THE NEWSLETTER FOR LIGHTWAVE 3D ANIMATORS

LIGHTWAVE PRO

an Avid Media Group, Inc. newsletter

Volume 3 Issue Number 5 • May 1995 • $8.00

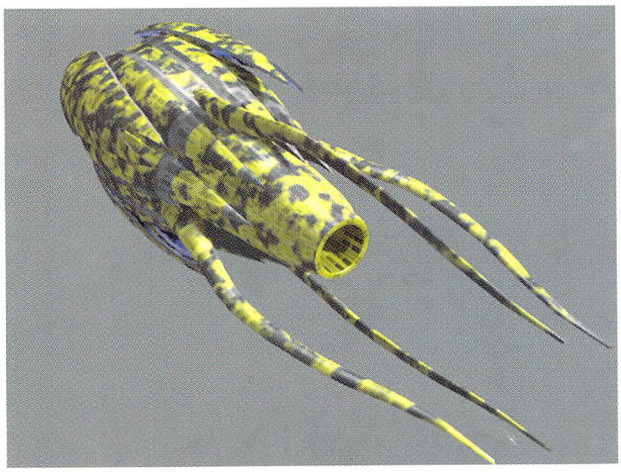
Inside:

LightWave 4.0 Benchmarks

Top 10 Fractal Noise Tips

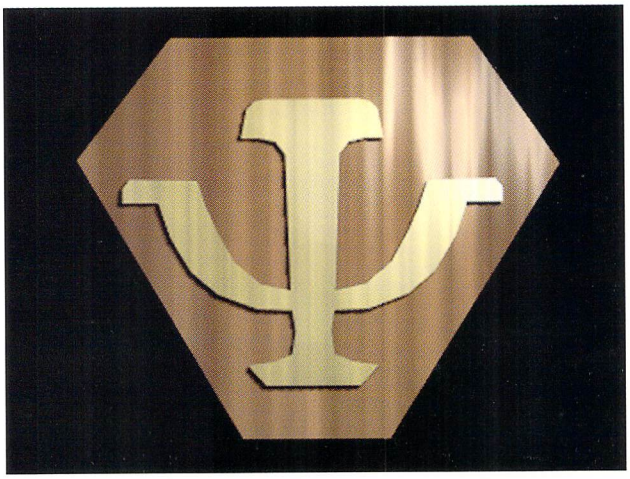
LightWave 4.0 Ships

ANIMATION STILLS



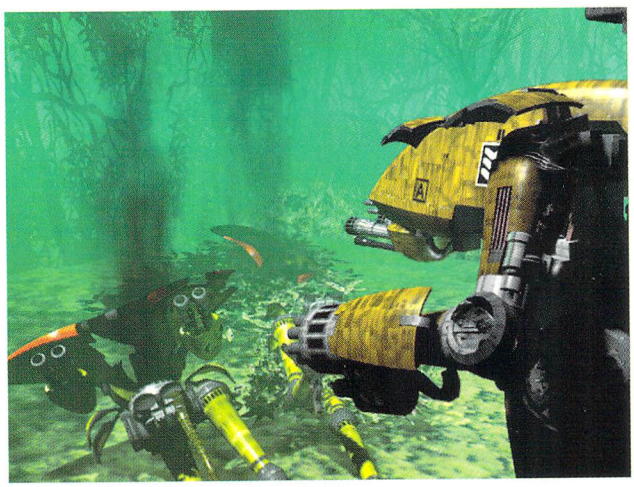
Vorlon Fighter

The Vorlon fighter from *Babylon 5* uses fractal noise as its main surface feature—almost like a cow. See "Top 10 Uses for Fractal Noise," page 8.



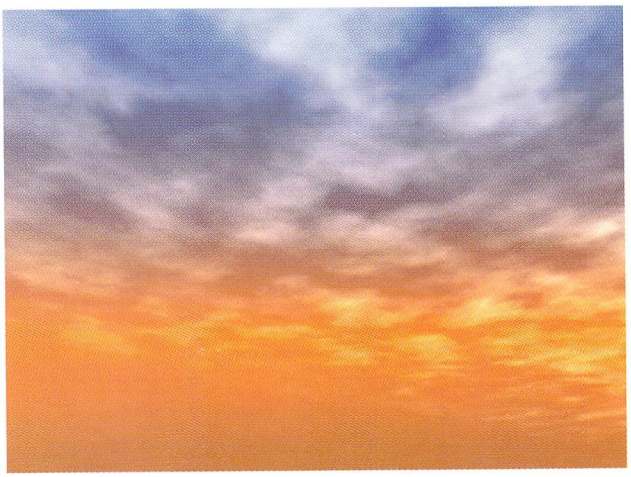
Fractal Logo

This logo uses a vertical fractal noise texture in the diffusion, specular and bump map channels to simulate a brushed metal look. See "Top 10 Uses for Fractal Noise," page 8.



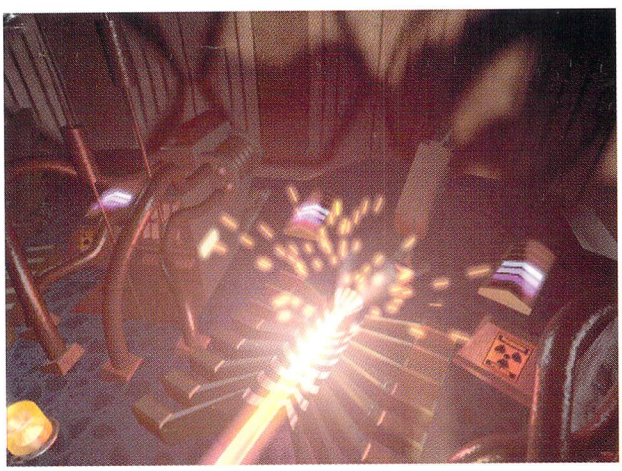
Fractal Smoke

The smoke seen rising from the wreckage of an unlucky robot is...Yes! Fractal noise! It's also used for much of the dirt seen on the foreground object. See "Top 10 Uses for Fractal Noise," page 8.



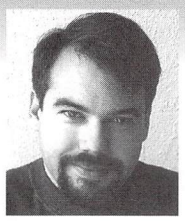
Bohemian Sunset—Not!

What you're actually seeing is fractal noise! See "Top 10 Uses for Fractal Noise," page 8.



Glowsurface

All glows in this image but the energy beam are created with the new Glow attribute in the Surfaces panel. See "Aspect Ratios," page 14.



EDITOR'S MESSAGE

by John Gross

This issue of *LIGHTWAVEPRO* takes a firsthand look at LightWave 4.0. Even though LightWave (at press time) has only been prereleased for Intel, Alpha and MIPS machines running Windows or Windows NT, it is close enough for a "pre-review" and some benchmark tests.

What's currently missing from the prerelease is any built-in plug-ins, new Bones features, a new motion blur option and some little things here and there.

One of those little things not found in the prerelease is a Content Directory feature in the Options panel that cleverly allows LightWave to make use of relative paths. What this means is that you can set a content directory path in which LightWave defaults to look for images, objects and scenes directories (among others). The beauty of this feature is that you can set up a scene using a selected content directory, and LightWave will save the scene file and object files using the relative paths. For instance, a scene will look for an object in an Objects path without using the content directory path itself in the scene file.

This is similar to an Amiga scene setup in which the objects are in the Toaster/3D/Objects directory. If you have ever read a scene file created this way, you would see that the scene reads "LoadObject Objects/object name," rather than "Load Object DH0:Toaster/3D/Objects/objectname."

By changing your content directory, you will instruct LightWave to save out relative paths so you may have a directory on your hard drive containing your Objects, Images and Scenes directories. From there you could easily move them to a floppy disk with the same path structure. Simply changing the content directory to the floppy enables the scene to load without a hitch. Also, you can set up a scene using a content directory on a PC, and then transfer it over to an SIG. Changing the content directory to the new SIG directory path enables seamless scene loading.

The upshot of all of this is that if you have several scenes saved on your current platform and are switching to another, you may want to make sure that your scenes are set up using the relative path structure. That way you can re-create this path structure on your new system and allow your scenes to load without a

see Editor's Message, page 9

TABLE OF CONTENTS

LIGHTWAVEPRO
an Avid Media Group, Inc. newsletter

"We provide the most valuable information to people who use technology to create messages with impact"

Editor ..... John Gross
Managing Editor ..... Jim Plant
Editorial Coordinator ..... Joan Burke
Associate Editor ..... Corey Cohen
Art Director ..... Helga Nahapetian Taylor
Art/Production Coordinator ..... Kristin Fladager
Production ..... Sergio "Berimbau" Miller
Circulation Director ..... Sherry Thomas-Zon
Circulation Assistant ..... Debra Goldsworthy
Subscriber Services ..... Kris Nixon
Contributing Writers ..... Dan Ablan
..... Arnie Boedecker
..... Philip Hice
..... Mojo
..... John F.K. Parenteau
..... Mark Thompson
Group Publisher ..... Michael D. Kornet
Editorial Offices: Avid Media Group, Inc.
273 N. Mathilda Avenue, Sunnyvale, CA 94086
Telephone (408) 774-6770; Fax (408) 774-6783
John Gross can be reached electronically at:
jgross@netcom.com (Internet); 71740,2357 (CompuServe)

Printed on recycled paper, 10% post-consumer waste.

Are you interested in writing for *LIGHTWAVEPRO* or submitting images? If so, contact us at our offices or electronically.

LIGHTWAVEPRO (Vol. 3, No. 5); (ISSN 1076-7819) is published monthly by Avid Media Group, Inc., 273 N. Mathilda Ave., Sunnyvale, CA 94086-4830. A one-year subscription (12 issues) in the U.S. and its possessions is $48 (U.S.); Canada/Mexico, $60 (U.S.); overseas, $84 (U.S.). To subscribe, call toll-free 1-800-322-2843. Allow 4 to 6 weeks for first issue to arrive. Second-class postage rate paid at Sunnyvale, CA and additional mailing offices. POSTMASTER: Send address changes to *LIGHTWAVEPRO*, 273 N. Mathilda Ave., Sunnyvale, CA 94086-4830.

About the cover: No, it's not the software package formerly known as LightWave 3D. But it is the new LightWave 3D logo, found on brochures, magazines and cool baseball caps.

LightWave 4.0 Benchmarks

by John Gross and Philip Hice

With the prerelease of LightWave for Windows and Windows NT machines, we thought it was time to provide some benchmark render tests. These evaluations are just designed to assess LightWave's rendering times, and we are not looking at things such as redraw time, Modeler functions and preview generation. Expect those tests in future issues of *LIGHTWAVEPRO*. We tested as many machines as we could get our hands on. Since we will keep our results as default benchmarks (and NewTek will include them with the final shipping versions), we will be able to examine new equipment as it becomes available to us.

We used four separate render tests for our benchmarks. Also, since Joe Dox's Full Metal Hummer scene (included on February's *LWPRO* disk) has been used as a benchmark scene by some people on the Internet, we have included render times for that scene as well (with some slight modifications) for some of the systems tested.

Since LightWave was designed primarily for video work, all scenes are rendered at 752x480 using one segment (segment memory set to 8,670,000) except where noted. These scenes were designed to test LightWave, but not be stupid about it. For instance, objects are modeled correctly and care has been taken to set up each scene so it will render as fast as it can (in most instances). All scenes were set to not render to any display devices, and when run on NT machines, the **Show Rendering in Progress** option was disabled. Any scenes using Antialiasing (all but ZBufSort.lws) use a default **Sampling Threshold** of 8. This may be overkill on certain scenes, and raising it can definitely speed up render times, but we left it for the best possible image quality.

The version of LightWave used for testing was one that is midway between the prerelease version (for Intel) and the final shipping version. Rendering times will not change in the latter version. Because of time constraints, not all scenes could be rendered on some of the slower machines. These will be noted where applicable.

At the end of this article, we have included some "times faster" comparisons using an Amiga 2000 with a Fusion Forty '040 card as our base unit of "1." This machine was selected because it is faster than a stock Amiga 4000 and many professional LightWave animators have a system that is faster than a stock 4000.

Also included at the end are render times for some of the slower machines that hadn't finished all of the tests at press time. These times are included for informational purposes and are not grouped in the average time calculations.

The Scenes

DOF.lws

This scene is composed of two simple text objects: one in focus, one not. Also, there are two shadow map light sources, each with a shadow map size of 600K. Antialiasing was set to High to give the best depth-of-field effect. While the objects are not huge polygon-wise, they could be considered standard logo-type objects.

