means that LightWave won't *react* to the Tracer shots. The shots will not be shown though transparency or seen in reflections. They don't cast shadows. They don't appear properly when LightWave is in stereoscopic mode or with Depth of Field. They also aren't obscured by fog.{Wow, a definite laundry list of limitations. In practice they're all minor except transparency and fog. I hope to add those abilities in a future patch.}

You have a lot of control over the appearance of the shots, but you're still limited to a glowing appearance, since the shots aren't affected by LightWave lights or shadows.

6.5 Tracer FAQ

Question: Why does Tracer take longer to render as my animation proceeds?

Answer: "Tracer has to compute the position of each shot, including the ones that are offscreen or have already hit their target. As a gun fires, more and more shots are created and each one adds a small amount of time to the rendering process.

Question: My shots are coming out, but they just hang in the air and don't move!

Answer: Actually, they probably *are* moving. But if you're firing 30 shots every second, each frame will add a new shot exactly where the previous shot used to be. Since you can't tell the difference between the shots, it looks like one immobile shot. Mathematicians call this effect "temporal aliasing." Filmmakers call it the "wagon wheel effect" from old Westerns. You can eliminate this problem by making sure the gun or target moves, by using a slightly different fire rate, or firing only a short burst instead of a continuous stream of fire.

6.6 Tracer Example Scenes

Tracer_DesignerAmmo.lws

Four examples of ammunition design. Frame~12 shows a good view of some of the varieties of ammunition you can create.

Tracer_FireRate.lws

A gun's fire rate can be animated, usually using a keyframed null object. This scene shows how variable fire rates are set up. You may want to render a low resolution animation of this scene to see the effect.

Tracer_FlameThrower.lws

Soft, diffuse shots blend together and make a fuzzy stream. This looks especially interesting when the gun or target is animated, causing a curved path as the gun barrel sways back and forth.

Tracer_Incoming.lws

Tracer's shots can be aimed at the camera. By lowering the Aim Accuracy, the shots will pass by the camera on all sides. This scene shows long, thin shots streaking by, almost like stars in a warp spaceship's display screen.

Tracer_Snowfall.lws

Tracer's effects don't always have to be weapons fire. This scene has a "gun" positioned well out of view above the camera. It's rapidly firing slow-moving white shots downwards. The Aim Accuracy is very low, so the shots spread widely and cover the entire field of view. This produces an effect like falling snow without using particles. You can use a similar technique for rain or even a meteor bombardment.

Tracer_Targets.lws

When you use the Aim Towards control to specify a target, the shots will disappear when they reach it. This scene shows four independently aimed guns. Frame 60 shows the laser bolts stopping at each target.

6.7 History

I visit large and small studios year-round for many reasons. I am not a very skilled artist myself,{In fact I don't know how to use some parts of LightWave, especially in Modeler.} so it's important for me to understand how active LightWave artists work and what they *want* . I ask a lot of really dumb questions, talk a lot, and listen even more. This is how I learn what new plugins are needed, and also how to improve my current tools. Feedback is essential!

In April 1999, I spent a week working at Foundation Imaging.{I worked *at* Foundation, not *for* Foundation. It wasn't for pay; it's an important way to meet more artists and learn more about LightWave production. } They're the largest LightWave studio, and there are always many exciting projects happening. During this visit they were just starting work on the *Starship Troopers* series.

John Allardice was particularly eager for help for several vexing problems. One of them was quite simply "We need to make gunfire. A *lot* of gunfire. Help us!" In fact, I had heard similar requests previously; John Gross of Digital Muse once fantasized for a tool to make automatic phaser shots for Star Trek.

After talking to John and other animators, we discussed what a gunfire plugin would need. Then I disappeared into a quiet corner and designed the appropriate algorithms in a notebook.{This is always my favorite part. I'm not the best programmer, I'm really an algorithm guy.} Then it was straightforward to turn those notes into a working plugin which I named "Tracer."