RayTrace.lws

This scene is designed as a FPU killer. It has three balls—one crystal, one reflective and one leaded glass—surrounded by a 14-sided cylinder with 100% reflective sides (to get recursive reflections). Finally, underneath the balls is a flat plane using a ripple bump map. There are three ray-traced light sources, and the crystal and leaded glass balls have internal and external surfaces with separate refraction indices. To slow down the ray tracing, the balls are overbuilt with 5,000 polygons (the crystal and lead glass balls each have 10,000 polygons). This scene has ray-traces, shadows, reflection and refraction and can be a real time-killer. Antialiasing is set to Low.

Textures.lws

This new texture examples scene contains 20 cubes, each with a different type of procedural or bump map. The scene, while not featuring a lot of "normal" polygons, was included to test some procedural times and provide a requisite new textures scene. Antialiasing is set to Low.

ZBufSort.lws

This scene is designed to test the time needed for Z-buffer sorting. It consists of an object composed of 5,000 flat planes stacked 1 mm apart along the Z axis. A setting of 1 mm was chosen to get the planes close enough so LightWave would have to do some thinking to decide what's behind what.

Hummer.lws

This scene was included on a previous *LWPRO* disk. We included it here because it represents a fairly common type of LightWave scene and some people have used it as a benchmark on the Internet. The only changes made to this scene for our benchmarks was that frame 163 was chosen for the test because it has some dissolved objects that are in view and some other objects that better display motion blur. Antialiasing is set to Medium, Motion Blur is selected, Ray Trace Shadows is on and the frame segment memory was bumped up to 8,670,000 in order to render the frame in one segment.

The Machines

Amiga 3000

Stock 68030
18MB RAM
AmigaDOS 3.1

Render Time

DOF.lws	03h 25m 28s (12,328s)
RayTrace.lws	39h 53m 22s (143,602s)
Textures.lws	00h 59m 23s (3,563s)
ZBufSort.lws	03h 43m 35s (1,3415s)
Hummer.lws	14h 46m 22s (53,182s)

Amiga 2000

Fusion Forty 040 33MHz
32MB RAM
AmigaDOS 2.1

Render Time

DOF.lws	01h 00m 24s (3,624s)
RayTrace.lws	07h 42m 09s (27,729s)
Textures.lws	00h 16m 23s (983s) (737)
ZBufSort.lws	00h 38m 51s (2,331s)
Hummer.lws	02h 48m 56s (10,136s) 7702

486DX-50 PC

486DX-50MHz CPU
32MB RAM
256K cache
Windows NT 3.5

Render Time

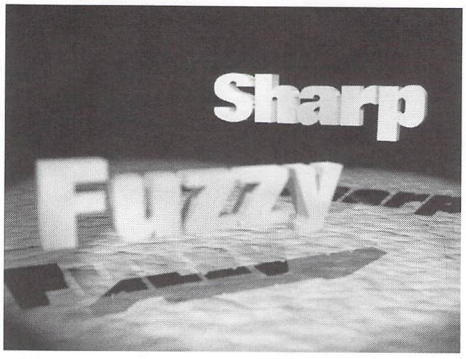


Figure 1: Depth of Field (DOF.tga) This image shows the result of the DOF benchmark test. 752x480, High Antialiasing with Depth of Field selected.

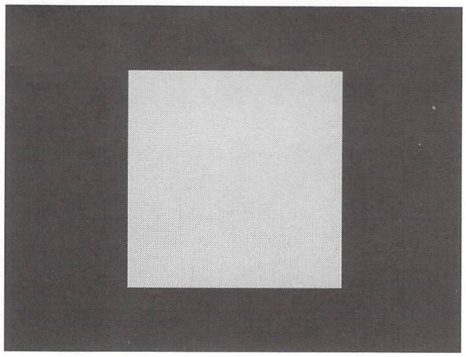


Figure 4: Z Buffer Sort (ZBufSort.tga) Five thousand polygons, each 1mm apart, make up this scene. 752x480, no Antialiasing.

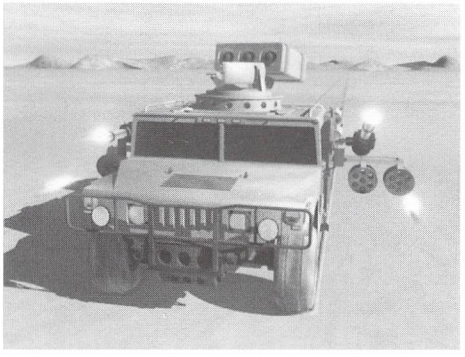


Figure 5: Full Metal Hummer (Hummer.tga) This test was conducted to benchmark a "typical" LightWave scene. Shown above is frame 163 of this scene by Joe Dox,discussed in February's *LWPRO*. It was rendered at 752x480 with Medium Antialiasing.

DOF.lws	00h 50m 49s (3,049s)
RayTrace.lws	*
Textures.lws	00h 14m 09s (849s)
ZBufSort.lws	00h 28m 33s (1,693s)
Hummer.lws	03h 48m 33s (13,713s)

** RayTrace scene did not finish in time to print results!*

Pentium PC

Pentium 100MHz

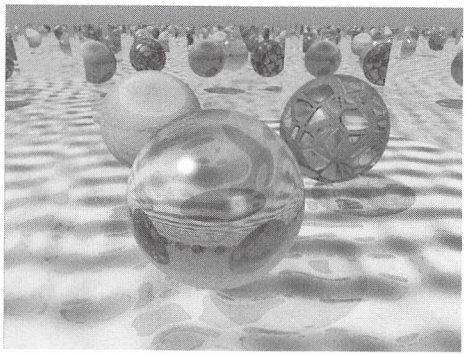


Figure 2: RayTrace (Raytrace.tga) This image is the result of the Ray Trace benchmark test. 752x480, Ray traced refraction/reflection and shadows, Low Antialiasing

32MB RAM
256K cache
Windows 3.1 for Workgroups and
Win32s/ Windows NT 3.5

Render Times—Win32s

DOF.lws	00h 10m 47s (647s)
RayTrace.lws	02h 03m 53s (7,433s)
Textures.lws	00h 03m 57s (237s)
ZBufSort.lws	00h 09m 06s (546s)
Hummer.lws	00h 51m 43s (3,103s)*

** The Hummer scene would not render in one segment under Win32s. It was set to a Segment Memory of 4,000,000 to render in two segments.*

Render Times—Windows NT 3.5

DOF.lws	00h 09m 17s (557s)
RayTrace.lws	01h 14m 18s (4,458s)
Textures.lws	00h 02m 43s (163s)
ZBufSort.lws	00h 06m 32s (392s)
Hummer.lws	00h 28m 16s (1,696s)

ShaBLAMM!

R4400 MIPS CPU clocked at 100MHz
32MB RAM
Windows NT 3.5

Render Times

DOF.lws	00h 08m 11s (491s)
RayTrace.lws	01h 06m 38s (3,988s)
Textures.lws	00h 02m 19s (139s)
ZBufSort.lws	00h 07m 23s (443s)

NekoTech Mach 2-289

21604 Alpha CPU clocked at 300MHz
64MB RAM
2MB cache
Windows NT 3.5

Render Times

DOF.lws	00h 03m 07s (187s)
RayTrace.lws	00h 23m 11s (1,391s)
Textures.lws	00h 00m 59s (59s)
ZBufSort.lws	00h 02m 22s (142s)
Hummer.lws	00h 08m 58s (538s)

Carrera EV5

21164 Alpha CPU
266 MHz CPU clock
64MB RAM
2MB cache

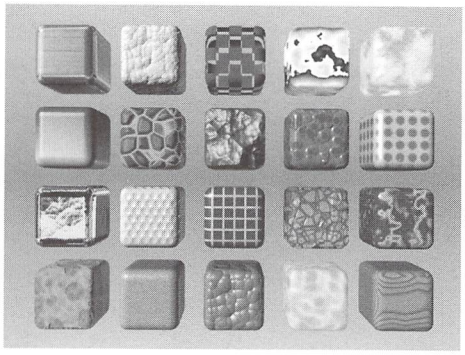


Figure 3: Textures (Textures.tga) This new texture examples scene was rendered at 752x480 with Low Antialiasing.

Windows NT 3.5

Render Times

DOF.lws	00h 01m 43s (103s)
RayTrace.lws	00h 16m 25s (985s)
Textures.lws	00h 00m 33s (33s)
ZBufSort.lws	00h 02m 07s (127s)
Hummer.lws	not tested

Carrera EV5M

21164 Alpha CPU
250* MHz CPU clock
64MB RAM
2MB cache
Windows NT 3.5

Render Times

DOF.lws	00h 01m 58s (118s)
RayTrace.lws	00h 18m 37s (1,117s)
Textures.lws	00h 00m 36s (36s)
ZBufSort.lws	00h 02m 17s (137s)
Hummer.lws	not tested

Carrera Cobra

21064 Alpha CPU
200 and 275 MHz CPU clock
32MB RAM(200MHz) or 64MB RAM (275MHz)
512K cache (200MHz) or 2MB cache (275MHz)
Windows NT 3.5

Render Times

DOF.lws	00h 04m 05s (245s)	200MHz
	00h 03m 19s (199s)	275MHz
RayTrace.lws	00h 33m 08s (1,988s)	200MHz
	00h 24m 30s (1,470s)	275MHz
Textures	00h 01m 17s (77s)	200MHz
	00h 01m 04s (64s)	275MHz
ZBufSort.lws	00h 03m 30s (210s)	200MHz
	00h 02m 33s (153s)	275MHz

The Results

The following table is included as a graphical representation of each system's performance divided by scene file rendered. The numeric value given is the performance rating of each system on the given scene in relation to the base system. In other words, a system 10 times as fast (1000%) as our base system would show a rating of 10 on the graph. (As stated earlier, the base system is an Amiga 2000 with an RCS Fusion Forty 040 accelerator running at 33 MHz.)

see LightWave Benchmarks, page 11

Digital Cinematography

by John F. K. Parenteau

So I'm sitting in a Henry bay writing this column. Just in case you aren't familiar with it (I wasn't sure myself what it was until recently), a Henry is an advanced paint box system, sort of a superhuman ToasterPaint. It's faster than a Harry (another paint box system) but much more expensive. Figure a rate card of $1,000 per hour, with a minimum of an hour for any work. We try to avoid using the Henry, but with some television effects schedules, a quick 2D system is invaluable. It's really an extremely mature DPaint in real-time. The best thing about it is the versatility of pulling mattes, rotoscoping, zooming in and out of a frame, or even bending and twisting the image. For you film buffs, the equivalent at motion-picture resolution is a Flame. Though they have quite a few different features, they are essentially the same type of box.

Well, back to the Henry bay. I'm hanging out here chatting with the Henry artist and asking for some fairly routine things: "Let's blow up this frame a bit, add some grain, etc." Simple requests, really. We're combining digital images with live action, pulling a matte for the character and rotoscoping some highlights.

It never really hits me until later how cavalier we've become in our modern image manipulation. I think back to the days of *Star Wars* and how complex it was to composite all the model passes for the battle sequences. Just look at that film on videotape now and you will be shocked at the horrible matte lines. [Editor's note: See July 1995 *VTU*'s "Last Word."] After looking at that and other examples, I've come to realize how important it is to be extremely aware of the spectrum of resolutions involved in digital cinematography. As a cameraman several years ago, I used to choose between 35mm, 16mm and even Super 8mm film without any thought other than the quality of the negative I was exposing. My greatest concerns were the speed of the stock, whether I was shooting daylight or tungsten, and what kind of grain structure I was looking for. As I move into the digital realm, I've come to realize how all these considerations are actually hacks of problems inherent in the system of exposing film. These same "problems" do not exist in a computer, though they must be carefully considered if we hope to match that "Film Look." In the world of CGI, and in particular our work with LightWave, we have, as a tool, an independent resolution renderer. Yet without the knowledge of what the various resolutions are, this function is of little use.

The basic settings (assuming the **Basic Resolution** is set at **Medium**), without activating the **Custom Resolution** button, are dependent on the frame aspect chosen. **D2 NTSC** (composite video) is 752x480, essentially the frame size you see on your home television. **D1 NTSC** (component video, also known as Abekas Aspect) is 720x486. D1 resolution is considered the highest resolution available on videotape. As a digital format, it doesn't suffer from signal loss in duplication. If you view D1-resolution images on a monitor (through TPaint, for example), they will be slightly mis-sized—squished, actually—on the horizontal. **Square Pixels**, a function used primarily for print work, is rarely utilized in video resolution. As the name suggests, this setting creates square pixels. (We'll get back to it later, as it applies to the film work we will discuss.) PAL settings (D1 and D2) are European standards and shouldn't be used for NTSC standard video. It is important to note that European video, known as PAL, is actually a higher resolution than our NTSC video.