John and the others were surprised and happy and started using the tool immediately. Over the next few days I made other minor tweaks. In late 1999, I prepared Tracer for Taft by polishing the interface and adding a few extra features including Aim Accuracy.

Tracer could be extended in many ways. You can imagine an entire program designed only to make gunfire effects! I have several pages of notes of ways to extend Tracer to make it even more useful.

Chapter 7 Sticky Front Projection

LightWave has a useful image mapping feature named Front Projection. It makes an object's surface color exactly match a background image, regardless of the object's shape or position.

This feature has a lot of applications, especially when compositing LightWave objects with photographs. Since the front projection mapping makes an object's surface exactly match the background, the object will effectively disappear from view.{Front projected images often use 100% Luminosity and no Diffuse to make the surface show the exact color of the mapped image. Our *Gaffer* plugin is also extremely useful when adding shadows to front projected, luminous objects.} Other objects can still be obscured by the front projected object. Effectively, your background becomes 3D, allowing you to make an object appear in front of or behind something which exists only in a photograph.

Taft's Sticky Front Projection plugin (StickyFP for short) is much more powerful than LightWave's front projection. The most obvious difference is that StickyFP is *sticky*. Like LightWave's Front Projection, the surface color matches your background image perfectly. However, StickyFP allows you to move the camera or the object and the map will *stay attached* to the surface. This allows for a huge variety of special effects.

7.1 Define Your Map

It's easy to set up StickyFP. Load the image you want to map using LightWave's Load Image button (LW~5.x), or the Image Editor's Load button (LW~6.x). In the Surface editor, apply StickyFP to the surface you want to map and open the plugin interface. Use the **Color Image Map} button to choose the image.**

You can apply transparency with StickyFP by selecting an image with the **Transparency Image Map** picker. This allows you to use a greyscale image to make parts of your object transparent. It also turns off specularity in the same areas. You can reverse the transparency effect with the **Reverse Transparency** button.

Front Projected transparency has a lot of applications, since it makes your object invisible in areas painted by transparency map. This is an easy way to make an object match the shape of something in your image.

For example, you may be trying to make a fence which shows gaps between the fence rails and posts. You could try to build a perfect model, making polygons which match the exact shape of the fence. But you can make a pretty good fence much more simply, using only a single polygon. Use a paint program to make a map which is all white except where the fence is. Apply this to the flat polygon with StickyFP's Transparency Image Map option, and the original photo as a Color Image Map. The single polygon will now look and act like a photoreal fence. It won't have any depth or detail that's not visible in the image, but often the effect is flawless.

7.2 Stick it to Me!

By default, frame~1 will show the exact same effect as LightWave's Front Projection. If the object or camera moves in later frames, StickyFP *remembers* the mapping from frame~1 and keeps the image stuck to the surface. It can be useful to choose a different frame to match by using the **Defining Frame** control to choose the frame number. On that frame, StickyFP will act like normal LightWave front projection mapping. The image will be stuck to your surface in frames before or after the Defining Frame.

Other simple abilities of StickyFP allow you to *shift* the applied image in both X and Y directions using the **X,Y Pixel Offset** controls. You can also use **Application Strength** to fade the map's opacity.

You can disable StickyFP temporarily with the **Disable All Effects** button, which is especially useful in LightWave 5.6 which has no method for disabling plugins without removing them.

7.3 Freeze That Map!

StickyFP mapping uses the camera's view to decide the mapping of the image. But often you're simply trying to surface a model for later use somewhere else, and it would be annoying and impractical to require you to move the camera to some magic location to set up the map in each scene that uses that object.

StickyFP has a button which solves this problem. It computes the current StickyFP surface mapping and *freezes* that map to the surface, even if you use the surface in a different scene. The camera *no longer* defines the mapping.

To freeze the mapping, first render a frame to make sure the image is applied properly. Then in the StickyFP interface, simply press the **Freeze** button. Once this mode is activated, the mapping will be remembered and you're free to move the camera and object on *any* frame. There's no "defining frame" anymore since the map is remembered from the setup when you pressed the button. You can even load the object in a brand new scene and the mapping will stay attached.

When you're defining maps in this way, I strongly recommend that you save a copy of the scene you use to define the map! If you ever have to change the mapping, you'll need that scene to recreate the object and camera positions. StickyFP will put up caution messages every time you activate or deactivate the Freeze mode, because you'll lose your settings if you're careless.

7.4 Photoreal Modeling and Front-only Application

StickyFP can be used as a tool to help model and map objects using photographs. This idea is simple but *very, very powerful.* It can completely change the way you use LightWave! The basic method is to build simple geometry and use StickyFP to map a photograph onto the surface. The rich detail of a photograph hides the simplicity of the underlying geometry. This is especially effective with buildings, signs, fences, walls, streets, grass, and water, since most of these can be modeled as a flat plane or box with a high quality image map.

It may seem silly to model a photoreal building as a featureless box, but the realism a photographic texture adds will make the box superior to even a carefully designed, high-polygon model. And it's a lot easier!

Taft's Camera Match plugin is often *crucial* when applying these kinds of maps. If you have a photograph of a building, you'll need to know the position of the LightWave camera that will make your simple box model match the position, size, and orientation of the photographed building.

One useful option for mapping models is the button labelled **Apply to Front Only**, which tells StickyFP to only apply to the object polygons which face the camera at the time the map is defined. This lets you define the front and back polygons of an object differently without having to manually separate the object into surfaces.

For example, if you were trying to map a photoreal book, you would build a rectangular object for the book's geometry. You'd take two photographs of the book, one showing the front, the spine, and the top of the book, and the other photograph showing the back, the edge, and the bottom of the book. Notice these two photographs show all six sides.

Then you would make a scene which matched the camera's view of the book to the photograph of the front of the book.(For this step, the Camera Match plugin is invaluable.) StickyFP will then attach that image onto the book's surface. The problem is that the front cover image will project all of the way through the book, and also apply itself to the back cover. Eeek!!

But if you activate Apply to Front Only, the map will not apply itself to the rear-facing polygons. So when you map the rear faces, you can again use the Apply to Front Only button to prevent the map from covering the whole object. The Sticky_Antique demonstration scene explicitly shows how to define this kind of mapping.

One limitation to the Apply to Front button is that it doesn't work when the object is displaced, deformed, or boned. Apply to Front uses the object's surface normals to determine whether that part of the object faced the camera or not, and displacements will change the normals. This option is therefore useful for books, cars, buildings, and furniture, but not for a lion's body, trees waving in a storm, or particles (which have no front anyway.) For those objects, you should simply make new surfaces in Modeler to specify where the map is applied.

7.5 Crude Standin Modeling and Back Transparency

Sometimes you'll want to make a simple background object from a photograph. A common technique to make objects from photographs is to make "billboards," flat polygons with an image applied to them. Seen from a distance, these can look quite good. Their biggest limitation is that their flat shape is apparent whenever the camera moves. StickyFP can help make these kinds of objects feel more like a 3D object instead of a flat map.

For example, you might want a pile of photoreal boulders in your scene. You photograph a wonderful roundish rock in your backyard. Unlike a book, you can't measure the shape of a rock perfectly, so how do you build it? One way is to *fake it!* Build a rough, convex, rock-shaped model. Move, size, and rotate it in Layout so that it just barely covers all of the image of the rock you photographed. Now apply StickyFP.

The mapped rock won't work well, since it has a rim of pixels from the background on it! Paint an alpha channel in Photoshop, marking the background, and use this as the transparency map for StickyFP. This will make the rock geometry show only rock, and no background.

Finally, activate StickyFP's **Make Back Transparent** button. This button makes all rear-facing polygons transparent. This will remove the back of the rock, so it won't show up when you see the side or rear of the rock. If you didn't have this option, you would see *two* thin eggshell layers, one for the front of the sphere-like object and one for the back. If we just show the front, it appears more natural. The curved shape of the mapped part of your object gives it true 3D depth, so it's more convincing than a flat billboard.