Before we move on into the world of higher resolutions, let's discuss the **Camera** panel in LightWave. (I'll assume we all have the fortune of working with version 4.0. Those who don't, go to the back of the class.) In the black area below **Basic Resolution** and **Pixel Aspect** settings is some of the most important information for understanding resolutions.

Full Resolution describes the number of pixels (width x height) you are asking the software to render. The denser the pixels, the more detail in the image, and the larger it can be blown up before degrading begins. For example, take a video-resolution image into LightWave. Create a simple plane and map the image onto the plane. Place the plane so it just fills the frame and render it out. Notice that the image in Figure 1 looks just fine.

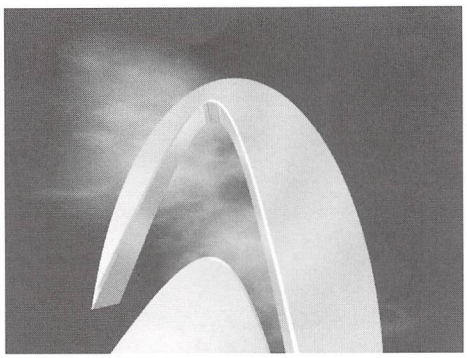


Figure 1: A full frame of video looks fine in full-frame format.

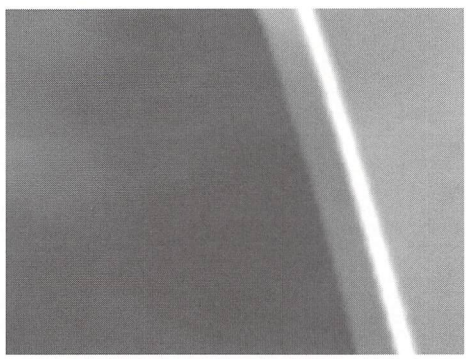


Figure 2: Pushing in on the video causes blurring and "pixelating."

Now move the camera toward the plane until only a small portion of the plane, and thus the image, is visible. Figure 2 shows the breakup of the image, called pixelating, as the information in the frame breaks down. Video resolution is barely high enough to withstand a full frame view. With some systems, like the Henry, a slight blow-up is possible. As you'll see later, the highest resolutions are required when rendering for projected film, for example, since the image will be viewed on a huge screen in a theater. It is actually a luxury to work at film resolution (anywhere from 1828x1332 or so and up). In Figure 3 I've shown a blow-up of a full-resolution film image. As you can see, the density of the pixels makes work such as rotoscop-

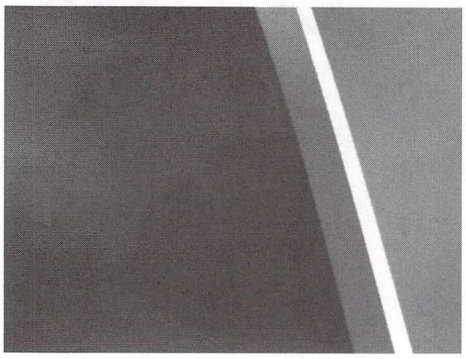


Figure 3: A film-resolution frame holds up much better when blown up.

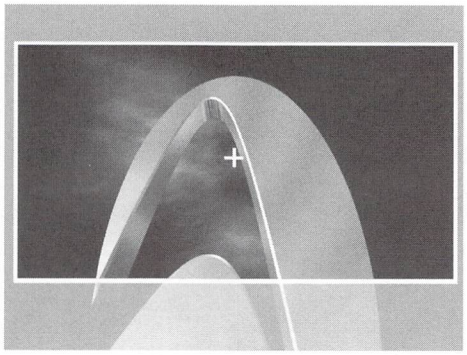


Figure 4: A sample framing with a 1.85 aspect matte.

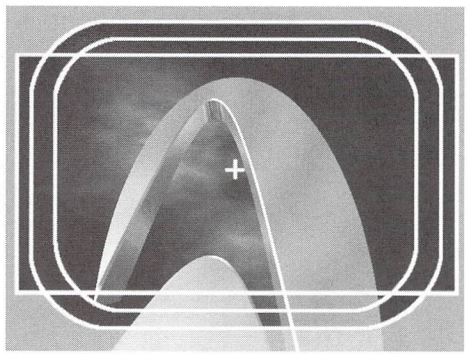


Figure 5: Using a combo matte (1.85 and 1.33 for television) allows for greater headroom.

ing in film resolution a breeze!

Just below **Full Resolution** is **Limited Region**, which we will pass over. This function is an effective tool for rendering only small portions of the frame while testing, but is not actually for final output (unless you wish to use it to crop a portion of a large image). Below Limited Region is **Pixel Aspect**. This can be the most confusing part of higher resolutions, so bear with me.

The pixel aspect is the ratio of the pixel width divided by its height, and describes the actual shape of the pixel. OK, keep that in the back of your mind and let me explain **Frame Aspect.** As you look at a television monitor, you might notice that it isn't actually square. A TV screen, and thus the image it displays, is really slightly wider than it is tall. If we go back to D2 resolution, you might remember it is 752x480, width by height. A pixel generated for television is not actually square, but slightly taller than wide. Even if a television screen was square, there would still be more pixels in the width than the height, to make up for the thinner video pixel. The Frame Aspect is the ratio of the height of the frame to the width of the frame. Allen Hastings, the creator of LightWave, gives LightWave animators the pixel aspects for D1 and D2 for both NTSC and PAL, so they don't have to figure them out. Just leave the basic resolution in Medium and toggle between D1 and D2. The Pixel Aspect changes slightly as the type of format is changed. The Frame Aspect is simply a function of the resolution width divided by the resolution height times the Pixel Aspect. Or, more accurately, the resolution width times the Pixel Aspect divided by the resolution height. It's all very confusing and not an issue for this article. [Editor's note: For more information, see Mark Thompson's "Aspect Ratios" article in this issue.]

What you really need to know is this: Pixels come in different shapes depending on the display device. Any method of viewing an image is considered a display device, from your home television to a projector in a movie theater. Video-resolution pixels, as mentioned before, are taller than they are wide. Film-resolution pixels are square for the most part, depending on some factors I will delve into later. As I mentioned earlier, a television screen is actually wider than it is tall. If you can picture most film resolutions as projected in a theater, these images are also wider than they are tall. This is the Frame Aspect. As we move into the realm of film resolutions, we must discuss the various Frame Aspects, or **Aspect Ratios**, available.

If you have purchased any laser discs in recent years, you might be aware of the availability of a "letterbox" version of many films. This is an effort on the part of laser disc manufacturers to retain the original style of the film, as seen in a theater. Motion picture film negative is exposed a single frame at a time, much like in a still camera, yet at a rate of 24 frames per second. If you were to examine a single frame of the print, you would notice that the frame is actually fairly square, yet when projected in a theater, the image is wider than it is tall. This is due to the Aspect Ratio the film was shot in. When camerapeople shoot a film, they use an etched piece of glass in the viewfinder to describe an area on the film negative (Figure 4). Even though this is the area chosen to be projected, the entire film is exposed and has an image on it. This choice is not arbitrary but rather a set of standards established years ago by cinematographers around the world.

Standard motion picture aspect ratio is considered 1:85, meaning the image is 1 inch tall and 1.85 inches wide. The etched lines in the glass the operator is looking through describe this area in a standardized position on the film. Once the film is exposed, it is processed as an entire frame, including the areas above and below the frame etchings. When the processed film is projected, the projectionist slides a metal mask between the film and the lens. This mask closely approximates the lines in the glass the operator uses to frame the image.

It's important to understand that when the film is exposed, the entire frame is exposed and the operator can see the entire frame size. The marks on the glass are just that: scratches on the glass. If, in the theater, you pulled the mask out from the projector, you would reveal the area not meant for viewing, often containing the boom microphone, crew people above the set, etc. Many films push the limit of the frame and accidentally reveal information they shouldn't. This is a rarity, since other factors are considered when framing the image, such as the eventuality that the final product will end up on television.

Commonly, 35mm theatrical film uses an aspect ratio of 1.85, as described above. Television shot on film is shot in a 1.33 aspect ratio, the frame being 1.33 percent wider than it is tall. Most motion pictures are destined for the video realm at one time or another (some quicker than others!), so camera operators use a Combo viewfinder, showing both the 1.85 and 1.33 aspect ratios (Figure 5). Anamorphic or Cinemascope film is shot using a special lens that squeezes the information onto the film. When examined manually—holding it up to a light, for example—the image will appear squeezed, with people and objects seeming tall and skinny (Figure 6). This **Aspect Ratio** of 2.35 (a whooping 2.35 percent wider than its height), provides a super-wide view for projection. A 70mm image, in most cases, starts on 35mm film, is shot in anamorphic 2.35, and optically blown up to 65mm film.

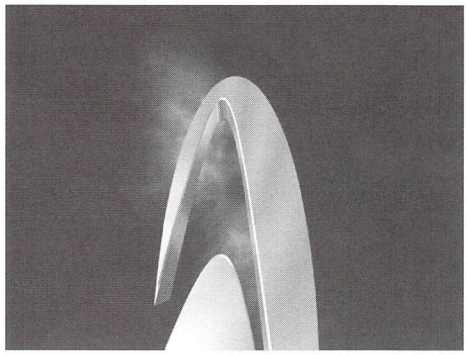


Figure 6: This anamorphic image looks "squeezed" before projection.

It's important to understand these facts as you translate into the digital arena. There are a couple issues inherent in the exposing of film stock that must be taken into consideration when composing your CGI image.

1) Remember that, as in live-action photography, you must provide information for the entire frame of film. As you compose an image—using a 1.85 aspect, for example—it is not acceptable to remove information from the areas above and below the 1.85 frame (e.g., using a mask.) If you remember the old Letterbox function in an earlier version of LightWave, a framestore was rendered out in an approximately 1.85 aspect, with black bars at the top and bottom. Though this looked

see Digital Cinematography, page 18

Top 10 Uses for Fractal Noise

by Mojo

When it comes to object surfacing in LightWave, fractal noise is your best friend. It is without a doubt the Pocket Fisherman of procedural textures. With a solid understanding of how the texture works and a little creativity, it can be used to make anything from cows to clouds. The following list includes just a few of its many uses; experimentation will yield many more. Readers are encouraged to send in their own Fractal Noise tricks for future publication and claim the glory they rightly deserve.

What It Is

Fractal noise is a procedural texture, which means the computer generates it mathematically. There is no actual artwork involved. Therefore, you can get as close as you want to any surface with this texture and never see any aliasing. Using the velocity channel, you can also animate the texture along any or all axes, giving it movement and life at whatever speed you desire.

The splotches this texture creates usually look best when sized to approximately one-tenth of the object size. (A default of one meter works well on a 10-meter surface.) If the texture is sized too large, you may see nothing. If too small, you'll get an overly fine pattern that shimmers like crazy.

Contrast determines how soft the splotch edges will be. A default of three creates a medium softness, while higher numbers bring the edge to a sharp line. The frequency values make either a more or less complex pattern. Low numbers create fewer and less-defined splotches (which may pulsate when animated). Higher settings provide sharper, more-defined patterns (and take slightly more time to render). The limit is 16.

OK, let's get cracking. Drum roll, please!

Top 10 Uses for Fractal Noise

10. Cows: The spots on the cow object provided with LightWave were produced using a high-contrast fractal noise pattern. Don't see much use for cow creation? The exact same technique (though with different colors) was used to form the pattern on the Vorlon ships in *Babylon 5*. A velocity along the Z axis provides the motion sometimes seen on the inside of the ship's petals. With a little bit of tweaking and perhaps a larger texture size, this same process makes a nice-looking camouflage texture, good for a tank, plane or anything else designed to kill and maim.

9. Dirt: Many people already use fractal noise to make a surface look dirty or smudged. This looks best when used with a low contrast and a color similar, yet darker, than the object's base color. It also helps to stretch out the noise in the direction it might naturally occur on a surface. The road object, for instance, might look best with dirt stretched along the Z axis.

A better way to create fractal dirt is to use it as a diffusion and/or specular texture. If an object already uses a color image map, this is a necessity; it also takes the guesswork out of deciding on a darker color for the noise. Simply make the diffusion value 100% for the surface and lower it for the texture value inside the fractal noise control panel (the lower the value, the darker the noise will appear). This will reduce the **Diffuse Level**, allowing less light to hit the object in the pattern of fractal noise and creating a more realistic-looking dirt.

By applying the same values in the **Specularity** channel, you can make the surface appear less shiny in the same dirty areas. This will greatly add to realism, especially when creating metal surfaces.

8. Brushed Aluminum: Speaking of metal, designing a good brushed-metal texture (made famous by the *Terminator 2* logo) is a snap with your old buddy fractal noise.

As explained earlier, apply the noise as a diffuse and specular map. The trick is to stretch the noise out along the X axis (assuming your logo is facing you) so far that the noise becomes a series of straight lines. Start with a size approximately 10 times larger on the X. If it is stretched too far, your lines may get too thin and alias like crazy. Also, take care to make your differences between surface and texture value small, again to avoid aliasing and to capture the subtle contrast in brushed metal.

Using the trick as a subtle bump map can make the surface look even better, though it greatly increases render time. Experiment!

7. Windows: Creating windows on buildings or spaceships is a snap. However, having a bunch of uniform, solid blocks of light never looks quite right. Looking at photos of real buildings at night reveals the caveat: there's stuff inside those windows. Whether it's people, drapes, furniture or plants, little things here and there break up the light coming from the window. By adding a fractal noise pattern in the luminosity channel of your windows, you can produce a good facsimile of this look. Follow the same rules for generating a channel of fractal diffusion and play with the size, contrast and texture value until it looks right.

6. Shadowing: Fractal noise can be applied to surfaces in less direct ways. In Hollywood, when a set is looking a little too flat, the light hitting it will often be broken up with something called a cookie, a piece of cardboard with random patterns cut into it. The cookie is placed a foot or two in front of the light, and presto! Lots of subtle shadows now cover the set. This same trick can be accomplished easily with LightWave and—you guessed it—fractal noise.

Make a simple, flat polygon and apply a clip map of noise to it. (Render it in front of the camera to make sure the size of the pattern is what you want.) Using a clip map literally cuts holes in your polygon. For this example, the holes are in the shape of fractal noise (contrast has no effect here). Place this polygon in front of a shadow-mapped light source and lots of subtle shadows will cover your LightWave set.

Moving this polygon toward and away from your light makes the shadows bigger or smaller, just like in real life. Adding a velocity to the noise shifts your shadows, perhaps creating the effect of leaves blowing in the wind outside. Make sure to turn on Double Sided for this polygon. If it faces the wrong way, you won't see anything!

5. Smoke: Begin by constructing a well-subdivided tube. This will be the smoke object, so place it in Layout wherever you want the effect to protrude from. Add a **Fractal Bump** displacement map to it, with a slow, upward velocity along the Y axis. This

makes the tube bob and weave like billowing smoke. Use **Fractal Noise** in the **Transparency** texture channel, with a similar velocity along the Y axis. A low contrast is probably best, as is overall slow texture velocities, for duplicating the smooth, flowing feel of smoke.

Make sure **Edge Transparency** is set to **Transparent** to fade the edges of your tube and click off **Double Sided**. (There's no reason to see the inside polygons of your tube.) Unfortunately, the one channel of fractal transparency is simply too dense and regular to effectively mimic smoke. Wouldn't a second layer of transparency be useful?

By clicking on the **Additive** button near the top of the **Surfaces** panel, you can effectively use the **Surface Color** channel to help add transparency to the object. With **Additive** selected, the color of anything beyond the surface (including other objects, the background, and even the object itself) gets "added" to the color of the surface. If the color of the surface happens to be dark or black, it will appear transparent when the background is added (anything + zero = anything).

By modulating the surface colors, you can make more splotches of transparency. While you can adjust values in any of the "big three"—**Surface Color**, **Luminosity** or **Diffuse Levels**—I generally use the luminosity and diffuse values. This way, you don't have to worry about areas of the surface that aren't lit, as Luminosity will make the surface appear self-lit.

In Luminosity, add another layer of fractal noise, perhaps making the texture size a bit larger than the Transparency channel (making the pattern seem less regular). Changing the **Texture Velocity** also adds a bit more randomness. And be sure the Texture Value is zero and the Luminosity value is 100%, as in the transparency channel.

To get our dark areas, the **Diffuse Level** should be zero or very low. If you like, add a very subtle fractal noise pattern in this channel to further break up the color scheme of the smoke. Add Bones to the tube for more realism, and use them to expand and contract the smoke slowly, or make it sway with the wind.

4. Pyro: Explosions are a snap with fractal noise. However, if you want to know the nitty-gritty of how to make one, go back and dig up the July 1994 issue of *LIGHTWAVEPRO*. It's described there in full detail, so I refuse to waste any more paper on it here.

3. **Sparks**: As a companion to the non-existent explosion trick, the sparks that go with them can also be enhanced by fractal noise. Although they are created using only single-point polygons, these sparks are nonetheless still affected by all surfacing parameters, including textures and fractal noise.

Try adding noise in the transparency channel of your sparks. This creates a greater variety in luminosities among the particles and a velocity that will make them twinkle. Noise can also be used on all kinds of particle systems. Try adding it to a starfield for a more realistic spread of densities, or to underwater detritus to fake different sizes. Use noise in the color channel as well to add variety.

2. Trickling Water: Whenever you have an object (say, a dolphin) breaking the surface of water, try running a specular-only fractal noise pattern down the Y axis. From a reasonable distance, this creates the effect of water beads dripping down and can greatly enhance realism. Did I say dolphin? Sorry, I meant submarine. Everyone knows that computer-generating a dolphin would be too hard.

1. Clouds: Yes, LightWave's horizon colors may be gorgeous, but they simply look boring without some nice, white, fluffy clouds thrown in. And so easy to make! Just pick up the September 1994 issue of *LIGHTWAVEPRO* and see how it was done for an episode of *Babylon 5*. Oh, OK, stop crying, I'll tell you anyway...

Simply make a large, flat polygon to serve as your sky. Give it a white color and add fractal noise to the transparency channel. With the right settings, the holes the noise creates will look just like clouds! Try stretching the polygon out a bit on the Z axis if the pattern looks too regular.

In addition, chances are that using only one channel of transparency will create too many clouds in too regular a pattern. Remember how the smoke effect managed to use **Additive** combined with **Diffuse Level** and **Luminosity** to "cheat" another fractal transparency channel? Well, you can use the same trick here. Simply click on the **Additive** button and add fractal noise to the luminosity channel. As long as your Diffuse Level is 0%, it will act just like transparency. Just remember to make the **Luminosity** value 100% and the **Texture Value** zero. It would probably also be a good idea to make this texture size larger (maybe even double) to break up the pattern. By tweaking this second channel a little bit, you should be able to produce very realistic clouds.

Well, that's all, folks! I hope you've enjoyed our journey through the wondrous land of fractal noise. With a little imagination, I'm sure that even you can come up with thousands of applications for this wonder-texture. Who knows? Maybe you will astound future generations of *LIGHTWAVEPRO* readers.

Next month, we'll look at the top 10 uses for the Dots texture! Maybe.

LWP

Mojo works as an animator/technical director for Foundation Imaging.

Editor's Message

continued from page 3

hitch.

As long as we're talking about saving scenes out, I should mention that with the advent of LightWave running on new platforms, certain filename conventions have been adopted. LightWave running under NT will default to looking for particular filename extensions. LightWave scenes end in a ".laws" extension (LightWave Scene). Objects end in ".low" (LightWave Object) and images end in the appropriate filename extension, such as .if, .tag or .lwi (LightWave Image). This last one is strictly used in LightWave and can represent any image that LightWave can understand and load. These filename extensions originate from Screamernet, where some type of system was needed to differentiate between certain files sent over to a rendering engine. You certainly don't have to follow these conventions, but I find it much easier to do so, especially since LightWave defaults to looking for them. Also, for me, it's a handy way of differentiating between items moved over from my Amiga and items originally created on the PC.

One final thing to remember for those who will be using new, "green" PCs: For our LightWave benchmarks, I designed a number of scenes to be used to test rendering times. Every time I tried to render the ray trace test on my Pentium machine, it would only get to an antialiasing pass before "locking up." Of course, I set up the scene before going to bed and found it hung up every morning. It turns out that my PC was set to go into "suspend" mode after 15 minutes of inactivity, and it just so happens that this always occurred during an antialiasing pass. Because the first pass had already rendered, and LightWave was just doing math in RAM, there was no "activity" monitored and my PC went into "suspend" mode, whereby the CPU clocks down to zero. I figured this out when I got the scene to render in the background while I was writing. The fix is to simply enter the setup for the BIOS chip during the boot-up process and set it so power management is disabled (suspend mode at least). With this option disabled, the scene renders perfectly. So you may want to check this out if you have a new PC and it uses power management settings.

LWP

In the June Issue:

The June issue takes a look at some benchmarks for Modeler and screen redraws, and features a new column for Amiga users doing double duty as PC people.

Taking That First Step

Actually Doing Something With Bones!

by Arnie Boedecker

Since the release of Toaster 3.0 (LightWave 3.5 for you standalone types), LightWave has taken a huge leap forward with the introduction of Bones. Following soon after were a bundle of informative articles on this new feature. In theory, the idea of Bones was great, until you actually sat down and moved one of 'em. Many users are still struggling to get any outstanding results in a reasonable amount of time.

Now, if you already know how to use Bones, and can get pretty nice results, I'm sad to say that this article has nothing to offer you. Go ahead, turn the page with disgust and mumble some derogatory comment like "Not another Bones article." For the rest of you, who want to actually do something specific with Bones but don't know exactly where to start, this article may help.

Just Do It!

OK, the first question is, "What do we want to do?" The most common thing that you see Bones being used for is to animate the inanimate. You know, give that piece of furniture a spring in its step, make that logo dance its little heart out, etc. So, to keep things within the Toaster arena, let's make a toaster come to life. We'll be using the ChromeToaster object that came with the Video Toaster. Now the question becomes, How should we bring it to life? To keep things simple, we'll put together a walk sequence for our little toastin' buddy. Let's get started.

Boneable Toast

The first step in using Bones to manipulate an object is to make sure that it is "Boneable." What I mean is, you must make sure that your object is constructed in such a way that Bones can distort it without destroying it. Generally, only a few things need to be done to accomplish this. You want to make sure that all of the polygons are tripled. This way, every polygon will have only three points, and cannot become non-planar when distorted, which would cause rendering errors. You also want to merge duplicate points so that the object doesn't "come apart at the seams" while being distorted. So let's make our Toastin' Buddy Boneable! (No snickering please.)

- Load Modeler.

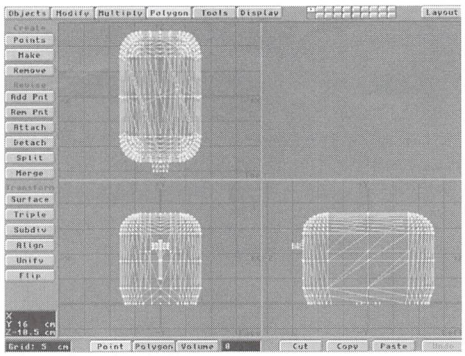


Figure 1: ChromeToaster after polygons are tripled and subdivided.

- Load the ChromeToaster Object (most likely found in the Objects/Kitchen directory) into layer 1.
- Hit the (a) key to fit the object into all three views.
- Triple the object's polygons using the Triple button in the Polygon panel (shift+T).
- To ensure that the object will distort smoothly, you might want to Subdivide the polygons once more (Subdiv in the Polygon panel or shift+D). Be sure to use Faceted Subdivide.
- Next, be sure to Merge Points (Merge in Tools panel or m:) using Automatic merging selection. If you Subdivided once, as I did, 26 points should have merged, and you should end up with 1,248 polygons (Figure 1).
- Save this object as Toaster.sub.

Any Skeletons in Your Toaster?

The next step is to create a skeleton for our toaster out of—what else—Bones! How you plan to manipulate the object determines how you set up your skeleton. I want to make our toaster "walk" forward with kind of an inchworm style, where the front end lifts and moves forward, and then the rear lifts and catches up. Let's do it!