This method of building objects is very crude. The geometry doesn't match perfectly, and it breaks down when you view it from an extreme angle. However, it *does* give a simple 3D perspective which is much more realistic than a billboard. And you can build this crude object *extremely* quickly, with no measurements or precise camera matching needed. The Sticky_Transparency.lws demonstration scene shows the advantages (and limitations) of this crude but effective technique.

7.6 The Advantages. The Limitations.

LightWave's built in Front Projection has several problems even using simple mapping. It does not show up in reflections or refractions. It doesn't react to motion blur or depth of field. StickyFP doesn't have any problems with these situations.

The biggest limitation of StickyFP is that the mapping is based on the *undistorted* shape of your object at the defining frame. If bones, morphs, or displacements are active on the defining frame, the front projection won't be accurate; it will be mapped as if the object was undistorted. This comes from a LightWave limitation which prevents plugins from learning about the shape of an object on a different frame.

LightWave 6 has a "Fixed" option for its front projection tool. However, its effect is very different than StickyFP, since the mapping doesn't actually stick during displacements or object motion. It also can't be used for modeling, since there's no freeze or front/back application options.

7.7 StickyFP FAQ

Question: Why does the texture change when I move my pivot point?

Answer: Moving an object's pivot point changes the way LightWave computes object positions. This offsets the StickyFP image mapping too. You can use a null object as a parent instead of changing the pivot point to get around this problem.

Question: How do I simultaneously map multiple sides of an object, like an actor's face, ears, and the back of his head?

Answer: Make your head with two (or more) surfaces. Each can be mapped with its own copy of StickyFP. If your object is not going to deform, you can also use the Apply to Front Only button and use two copies of StickyFP to make separate front and back mappings on a single surface.

Question: I cut and pasted a surface onto another object, but I don't see StickyFP's image on the new object! Where did it go?

Answer: StickyFP uses the position of the object and camera to define the mapping. If you transfer these settings to a different object, they probably aren't suitable for that new object. So if you use LightWave's preview sphere or the StickyFP L/S buttons to copy settings to a new object, you probably won't get anything useful.

Question: Why don't I see StickyFP on LightWave's Sample sphere or cube preview?

Answer: StickyFP bases its mapping on your real object's shape and the camera position. The Sample sphere doesn't have a camera, so showing StickyFP doesn't make much sense.

Question: Image Sequences don't work when I reload my scene!

Answer: LightWave doesn't fully support the use of image sequences by plugins. (Specifically, LightWave has no easy way to have a plugin *load* an image sequence when the scene is reloaded.)

7.8 Many Possible StickyFP Applications

- Use StickyFP for anime style cel animation. StickyFP can pan, zoom, and rotate a background matte painting. Foreground characters and objects can be flat polygons mapped with StickyFP.{ *South Park* uses this animation method, though with Alias/Wavefront Maya.}
- These techniques can also be be used with normal LightWave flat mapping, but StickyFP gives the advantage of pixel-accurate alignment when you have to match other elements outside of LightWave.
- Subsurface effects. Use StickyFP on a layer of polygons, then use bones or morphing to make the surface move. This can make skin bulge as if there's something crawling underneath it, grass or pavement heave upwards from an underground blast, or a metal cage dent outwards as angry monsters pound the wall inside. All of these effects are photoreal because you're starting with a photoreal map.
- Low polygon modeling. An object mapped with a photograph often only needs a few polygons to define the rough shape of the object. The map's detail adds the realism.
- StickyFP a photograph to a polygonal ground plane. Animate an earthquake! Distort, move and break apart the ground and the grass, dirt, sidewalks will stay attached to the ground.
- Explode things! Make a crude object and use StickyFP to make it match a real photograph. Now explode your object and your photoreal rendered version will also explode.
- Animating the Defining Keyframe makes an interesting active-camouflage effect. If the defining frame is
- Many 2D animated photograph effects. Manipulate portions of images using polygons mapped with StickyFP. "Monty Python" style animation is extremely easy!
- Surgery simulation. Use StickyFP to map a photograph of a torso or arm onto polygons. Now you can animate the polygons being cut and retracted. You can have a *second* layer of polygons underneath which is *also* mapped with StickyFP, showing a photo of muscles, organs, bones, or whatever. This is a very easy, photoreal way to dissect a body.
- Forget surgery, use the same technique to chop up bodies or make realistic gore-filled wounds.
- In any kind of diagram/explanation, you can "peel away" part of a photograph to show something underneath. One example is to take a photograph of a building. Use StickyFP on a polygon to match the building. Then peel the polygon away, revealing another layer underneath which shows something like a floor layout. This is a very effective way to connect a schematic, blueprint, diagram, or simplified view of an object with its photoreal appearance.
- One studio used this technique for an accident reconstruction, lifting away a car's roof to show where the occupants were sitting during a crash.