- Quit Modeler.
- Load the object Toaster.sub into Layout.
- To achieve my "inchworm" effect, I'll need separate Bones in the front and rear. I'll also need some extra Bones in the front, which will allow for more fluid movement and help the toaster "dip down."
- Add the first Bone to the toaster by clicking on **Add Bone** in the **Object Skeleton** subpanel (**Objects**).
- Rename this Bone FrontFootBone.
- Give it a Rest Length of 0.1 and hit **Continue.**
- In Layout, select **Bone** as your current edit item and move the Bone to (-0.05, 0.025, -0.125).
- Create a keyframe at frame 0 and hit the (r) key to signify this as the Bone's rest position.
- With **Bone** still selected as your edit item, hit the (p) key to bring up the Bones subpanel, and click

Key	FRONT FOOT BONE(1)	TOP BONE(1)	REAR FOOT BONE(1)
0	-.05, .025, -.125	0, .125, 0	-.05, .025, .015
5	none	0, .125, 0	none
15	none	0, .075, .005	none
20	-.05, .025, -.125	0, .075, .005	none
25	-.05, .060, -.130	0, .095, 0	none
30	-.05, .025, -.175	0, .105, -.012	-.05, .025, .015
35	none	none	-.05, .060, .005
40	none	0, .105, -.012	-.05, .025, -.045
50	none	0, .075, .005	none
55	-.05, .025, -.175	0, .075, .005	none
60	-.05, .060, -.180	0, .095, 0	none
65	-.05, .025, -.215	0, .105, -.012	-.05, .025, -.045
70	none	none	-.05, .060, -.055
75	none	0, .105, -.012	-.05, .025, -.095
85	none	0, .075, .005	none
90	-.05, .025, -.215	0, .075, .005	none
95	-.05, .060, -.220	0, .095, 0	none
100	-.05, .025, -.265	0, .105, -.012	-.05, .025, -.095
105	none	none	-.05, .060, -.100
110	none	0, .105, -.012	-.05, .025, -.130

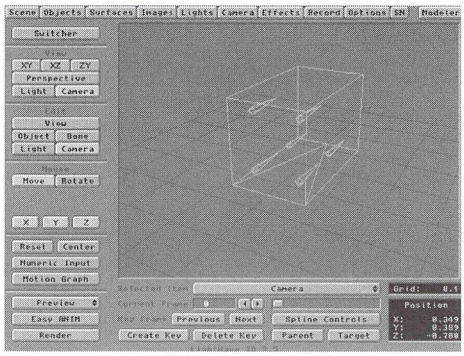


Figure 2: Object Skeleton for ChromeToaster.

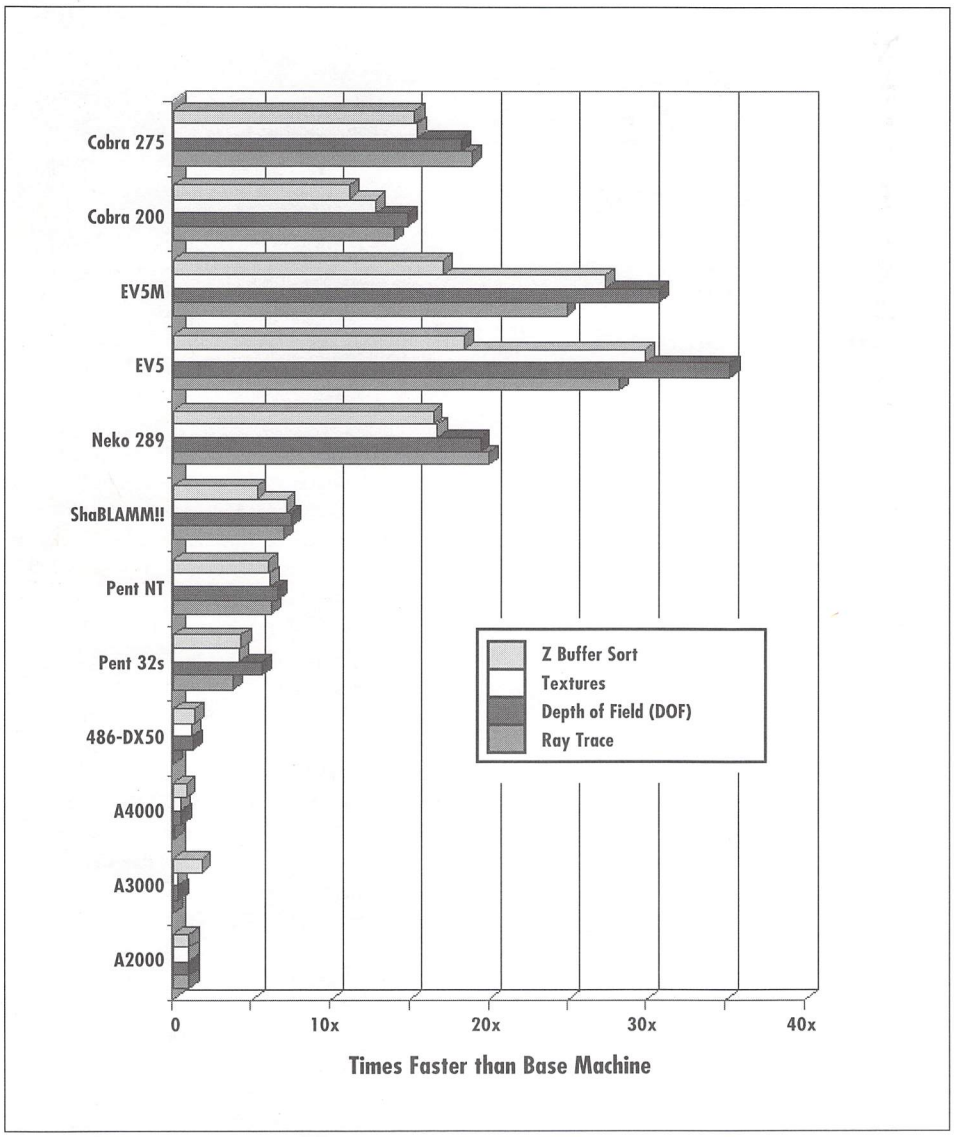
on **Add Child Bone.**

- Rename this new child Bone FrontFootBone as well. You should now have FrontFootBone (1) and (2). Hit **Continue.**
- Move FrontFootBone (2) to (0.1, 0, 0). Make a keyframe at frame 0 and hit (r) to designate its rest position. Both FrontFoot Bones will be controlled by FrontFootBone (1).
- Create two rear Bones in the same manner: Add a Bone, renaming it RearFootBone. Give it a Rest Length of 0.1 and move it to (-0.05, 0.025, 0.015). Create a keyframe at frame 0 and assign its rest position (r). Then add a child Bone, renaming it RearFootBone. Move RearFootBone (2) to (0.1, 0, 0). Create a keyframe at frame 0 and assign its rest position (r).
- Next, we want to add two more Bones up front, which will help the toaster "dip down," as mentioned earlier. In the Skeleton subpanel, add another Bone and rename it TopBone. Give it a Rest Length of 0.1 and hit **Continue.**
- In Layout, parent the TopBone to FrontFootBone (1).
- Move TopBone to (0, 0.125, 0). Create a key at frame 0, and assign its rest position (r).
- As before, add a child bone to TopBone. Rename the child TopBone as well. Move TopBone (2) to (0.1, 0, 0). Create a key at Frame 0, and assign

see Taking That First Step, page 15

LightWave 4.0 Benchmarks

continued from page 5



The EV5 and EV5M are based on the DEC Alpha 21164 processor. Both machines tested were first-run versions of new designs (the EV5 is based on DEC technology, the EV5M is based on technology from DeskStation Technology). Each had a 266MHz processor, but the EV5M was only able to reach the 250MHz mark. Although this did not hinder performance substantially, more refined versions of these designs are on the way and will no doubt show even greater performance figures.

As you can see, the once-trusty Amiga is just not in the same league as the other systems tested. Our first candidate was Intel's Pentium-based, generic PC clone running at 100 MHz. This machine, though not a standout in our testing, did show performance gains that even the most diehard Amiga user would be overjoyed to have. Improvements over our base unit were anywhere from 373 percent to nearly 700 percent. The reason for the differences in rendering time when running under Windows 3.1 and Window NT 3.5 is that regular Windows 3.1 is merely a GUI front-end for the old lackluster 16-bit DOS operating system, while Windows NT is a true 32-bit operating system allowing the processor to receive data in 32-bit blocks.

Next we have the MIPS CPU-based system from ShaBLAMM! This R4400 100 MHz machine is actually a cross between an accelerator card and a computer. It is a computer in that it contains all the necessary components for the processor to fully function, and it is like an accelerator card in that it requires an Intel 486-style PC with a VLB (Vesa Local Bus) slot. This slot is used to power the card, and to communicate with the other cards in the host system. The test results from the R4400 are also impressive compared to the base unit. These tests show the ShaBLAMM! to have anywhere from 525 percent to 738 percent greater performance than our A2000. As you can see, although impressive, the ShaBLAMM! wasn't much faster than the Pentium-based system tested.

Last, and most certainly not least, we have the DEC Alpha CPU-based systems. These are by far the supreme performers for LightWave, or any other application you have to run. Throughout the testing, Alpha-based machines performed with outstanding results, even from the "slowest" versions of the chip. All of the Alpha machines, with the exception of the Cobra 200 (32MB RAM, 512K cache), ran 64MB of RAM and 2MB of cache. As the graph shows, all of the Alpha processors soared in the benchmarks, with relatively "scaleable" results based on processor design and speed. Performance improvements over the base system are anywhere from an amazing 1,100 percent to an unbelievable 3,500 percent. That's 35 times the power of our A2000!

Well, from these results, it's pretty obvious that if you

see LightWave 4.0 Benchmarks, page 13

Cut It Out!

Moving Clouds With Only One Image

by Dan Ablan

Can someone answer a question for me? Why is it that every client is always on a tight budget when it comes to animation? A budget that is not only short on money, but time. This tutorial is from a situation that came up a couple of months ago in one of those much too familiar animation projects. I was hired to put together a logo for a new business that was starting up in the area. The job was for a former Chicago weatherman who was also a pilot. He had the idea that the logo we were animating should give viewers the feeling that they were flying. And, of course, there's only about a day and a half to get it done. So, I came up with a few ideas, presented them to the client, and they weren't what he had envisioned. Later that same day, I met with him again, and showed him his logo flying out from behind some clouds. He liked that very much. Remember, he's a pilot. "But Dan," he asked, "what would it take to get these clouds moving? Can you get digitized clouds?" Well, the answer was "Yes, but not in one day." It would have taken too much time to locate the right cloud footage and have it digitized. Not to mention the realities of that limited budget thing. So, somehow, I came up with the idea of animating the clouds from one still image.

The Right Image

What I noticed while working on this project is that is very important to find the right set of clouds that won't be too difficult to pull apart. Luckily, I happened to have a CD of 100 royalty-free PhotoCD images of clouds. This is something that I was very thankful for. When I began animating full time, I began collecting anything I could, even if I didn't need it at the time. And I started a file containing items to scan, such as a beer bottle label, texture samples, and so on. When I came across the right CD sets, I picked up as many as I could. That way, when it was down to the wire, I wasn't spending my time looking for a decent image. When looking for an image, especially clouds, find one that has some definite separations in it. A cloud image that is too intricate will be a great pain to use for this tutorial.

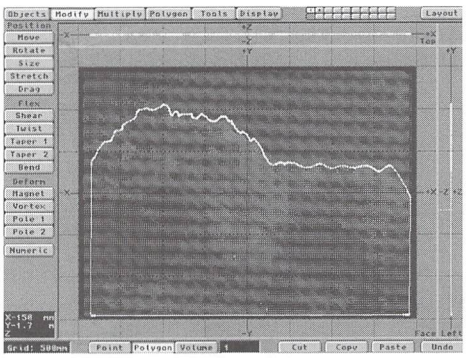


Figure 1: A polygon matching the shape of one portion of the original cloud image is constructed in Modeler.

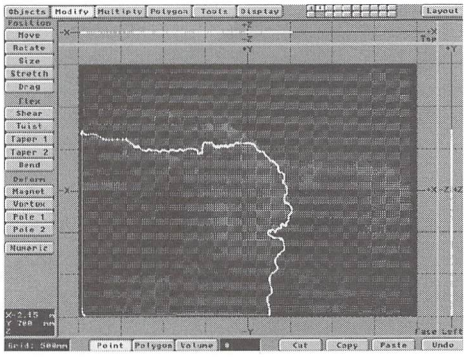


Figure 2: Another polygon is made matching the shape of a different area of clouds.

How It Works

All we're really doing here is taking one image (see Color Pages), cutting it up into two pieces, and painting the existing piece for a background. Once I selected the image, I loaded it into LightWave. Then, in Modeler, I used the display **BG Image** feature to see my clouds. Using the create point tool, I began setting points around the first cloud that I intended to move (Figure 1). I made the second cloud the same way (Figure 2). Next, I exported the first cloud object to LightWave and rendered it out fully white and luminous, and did the same for the second cloud object.