7.9 StickyFP Example Scenes

Sticky_AntiqueShow.lws, Sticky_AntiqueMapper.lws

This pair of scenes shows the effectiveness of using photographs to texture map an object. The first scene, Sticky_AntiqueMapper, uses two copies of StickyFP to place two image maps onto the surface of an antique

cabinet. The cabinet object was modeled to match the real object's dimensions. Camera Match was used to determine the camera locations used for each photograph. The camera is placed to map the front and left side of the cabinet at frame 0 and the back and right of the cabinet at frame 1. If you enter the StickyFP interface, you'll see that the Freeze button is already activated. You can deactivate it and render, since that frozen pose was defined using this scene. Once frozen, though, the object can be used in *any* scene.

The Sticky_AntiqueShow scene shows how frozen StickyFP maps can be used in later scenes. The antique is loaded multiple times and you can see the cabinet in many different poses simultaneously. There's no restriction on moving the camera, or moving, rotating, or scaling the cabinets. This shows the power of photo mapping your objects!

Sticky_Breakup.lws

This is a straightforward demonstration of StickyFP. Multiple objects share a single StickyFP surface. Even when the objects move apart, the front-projected image stays attached. Render the final frame to see the sticky effect.

Sticky_Transparency.lws

This scene shows how even crude geometry can give a background object depth, even though the object is extremely simple and doesn't really match the object shape carefully. In this scene, a photograph of a car is applied to a flat plane, a sphere, and a very crude, rough, car-like shape. The transparency map option of StickyFP is used to "cut" away the extraneous parts of the object.

The scene shows the objects rotating in place. You can see the advantages (and limitations) of the technique. In particular, the flat billboard object has no depth effect at all, so it doesn't appear realistic if the camera moves very much. The other objects have a much more 3D behavior. At extreme angles, they're far from perfect (you can see distortion especially in the car tires) but small angles give a convincing effect. These models are *extremely* quick and simple to build, and for that reason alone the transparency technique can be useful. True photoreal objects should be built with correct geometry and camera-matched mapping, as in the Sticky_Antique scenes.

Sticky_Compare.lws

LightWave's Front Projection mapping has several limitations. This scene compares LightWave Front Projection, on the left, to StickyFP on the right. StickyFP appears correctly in reflections and refractions.

7.10 History

In 1996, Tom Williamson{Tom and Dave at Computer Cafe also financed and inspired a lot of other Worley plugins.} at Computer Cafe contacted me about a shot he was making for a science fiction movie.{Released as the surprisingly fun kids movie, *StarKid.* } A character had to be turned into a cloud of animated particles. The particles were easy, but he needed to texture them to match the original pose of the actor. A flat image map *almost* worked, but there was a noticeable mismatch between the actor's shape in the instant he turned into particles.

StickyFP solved this problem immediately, since the particles matched the original plate perfectly, and the transparency option was able to turn the extra particles outside the actor's body invisible. When the particles were animated, their colors stayed constant so you could see the body dissolving.

A lot of studios used this early version of StickyFP for several years, essentially with no modification. Ken Stranahan, a very, very talented freelance artist was a particularly enthusiastic user. In 1999, I started to polish the plugin for Taft's release. We wanted to make a demonstration photoreal flyby scene using a building. Easy, right? Just make crude polygons, match the view, and map them.

It was then that we realized how difficult matching was! We wasted a *full day* trying to match a single frame. It was terrible. Now, finally, I understood just how badly artists were hurting for a matching tool! Luckily I had already written a matching prototype, so I dusted it off and readied it for Taft. This enabled StickyFP to evolve into a truly powerful tool.

In January 2000 I added the Front and Back application options, which saved a lot of time. I removed a "standin" option which let you use any object, not just the camera, to define the mapping. But I replaced it with the Freeze button, which was both simpler and much more powerful. That changed \hbox{StickyFP}

and made it useful for modeling, not just front projection tricks, since you could simply load and use an object without worrying about cameras, defining frames, or anything.

StickyFP is definitely well appreciated by artists. When I demonstrated it at one studio, over 20 people literally cheered when they saw a demonstration of the antique chest. It really can change the way you make objects for LightWave!

Chapter 8 HeatWave

Heat makes air shimmer and sway, distorting your view through it. It's usually not noticeable unless you look at hot pavement or a desert.

HeatWave is an image filter which approximates this effect. It's really a simple image processing distortion, but it is often effective especially when subtly used on a background.

HeatWave's animated shimmers are particularly effective, since the heat waves will slowly rise upwards as well as change their appearance.

8.1 Heat this Place Up!

HeatWave is the easiest plugin in the Taft collection to use. Add it to your scene in the Effects panel (LightWave~5) or Image Process panel (LightWave~6). HeatWave only has a few simple controls, and its realtime preview makes most adjustments simple.

The **Horizontal Waver** and **Vertical Waver** adjust how much your image will sway and distort. Aesthetically, the effect tends to look better with greater horizontal movement.

Scale Size and **Small Detail** change the appearance of the distortion by making the waves vary over larger or smaller areas.{This is hard to explain with mere English, but luckily you can simply see the effect for yourself in the preview. Whew!}

HeatWave's ripples will automatically animate to make the image shimmer. You can change the speed of this shimmering with the **Animation Rate** control. The ripples will also rise upwards. The **Rise Rate** control will let you select the rise speed or even stop it.

By default, HeatWave will apply to all of the objects in your scene as well as the background. You can keep the background undistorted by deselecting the **Apply to background** button.

If you only want certain surfaces to shimmer, you can flag them to be treated specifically by HeatWave. The option will make only the flagged surfaces shimmer. To do this, first select the **Apply to flagged surfaces** button in HeatWave. Next, in LightWave's Surface panel, visit the surfaces you want to restrict HeatWave's effect to. Use the Special Buffer button to enter a value between 0 and 1.0. A smaller number will apply a weaker HeatWave effect, with 0 meaning no effect and 1.0 full effect. (In LW 5.6, the values range between 0 and 255.)

8.2 HeatWave Application Ideas

- The exhaust of a jet aircraft, rocket, or vehicle tailpipe.
- The effect of a fire, explosion, or blast.
- Weapon effects. Maybe a "stunner" will make the victim's image waver.
- A spaceship cloaking field.
- Video interference.

8.3 History

HeatWave is quite old. I believe I wrote the first version in early 1995 for my friend, Andrew Denton. He was making an animation of a flying dragon for SIGGRAPH's Electronic Theater and wanted several effects to give the animation more detail. One of the effects was heat waves. I spent a few hours making a tool which wiggled image pixels to the left and right.

That simple tool got added to a growing collection of "proof of concept" tools. Many Hollywood studios had copies of these tools, which included HeatWave, but I never got any feedback on it. I didn't expect any, since it was such a trivial tool.