Once the rendering was finished, I went into ToasterPaint and called up the first cloud polygon image. Using the (j) key, I swapped screens and loaded the original full-color cloud image. Next, using the other layer as a template, I began cutting out the first cloud from the color image, to match the first polygon (Figure 3). The same process was performed on the second cloud (Figure 4). Using the white rendered polygon images as templates, I then guestimated where they would be when they started moving and where they'd end up. Based on that knowledge, I began painting out the clouds from the full-color image, which would be mapped onto the flat polygons (the ones that were just cut out). This was one of the most important parts of the project, because if I cut a section of clouds from a larger image and mapped it onto a polygon that moves over that original image, the same cloud would be revealed underneath. It was crucial that I change the final background image, or else the illusion wouldn't work. I did the same for the area of the second cloud.

Figure 3: Using the newly created polygon as a guide, the actual image is cut to match.

Soft Polygons

I was surprised that this idea was actually working at this point. But, given the peculiar size of the image and polygon, I had to mess with the sizing quite a bit to have the image cover all edges of the polygon. **Automatic Sizing** wasn't enough. The other major problem I encountered was that it was very obvious that there were polygons in the scene with cloud images mapped on them. Then, I remembered the good old transparency map trick. I went back into TPaint and loaded the first black and white cloud

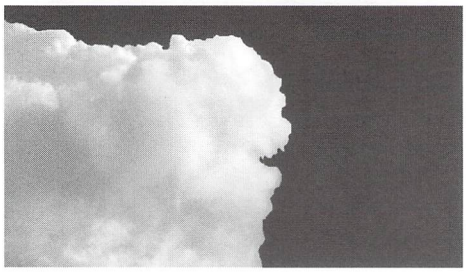


Figure 4: The same process is done for the other polygon.

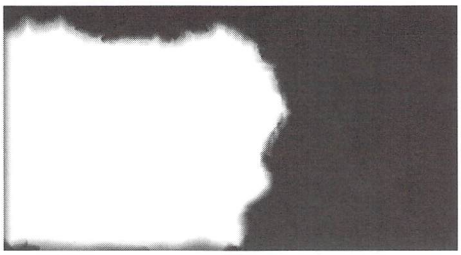


Figure 5: A transparency map is made for each cloud polygon.

polygon image. I proceeded to fade off the edges all the way around the polygon (Figure 5). A transparency map reads the black and white values, and whichever part of the object has the lighter part of the image mapped onto it will be more transparent. When the color fades from black to white, the object fades from itself to transparent. To get this image to work, we will need to use **Negative Image** in the transparency panel. This way, the black and white values will be reversed, causing the image to fade away to white. This feature is really good for making those streaking wisps that you see flying behind a nice logo. You could also use a transparency map to make spaceship trails that fade off to nothing. Cut and save these as brushes.

I finished making the transparency maps and went back into LightWave. The transparency images were mapped using a planar image map on the Z axis, under the **Transparency texture** (T) button. I used the exact same size coordinates I had for the actual image map on each polygon.

Animation Speed

After a good bit of tweaking, it was difficult to see where the image-mapped polygons ended and the background cloud began. Because of the soft falloff on the edges, you couldn't see a seam unless you were looking for it. The animation still needed to be accurate, and look like real clouds moving. One step was taken care of by using a real image. The next step was to set the movement and timing right. The first cloud polygon was moved along the X axis from left to right and the second polygon was moved from right to left. Each moved just enough to be noticed, but didn't move fast. As a matter of fact, the amount they move across the screen is too minimal to show the difference with just a still image. The speed matched that of real clouds—at least, that's what the pilot told me.

As with any logo or object, it's a good idea to add some spline control to the movements. Even with the cloud polygons, I set a tension of 1 at the first and last keyframes. Now, all that was left was to put the logo in. The client liked that it traveled out from behind the clouds, so that idea was kept. Back in Modeler, I made a very simple polygon to use as a front projection image map on the clouds. This was placed behind the moving cloud polygons, with the final background image front projected onto it. Finally, the logo was animated up through and over the clouds to its resting position in front of them. Additional titles slid in from the left and painted brush strokes were revealed across the top of the logo. A transparency map texture was set to the Front Projection image map type so that there was no hard edge when the logo appeared (see Color Pages).

In the Beginning

When I was learning how to animate, I believed that in order for an animation to be decent, it had to be very intricate, with several trade secrets going into any project. What I soon learned is that some of the best-looking animations are the simplest. Some credit should be given to those who are purists, and build an object to exact scale, without any missing details. Then, there is the practical side, where clients get what they pay for and it looks good on tape. With a little work and some thought, you can come up with some terrific ideas.

LWP

Creating Cloud 9

1. Make a polygon the size of the part of the image to be animated.
2. Render it out fully luminous, so that you have a completely white surface.
3. Use ToasterPaint or a similar program to cut out the same areas of clouds for an image brush.
4. Use ToasterPaint or a similar program to soften/fade off the edges of the white rendered polygons.
5. When softening, use the range tool to fade from white to black, about 1/4-inch from the edges.
6. Remember that the ramped values from black to white will act as the "falloff" for the cloud polygons.
7. Cut out and save these fixed white images under a different name, for safety.
8. Place the black and white image as a transparency map, with the same size coordinates as the image maps.
9. Move the cloud polygons from left to right at a very smooth, slow pace.
10. Make one more polygon to match the top edge of the clouds for use as a front projection image map. This will allow you to "hide" a logo behind the background image of the clouds.

Dan Ablan is a LightWave animator for his Chicago-based company, AGA. Because his 8-track tape player recently broke, he can't listen to his favorite Bay City Rollers tapes, and luckily, has extra time to write for LIGHTWAVEPRO. *Dan can be reached via the Internet at dma@mcs.com.*

LightWave 4.0 Benchmarks

continued from page 11

are serious about LightWave rendering times being the fastest they can be, you should be running LightWave on an Alpha. If you need a fairly fast machine, but want to be able to run all sorts of Windows applications, a 100MHz Pentium would be a good idea. The other thing to keep in mind when running LightWave on Intel-based machines is that Windows NT is definitely the way to go. For the price of NT, the speed increase, and the ability to use long filenames are worth every penny.

LWP

Companies mentioned:

Carrera Computers
23181 Verdugo Dr., Bldg. 101
Laguna Hills, CA 92653
(800) 576-7472

DeskStation Technology
13256 West 98th St.
Lenexa, KS 66215
(913) 599-1900

NekoTech
102 Tide Mill Road Ste. 6
Hampton, NH 03842
(800) 635-6895

ShaBLAMM! Computer Inc.
21040 Homestead Rd., Ste. 201
Cupertino, CA 95014
(408) 730-9696

Aspect Ratios

by Mark Thompson

From day one the Amiga has had the unique advantage of being designed specifically for video use. This, of course, is one of the major reasons the Video Toaster was developed for the Amiga and the Amiga alone. However, this design feature has not come without a few "quirks." No veteran Amiga user can forget the wonderful flickering screens we were forced to working on until scan doublers were introduced. But another more insidious and hidden quirk has brought grief to Amiga graphic artists everywhere: the pixel aspect ratio.

Pixel Aspect Ratios

Unlike other formats such as the PC, the Amiga's pixel aspect is not square. This can lead to various problems when rotating a graphic element on the screen or transferring imagery to or from other systems.

So what exactly is the pixel aspect ratio? It is simply the width divided by the height of an on-screen pixel for a given display device. To determine the pixel aspect, you must first know the relative size of your display. In the case of NTSC video, it's a known fact that the screen aspect is 4-to-3. Assuming that the screen area for a given digital video format in NTSC is constant, it follows that the [(Horizontal Resolution) / (Vertical resolution)] x (Pixel Aspect) = an NTSC screen aspect of 1.333. For D1 video, the resolution is 720x486. Applying these values to the above equation yields a 0.9 pixel aspect. However, the Toaster (and the PAR) uses a D2 aspect and sample rate. D2 resolution is 754x486, and although the Toaster only uses a portion of the frame (752x480), for the sake of convenient numbers, it still has the same aspect. So the D2 or Toaster pixel aspect is 1.333 x 486 / 754 = 0.859. As mentioned earlier, the PC uses a square (1.0) pixel aspect. To make sure the above equation applies to the PC as well, plug in the numbers. The result is that 640 / 480 x 1.0 also equals the 1.333 NTSC screen aspect.

Frame Aspect Ratios

If you examine the **Camera** panel in LightWave, you will find that these numbers are provided for you based on the digital video format or resolution you have selected. Note that there are also two PAL format choices. A PAL screen occupies the same physical space that an NTSC screen does, but there are approximately 100 more scan lines in that space. This results in a significantly larger pixel aspect ratio. There is one other number presented in the panel of interest, the **Frame Aspect Ratio**, which is the result of [(Horizontal Resolution) / (Vertical Resolution)] x (Pixel Aspect). For video resolutions, this sounds essentially the same as the 4-to-3 NTSC screen aspect we have been talking about. However, there is a subtle difference. Rather than using the resolution that is defined by the video standard, the actual rendered resolution is used instead. And since the Toaster only uses a portion of the D2 screen space, this results in a frame aspect ratio that does not exactly match 1.333. In the case of NTSC, the frame aspect comes out to 1.346, while PAL is 1.330. These numbers turn out to be quite significant.

Matching Video With LightWave Frames

Let's say you want to use LightWave to create high-quality DVEs much like the third-party packages Visual FX and Hollywood FX. A typical use might be to fly the video off into the distance while the new video flies into view. In order to do this, you must map your video imagery onto a rectangular polygon that exactly matches the camera view. The slightest deviation causes a visible jump in size and/or position when going from the live video to the mapped video and back. This is where the LightWave Frame Aspect Ratio comes in. A rectangular polygon possessing dimensions with the same aspect will fit the camera perfectly in both X and Y. It's a common mistake to think that a 752x480 rectangle will do this. This shape is way off for a D2 frame since it does not account for the pixel aspect. Here is the general procedure:

- Make note of the Frame Aspect Ratio (FAR) in the Camera panel.
- In Modeler, create a box with the dimensions
  Xlow = -FAR
  Xhigh = FAR
  Ylow = -1
  Yhigh = 1
  Zlow = 0
  Zhigh = 0
- Save the object, load it into LightWave and create a key for it at 0, 0, 0.
- Set the camera's **Zoom Factor** (ZF) to your desired value and then place the camera at position 0,0,-ZF.

The result is a polygon that is perfectly sized and positioned for that camera's field of view. So for the most common case (D2-NTSC with 3.2 zoom factor), the product is a polygon sized 2.692x2.0 and the camera at 0,0,-3.2. If your output is to D1 or a PC video card such as a Targa, the polygon size would be 2.6667x2.0.

What if you want a polygon of some other size to fill the camera view? As long as the exact X-to-Y ratio matches the **Frame Aspect Ratio**, there's no problem—make the shape any size you like. Take the height of the object and divide it by two. Then multiply the result by the negative camera zoom factor and you will have the correct camera Z position. So if you have a polygon with a height of 15.5 m and a camera zoom factor of 8, the camera should be placed at Z = -62.0 m. If you were using a camera set for D2 (PAL), the polygon would be 20.61 m wide.

What about custom resolutions? The only thing affected is the frame aspect, and it changes according to our first equation described above. So the camera position remains the same, but now the width of our polygon should be 15.5 m times the new Frame Aspect Ratio.

At this point you have meticulously followed all of my instructions and set up a polygon that perfectly frames the camera. However, after rendering out some images, upon close inspection of the frames you'll notice the background poking through ever so slightly on the left edge. Well, before you begin shredding this article and exclaiming "Mark Thompson is a clueless bonehead," you should know that due to round-off errors, this one-pixel-wide gap can occur. Allen Hastings is aware of this minor problem and has a fix for it that may show up in the final 4.0

release. For video, the obstacle is undetectable since it's at the edge of the overscan area. But if it presents a problem for your application, size the polygon just slightly wider.

$$\frac{\text{Horizontal Resolution}}{\text{Vertical Resolution}} \times \text{Pixel Aspect Ratio} = \text{Frame Aspect Ratio}$$

$$\frac{\text{Frame Polygon Height} \times \text{Zoom Factor}}{-2} = \text{Camera Z position}$$

Frame Aspect Ratio equations

Importing/Exporting to Other Platforms

Importing and exporting graphics between different platforms is another area where aspect ratios in LightWave become important. And with the release of LightWave 4.0 on the PC, Alpha, MIPS and SGI, the likelihood of such transfers is greatly multiplied. Fortunately, all the new LightWave platforms typically use a square pixel aspect. So transfers between these platforms and the Amiga are the only ones of real concern.

First consider the case of importing images to the Amiga. At 640x480 resolution with a square pixel aspect, images from the PC (and the other platforms) will appear squished in the X direction when displayed on the Amiga due to the Amiga's narrower pixel aspect. To correct this, the image must be scaled in the X direction by 1.164 (1/0.859), which yields an image of 745x480. If you want the result to be 752x480, your PC image will have to originate as 646x480. Some PC software allows you to set the pixel aspect ratio. In this case, set it to 0.859 at 752x480 and it will import perfectly into Amiga LightWave without scaling.

The process of exporting Amiga images to the PC is essentially the same, only inverted. Scale the Amiga images by 0.859 in the X direction. If you want them to end up 640x480, start with 745x480 on the Amiga side. Of course, if you are using LightWave to generate the images, simply select Square Pixels and render at 640x480. Note that when bringing an image into LightWave as a background, foreground or alpha element for compositing, the image will be scaled to match the output resolution. It is therefore important that your images match output if you do not want distortion to occur while compositing.

There has always been a fair amount of confusion as to what to do when rendering images for print output. It's actually very simple. First, always use a square pixel aspect ratio. Then, to choose your resolution, consider the size in inches your image will occupy on the page and what the dot density (dpi) will be. If it's going to be a 5.25-inch x 3-inch image at 600 dpi, that means your custom LightWave resolution should be 3150x1800. Note that the print industry relies upon lpi rather than dpi and that 1 lpi = 2 dpi. Also, magazines commonly print graphics at 133 lpi or 266 dpi. Refer to John F.K. Parenteau's "Digital Cinematography" in this issue for information on dealing with film aspect ratios and resolutions.

LightWave to the Rescue

Before wrapping up, there are a few new features in LightWave 4.0 that are quite relevant to our topic. The **Camera** panel now allows you to specify a custom pixel aspect ratio. This is of particular interest to people who wish to use LightWave for non-video applications such as film. Between the custom resolution settings and the new custom pixel aspect, LightWave can appropriately deal with any digital image input or output medium. Another notable feature is frame boundaries for Layout. These dotted borders indicate the actual active area for a selected resolution and aspect. So as your pixel aspect varies, the borders shift to reflect the change in frame aspect. Before, it was very difficult to rotoscope to live action with anything but D2 aspect footage. The new frame boundaries correct that problem. Finally, a new **NTSC Widescreen** button has been added to the Camera panel. It increases the camera's left-to-right field of view by four-thirds without affecting the pixel aspect.

While fundamentally a simple topic, aspect ratios continue to be a source of confusion. It's even easier to become puzzled when going back and forth between dissimilar systems with multiple aspect changes. Hopefully, this article has shed some light on the topic and helped eliminate some of the confusion.

I would like to extend a special thanks to Allen Hastings for clarifying some of the technical details regarding aspect ratios in LightWave.

LWP

Mark Thompson is director of animation at Fusion Films, Inc. Send questions or comments to Fusion Films, Inc., 51 Derry St., Merrimack, NH 03054, or e-mail to mark@fusion.mv.com.

CALCULATION CHECK

Aspect numbers computed to a higher precision than shown in LightWave:

D2 (NTSC) pixel aspect = 0.859416

D2 (NTSC) frame aspect = 1.346419

D1 (NTSC) pixel aspect = 0.9

D1 (NTSC) frame aspect = 1.333333

D2 (PAL) pixel aspect = 1.018568

D2 (PAL) frame aspect = 1.329797

D1 (PAL) pixel aspect = 1.066667

D1 (PAL) frame aspect = 1.333333

Taking That First Step

continued from page 11

its rest position (r).

- Save the scene as Toaster.Skeleton, or something similar. That's it! You've got a skeleton to manipulate your toaster (Figure 2).

Walk the Walk

OK, let's put this toaster into action! The movements forming the walk sequence will be as follows: The front end will dip slightly, as if to "push off." The front end will rise from the ground, move forward, and descend to the ground again. Then the rear will lift and move forward to meet the front again. Ready?

- Begin by creating a keyframe at frame 5 for TopBone (1), and give it a spline tension of 1. This will let the toaster rest for a few frames before going into motion.
- To sink the front end down slightly, move TopBone (1) down to (0, 0.075, 0.005) and set a key at frame 15. Make a key at frame 20 to give the dip a short pause.
- The toaster will now begin its first step. Create a key for FrontFootBone (1) at frame 20, with a tension of 1, and then move it to (-0.05, 0.06, -0.13) and keyframe it at frame 25. Move TopBone (1) to

see Taking That First Step, page 17

What's on the Disk:

This month's *LWPRO* disk includes most of the benchmark tests and objects as well as some Film Aspect ratio scenes.

LightWave 4.0 Preview

by Mark Thompson

It's arrived: the long-awaited 4.0 release of NewTek's premier software package—LightWave. Or has it? What is being distributed is actually a prerelease that gives us a very good idea of what the final 4.0 version will look like. It is sort of like the Flyer 0.9 "under construction" release, except perhaps a bit more solid. So don't be completely stunned if, while working, you are met with the always-disturbing "software failure" message. Many of the major bugs have been squashed, but it's still an unfinished program, mostly for those of you who just can't wait. But enough of the scare tactics and warnings. On to what everyone is most interested in: what's in the new LightWave.

Unlike previous releases, the list of new features is not large. This is due to the huge amount of effort required for two major new additions: multi-platform operation and software plug-ins. Software plug-ins open up LightWave to a whole new range of capabilities from third-party developers. These will allow its feature set to grow immensely, as if hundreds of people were working on LightWave development rather than just Allen Hastings and Stuart Ferguson. And unlike previous third-party products, they will seemlessly integrate with LightWave. Programs like MetroGrafx's Sparks will become easier to operate by virtue of having all of LightWave directly connected to them. And, of course, one of the most anticipated new plug-in capabilities is greatly expanded texturing functions. No longer will you be limited to just a few basic procedurals.

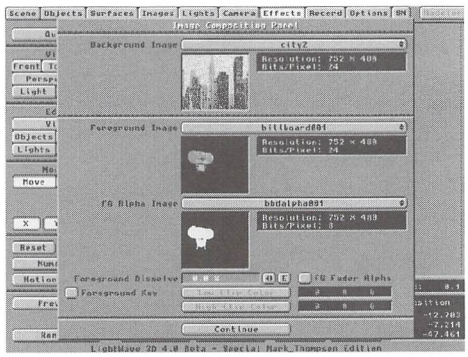
Beyond plug-ins, there have been numerous enhancements made to LightWave. The **Scene** panel now allows you to toggle through different colors for each of the items in the scene overview (non-Amiga versions only). This will change an item's color accordingly in the Layout window. The **Objects** panel has two new buttons: **Unseen by Rays** and **Unaffected by Fog**. Unseen by Rays brings additional control to selective ray tracing by allowing you to exclude individual objects from reflectivity and refraction calculations. These objects will not be reflected in a mirror or distorted in a refraction. Tremendous reductions in render time can be achieved in complex scenes by excluding everything but the most critical objects. Unaffected by Fog does exactly what the name implies: it prevents objects from being obscured by fog settings. This can be handy when you have created a lighting effect based on a modeled object and you do not want the effect faded or diminished by current fog settings.

The **Surfaces** panel has several new additions. The most prominent is a preview area that has a number of windows (depending on your resolution) for sample surface renders (NT and SGI versions only). On the Amiga, sample surface spheres will be rendered to your selected Display Device by using the (s) key shortcut. With surface samples, you have virtually instant feedback about what a particular surface setting will look like on your object. And the previews exist in all levels of the surface menus so you don't have to back out of a texture specification to see the results. The samples are rendered on a sphere of definable size, and planar samples will likely be added in the final release.

Reflections now have a subpanel with several different options, including Backdrop Only, Ray Tracing + Backdrop, Spherical Reflection Map, and Ray Tracing + Spherical Map. But by far my favorite new surface feature is the Glow Effect. This post-processing operation creates a soft glow on and around the specified surface. Now, instead of modeling slightly larger transparent surfaces around your glowing object, simply toggle on Glow Effect and you're all set. This greatly reduces the need for clusters of lens flares for various effects. [Editor's note: The glow effect has been removed from the prerelease version since it will be reintroduced as a plug-in in the final shipping version.]

The significant changes to the **Lights** panel are mostly related to lens flares. Several new flare options have been added. **Anamorphic Squeeze** has been renamed **Anamorphic Distort** and an associated distortion factor parameter has been added. There are now inputs for **Streak Intensity, Streak Density** and **Streak Sharpness**. Some nice effects can be achieved by softening the sharpness value while reducing the density. Envelopes for these three new parameters would make a nice addition.

Outside of the flare subpanel, a Global Flare Intensity envelope has been added, which will be very useful when dealing with large numbers of flares. The



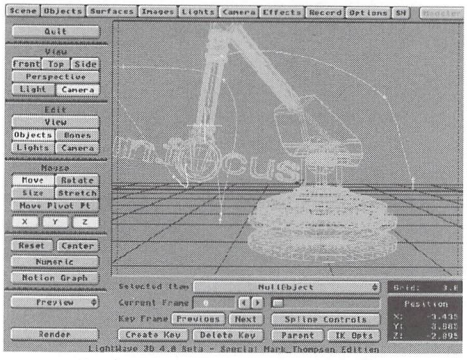
LightWave's new Image Compositing panel

one non-flare-related addition is an envelope for light intensity falloff. I haven't dreamed up a good use for this one yet, but I'm sure you will.

There are only a couple of new additions to the **Camera** panel. First is the ability to specify a custom pixel aspect ratio (see my related article on aspect ratios in this issue). This setting allows LightWave to better work with any input or output image format. Second, a button labeled **NTSC Widescreen** increases the camera's left-to-right field of view by 1.333, without affecting the pixel aspect. This is handy for creating animations intended for NTSC widescreen format. Also, for informational purposes, LightWave now reports the camera field of view angle in both the X and Y directions. This angle is affected by the zoom factor, the pixel aspect ratio and the state of the NTSC Widescreen button. **Custom Size** resolution values now work exactly as they should instead of incorrectly scaling the output image to 4-to-3 aspect.

Earlier I mentioned the new glow surface attribute. Well, the intensity and radius of the glow effect is globally controlled within the **Effects** panel. [Editor's note: As mentioned earlier, the glow effect is not included in the prerelease but will be in the final shipping version. I'll leave reference to it here so you can get a feel for what it does.] Therefore, all surfaces with glow enabled will have the same intensity and radius. You have some individual control over glow intensity by changing the surface's effective brightness via the surface attributes. The intensity and radius settings in the

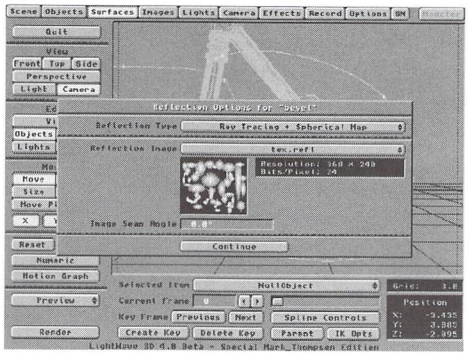
Effects panel also have associated envelopes, allowing for much more dramatic glow effects. The image-compositing controls, which used to occupy the upper quarter of the **Effects panel**, now have their own subpanel. The functionality is the same but thumbnail preview windows (like the one in the **Images** panel) have been added so you can see the images you have selected for compositing.



An example of a scene using Inverse Kinematics

The **Record** panel's only addition is a button for selecting different output filename formats. The **options** available are Name001, Name001.xxx, Name0001 and Name0001.xxx. The "xxx" denotes the filetype and is either iff, tga or raw. The Options panel also has one new feature, a button labeled **Show Rendering in Progress.** When enabled, a low-res color image of the frames being rendered is displayed. This is much like the old gray preview screen in previous Amiga versions, but it now takes advantage of the color capabilities of the display card in your PC, Alpha or MIPS box.

Outside of the panels in Layout, there are a few very significant changes. The first is the widely publi-



LightWave's new Reflection Options subpanel

cized new inverse kinematics capabilities, which, believe it or not, are all enabled from a single button. It is labeled **IK Opts** and brings up a scrolling list requester for a goal object. To make IK work, after setting up a hierarchy for a set of objects or Bones, the child object of a hierarchical chain is adjusted to be goal-directed toward some other object (usually a null placed near that child object). Then, as the goal object is moved around, the child and the parents up the hierarchy rotate to follow the goal. Additional control is provided for each object in the hierarchy by enabling or disabling each rotational axis in the Mouse section of Layout. The current implementation is very basic, but can still make many animation tasks much easier.

The overall Layout panel has changed in two ways. First, it's now resizeable on the new platforms with a minimum size of around 640x480. Second, dashed-line camera frame boundaries indicating the camera's limit of view have been added to the Layout window. These are particularly important when resizing the Layout panel or using various output resolutions and aspect ratios.