When I finished Polk in 1998, I meant to polish HeatWave and include it in the bundle, but Polk was already loaded with over 30 plugins, so I never did. After I released Polk, the studios now had a new, polished,

collection to replace all of those test plugins. Soon after, I started getting phone calls: "Where's HeatWave! Hey! We still need it!" Apparently HeatWave still had its uses in studios, even though it was so simple.

For Taft, I polished HeatWave by adding new animation options and a preview. It's still an awfully simple tool, and limited in many ways, but it is effective at adding shimmers.

My future plans for HeatWave are to add a new way to apply the effect instead of via surfaces. I should allow "growing" the heat region, softening it with a blur, or maybe even allowing an image to define where it's applied. That would make applying it to limited regions a lot simpler.

Chapter 9 Mosaic

Mosaic is an image filter which pixellates LightWave's output images. This makes the image "blocky," like a low resolution image that you've magnified until you can see individual pixels.

This pixelization technique is seen on television where it is used to obscure a secret informer's face, a license plate number, or the naughty bits of someone showing a little too much of themselves to the camera.{You can sometimes see the mosiac effect in late-night, edited-for-cable adult, ah, "art films."} The Mosaic plugin isn't limited to this simple effect, though it's the one you see most often.

Mosaic has a realtime preview in its interface. It uses the last image you rendered to show the effect of your current settings. You can see more detail by using LightWave's Limited Region tool to zoom into smaller areas.

9.1 Using Mosaic

The **Width** and **Height** controls set the size of the blocks. They are percentages, so a width setting of 5% will make blocks which are 5% of the width of the image. Lower values make the blocks smaller, and therefore shows *more* of the original image detail. The width and height can be set independently, so you can make tall but thin blocks, not just square ones.

There are four types of mosaics. The default, **Grid**, is the usual "giant square pixels" effect you usually see on TV. It makes rectangular blocks. The **Radial Mosaic** option makes a similar effect using rings of pixels, not rows. These rings are similar to the pattern that bricks are laid to form a circular patio or walkway. When this mode is used, Width sets the radial size of the blocks, and Height sets the angular size.

You can choose the center of the Radial Mosaic effect. The default uses the center of the image. The **Radial Center** item picker selects an item (often a null object) which defines the center as seen from the camera. You can animate the item to move the center.

The **Random Mosaic** and **Very Random Mosaic** options both make random polygonal blocks of pixels shaped like flagstones.

The **Animation Rate**{Rate means "speed" here. It would be great to have a plugin that let you choose the amount of money you got paid for animating.} control is used to make the "Random" mode block pattern *change* over time. This is a strange effect, since it makes the image shimmer in a unique way. This shimmer is useful, for example, when your animation is supposed to be a transmission from a mining asteroid, and its video transmitter is low quality. Instead of showing 1960-style white-snow interference, you can show year 2060-style animated-flagstones video interference.

The Animation Rate spins the Radial Mosaic effect around its center.

Mosaic doesn't have to affect the whole image. You can restrict it to a circular spot by specifying an item to use as the center with the **Effect Center** control. Like Radial Center, the position of the item in the camera view determines the center of the spot. This is useful when you have a moving object that you want to keep obscured, since if you pick that object (or a null that you parent to it,) the obscuring mosaic will follow. **Radius** sets the size of the spot. **Invert** makes the image obscured everywhere except around the center.

The effect can also be restricted to appear only on certain surfaces. "Special Buffers" are a feature of LightWave which allow you to flag surfaces for plugins like Mosaic. To do this, first select the **Use Special Buffer** button in Mosaic. Next, in LightWave's Surface panel, visit the surfaces you want to restrict Mosaic to. Use the Special Buffer button to enter a value between 0 and 1.0. A smaller number will apply a weaker Mosaic effect, with 0 giving no effect. (In LW 5.6, the values range between 0 and 255.)

9.2 Mosaic Application Ideas

- Show a display screen or video monitor which has a poor connection. The Random mode can be animated to make the whole image "crawl." The simple Grid mode works well too, especially if you animate the Width and Height controls to make it appear that the connection degrades. You can pulse the size settings to make it feel like static, or slowly increase them to make the transmission "fade away" as you watch.

- Simulate a military briefing. Maybe they don't want to clearly show details of their latest antimatter spacecraft, but they do want to show the effect of an attack it was used in. The United States did this with some videos during the 1991 Gulf War.{Though the videos showed airplanes and tanks, not their antimatter spacecraft.} You could also obscure "secret data" like map captions or HUD overlays.

- The movie *The Terminator* had a great scene which showed you the view from the Terminator's viewpoint, identifying the items he was seeing. The Radial Mosaic option is particularly useful for this kind of "computer analysis view" since the circular shape attracts your eye to the center.

- When your human model is so detailed that you need to obscure the interesting parts, parent a null over the area that you don't want people to view clearly.{Of course, the obscuring effect attracts people's attention, so they end up focusing on that region much more than usual.} Use this as the Effect Center and the blockiness will follow even if the camera or object moves.

9.3 Mosaic Example Scenes

Mosaic_Naughty.lws

The area of effect of Mosaic can cover a moving object. This scene shows a Mosaic grid obscuring a statue.{Don't render this scene without Mosaic or you'll be morally corrupted forever!}

Mosaic_Focused.lws

The center of the Radial effect can be animated. This scene shows the center following a moving car{It's only a model. Shhh!} to attract your attention to it.

9.4 Mosaic FAQ

Question Can I make blocks so thin they're just one pixel wide?

Answer: A Width setting of 0% will make one pixel blocks. This is an interesting effect since you'll have a "streaky" effect instead of a blocky one. You can set Height to 0% too.

Question How do I make an animated Random Mosaic freeze momentarily, as if there were transmission problems and dropouts like on real TV?

Answer: The Animation Rate control can be animated itself. When the rate is 0%, the effect freezes. The Width and Height controls can also be animated. Setting both to 0% makes one pixel blocks, so the output image has full quality with no degradation. To degrade the image over time, animate the sizes getting larger.

Question How can I zoom the preview area to see more detail?}

Answer There aren't any pan or zoom features in the image preview. You can make a zoomed version by renderering once using LightWave's Limited Region. Mosaic will enlarge the area to fit the preview box.

9.5 History

The very first graphics program I ever wrote, in 1990, was very similar to Mosaic. I had a black and white image of my friend Evan captured with a NewTek Digiview video digitizer. I had access to a laser printer and I decided to manipulate his picture in some funny way and print it. I wasn't very skilled at using a paint program, so I decided to write a program to distort the image.

It wasn't easy, mostly because of the hassles reading image files, but I finally made a program that shifted the image brightness to make Evan look spooky. I put the image on the wall outside my dorm room.

I also experimented with other image effects. One made the image wiggle back and forth so it looked like a reflection on a distorted mirror. One was "radial quantization" which is really the same algorithm as the Radial Mosaic option in this plugin.

I made about 20 different image distortions just for fun, and I posted one in the hallway each day. I titled the collection *The Image Transform of the Day*. Visitors enjoyed the collection, and often wrote funny comments on different images, mostly sympathizing with the terrible tortures applied to Evan.

The Mosaic plugin for LightWave was first written in 1997 for a small studio for a TV commercial. They wanted to obscure the label of a (rendered) product. They first used the 2D effects program After Effects to do the obscuring, but exporting the proper alpha channels was annoying and had to be repeated every time the animation was modified. An hour of work produced a simple tool to do this in LightWave and solved their problem.

I added the radial effect for a different studio which needed to animate a "computer analysis" of an image. The same studio later used Mosaic for several other projects, and I added the image preview and object tracking abilities. In 1999 I decided to polish Mosaic to include with Taft. I added the two Random Mosaic methods and enhanced the image preview significantly.

Chapter 10 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 website 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.

10.1 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.

10.2 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 website. 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

Website: http://www.worley.com

Email: support@worley.com