But of all the new LightWave features and changes, by far the most significant is the multi-platform operation. As much as I have been a dedicated Amiga enthusiast from the day I bought my first A1000 back in September 1985, I have recently watched with horror and dismay as its development has slowed to a painful halt while other systems improve exponentially. This is not to say the Amiga is no longer a useful machine. My packed-to-the-gills 2000 is used in my studio daily. However, the speed provided by the new generation of machines is too great to ignore. I have been working with a beta copy of LightWave PC on a 289MHz DEC Alpha workstation, and with it, my productivity has multiplied dramatically. Modeling is a joy, Layout previews blow by, and rendering is typically a magnitude faster. Opening LightWave up to other platforms truly makes the program a viable solution for big-screen production. Besides the speed advantages, bringing LightWave to the PC will yield explosive growth in third-party support products, making the new plug-in capabilities all the more important.

So is the latest version of LightWave a software revolution or evolution? Well, it certainly doesn't represent the wealth of new possibilities that came with the 2.0 or 3.0 releases. In fact, the list of new features is relatively sparse. However, the much-needed groundwork has been laid for tremendous expansion. And the improvements in productivity far outshine any flashy features that might have otherwise been implemented. With this only being a pre-release, who knows what other goodies may slip into the final 4.0 software?

LWP

Mark Thompson is director of animation at Fusion Films, Inc. Send questions or comments to Fusion Films, Inc., 51 Derry St., Merrimack, NH 03054, or e-mail to mark@fusion.mv.com.

Taking That First Step

continued from page 15

(0, 0.95, 0) and key it at frame 25, as well.

- The front end will descend to the ground, so move FrontFootBone (1) to (-0.05, 0.025, -0.175) and key it at frame 30. Move TopBone (1) to (0, 0.105, -0.012) and key it at frame 30 as well.
- At this point, the rear begins to rise, so create a key for RearFootBone (1) at frame 30, with a tension of 1. Then move RearFootBone (1) to (-0.05, 0.06, 0.005) and key it at frame 35.
- The rear will descend to the ground. Move RearFootBone (1) to (-0.05, 0.025, -0.045) and key it at frame 40.

Our toaster just took its first step! (Honey, get the camera!) To have our Toastin' Buddy continue walking, repeat the steps we followed for the first step, only continue to move the bones on the Z axis. You'll notice that the object itself never moves, as it only has one keyframe (so don't try a bounding box preview!). It's the bones distorting the object that actually make it move. Keyframes for the next couple of steps, as well as the ones just covered, are detailed below. From there, try and figure them out yourself.

Ready to Render

Once you're happy with the walk sequence, you're just about ready to render. The ChromeToaster's default surface has fractal bumps and reflects its surroundings (it's chrome). To render this out you might want to remove the BumpMap (rendering will be much faster) and you might want to make the toaster reflect an image like fractal reflections, rather than its surroundings. If you tried to test render the walk sequence using only the default black background with the current surfaces, the chrome would just be black. One other note: I found that the black slot on the front of the toaster flickered during a test animation, since the slot is just a polygon placed on the side of the toaster. To fix this problem, I used the Stencil tool (**Stencil** in the **S Drill** requester, **Tools**) to cut in my own slot. If you do this, be sure to triple the new polygons.

Finally, with the addition of a floor and Trace Shadows (Camera panel) on, the toaster comes to life as it struts its way across the screen (see Color Pages).

What's Next?

Anything you want! Once you get the feel for how Bones work, you can make your objects do all kinds of things. Mess around and you'll have your toaster jumping through hoops at a circus in no time (see Color Pages)!

LWP

Arnie Boedecker has sold his spleen to be allowed on the Internet, and can be contacted at awb@ais.net. Or you can call him the old-fashioned way at (815) 385-8198. Arnie has no interests other than animation, and should therefore be considered potentially dangerous.

continued from page 7

great on screen, it would never suffice for delivery to a film scanner. The science of masks on projectors is hardly that, but rather a close approximation of the standard established. The final product may actually go to television someday, and the 1:33 aspect is taller than 1:85.

2) Though no adjustments are required when rendering your image, it is important to understand the difference between flat and academy frames. Though the film negative can be exposed in any manner you desire (film is just a long strip of light-sensitive material; the frame is established in the camera), certain areas, by industry standard, are reserved for various uses. Thirty-five-millimeter film, as exposed in-camera, is considered Flat in that it is exposed in the center of the negative, relative to the left and right edges of the frame.

Finished 35mm film, called a Release Print, must allow space for the soundtrack on the right edge. Thus, the image is actually moved to the left to avoid this space. This adjustment is called Academy alignment. The process live-action film goes through from exposure on set until release print is long and arduous. Many versions are printed, sliced up, re-processed, optically affected and manipulated. Live action tends to remain in the Flat format until the final processes add sound to picture. When CGI records to film, none of these processes are likely to occur, unless the client intends to re-process your negative. If your CGI is to be cut right in with other finished negatives, requesting Academy from the facility that transfers your digital imagery to film would most likely be appropriate.

Now that you have a basic understanding of various frame aspects, let's break our concepts down to CGI language. Though the frame aspects of 1.85 and 1.33 are important to understand, the actual frame size is larger. Under the **Options** panel in LightWave, you will see a new button (as of 4.0) labeled **Show Field Chart.** This is a standardized chart that separates the vertical and horizontal frame into segments. Each direction is broken into 24 segments, reaching to the edge of the frame.

Go into Modeler (I've provided some of these objects on this month's *LIGHTWAVEPRO* disk, but it is handy, and simple, to construct them on your own) and create a box that is 1.85 meters wide and 1 inch high. Make sure the object is centered and load it into LightWave. In the **Camera** panel, set **Custom Resolution** to an Academy 2K resolution, 1828x1332, and the **Basic Resolution** to **Print**. It is important to reset the **Pixel Aspect** selector to **Custom** and input an **Aspect** of 1.0. With **Field Chart** on, move the box you just created toward the camera on the Z axis only, until the top and bottom edges rest halfway between the eighth and ninth hashmarks from center. Make sure your screen (if you're working on a PC) is large enough so you can see the dotted lines of the edge of frame. You have now established a 1.85 aspect ratio to frame your action (Figure 7). Hopefully, by the time this makes it to press, Allen Hastings will have added these custom safe areas for film into the program, but just in case, you now know how to make them yourself.

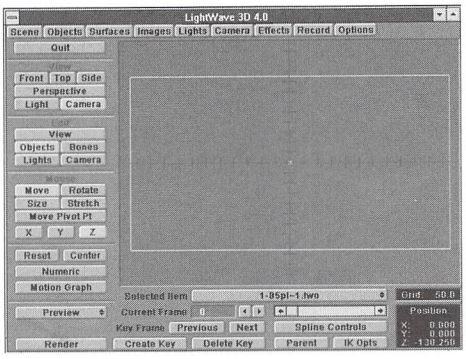


Figure 7: A 1.85 matte positioned in Layout.

Though you are framing for 1.85, there is a lot of area above and below the frame, and a bit left and right, not within your new motion safe area. Examine the Frame Aspect information field in the **Camera** panel. Notice that though we are framing for 1.85, our frame is actually 1.372. Go into Modeler and create a box 1.372 wide by 1. Bring that into LightWave and pull it forward on the Z axis. You'll notice that it fits exactly to the dotted lines left and right and top to bottom of frame. This is called Full Aperture, the entire exposed area of the film.

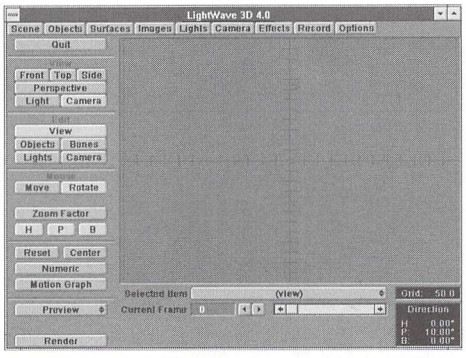


Figure 8: The 12-field chart now provided by LightWave assists in proper framing.

Figure 8 is a full-field chart that includes the fields and framing areas for 1.85, 1.75, 1.66 and full-aperture film. LightWave will always generate an image that fully fills the motion-picture frame. How you frame the images within that frame is up to you.

Anamorphic film squeezes an extremely wide image into the area with a frame aspect of 1.175. Earlier we discussed how pixel aspect was dependent on the display device. For most applications in motion-picture film, a standard square pixel with an aspect of 1 is used. Anamorphic, however, has a pixel aspect of 2, with pixels twice as wide as they are tall once they are expanded in projection. Standard 2K resolution for anamorphic, or Cinemascope (simply a brand name for anamorphic), is 1828x1556.

Input this resolution under **Custom Size**. If you leave the pixel aspect at 1, Layout will show a tall, thin frame. Set **Pixel Aspect** to 2 and look in Layout. Now the frame is much wider than it is tall—exactly 2.35 times as wide. When rendered, LightWave will compress the frame horizontally to fit within the film frame. On 35mm film, the 1.175 frame aspect is smaller than full-frame film, allowing room for the optical soundtrack. In actuality, 70mm film is 65mm film with an extra 5mm horizontally added for the magnetic soundtrack. Thus, an anamorphic image, blown up to 65mm film, fills the entire viewable area. In this case, it isn't important to provide information for the area above and below the frame since there is none! (A handy function of the resizable window in the NT and SGI versions of LightWave comes in handy at this point, and makes for a cool-looking interface. If you are working in Anamorphic resolution and your monitor is large enough, re-size your window lengthwise to the edges of the dotted lines. It not only looks cool, but it's fun to work in wide screen!)

Finally, I've listed common resolutions for various film formats. Most film work is done at 2K resolution. 4K is nice, but, of course, takes much longer to render. Remember, Flat means full-aperture film, and does not allow for the soundtrack on standard 35mm film. Anamorphic does not need to leave space for the soundtrack since it is added on later.

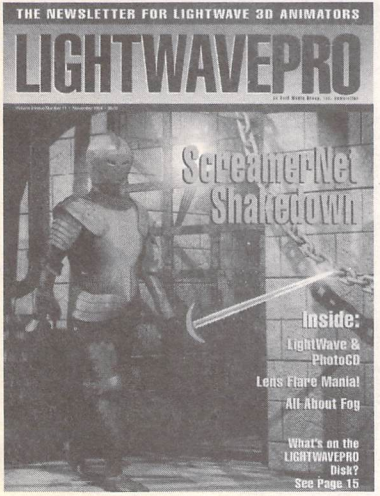
Flat	4K	4096x3112
	2K	2048x1556
Academy	4K	3656x2664
	2K	1828x1332
Anamorphic	4K	3656x3112
	2K	1828x1556

**Pixel aspect for Flat and Academy is 1, while Anamorphic pixel aspect is 2.*

Don't forget, since film resolution shows higher detail, it is important to build your CGI objects in a much higher resolution than you would for television. Otherwise, animation for a film finish is not much different from video resolution, regardless of what some so-called "professionals" think. With a bit more attention and care, any accomplished "video animator" can create just as accomplished film effects.

LWP

John F.K. Parenteau is a vice president and general manager of Amblin Imaging, and now, thanks to his editor, knows a bit more about the relation of video pixel and frame aspects to the world of film.



TO ORDER WITH YOUR VISA OR MASTERCARD CALL TOLL FREE!

1(800) 322-2843

LIGHTWAVEPRO

THE NEWSLETTER FOR LIGHTWAVE 3D® ANIMATORS

Make a Smart Investment

Subscribe Today!

Canadian/Mexico: Add $US12
Overseas: Add $US36
Allow 6-8 weeks for delivery.

Make checks payable to LIGHTWAVEPRO.
Prepayment required on all overseas orders.

YES! Enter my one-year (12 issues) subscription to LIGHTWAVEPRO at the Special Introductory Rate of only $48—that's 50% off the cover price!

Name_____

Address_____ Apt. #_____

City_____ State _____ Zip _____

☐ Payment Enclosed ☐ Bill Me

BFRLW

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

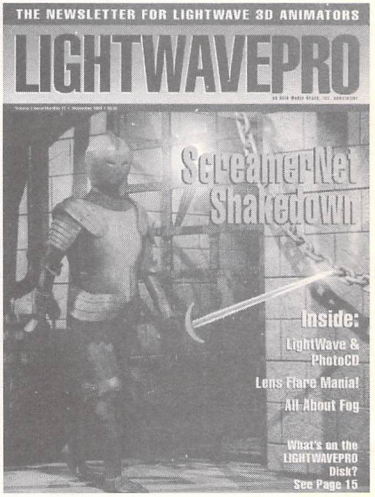
BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 2263 SUNNYVALE,CA

POSTAGE WILL BE PAID BY THE ADDRESSEE

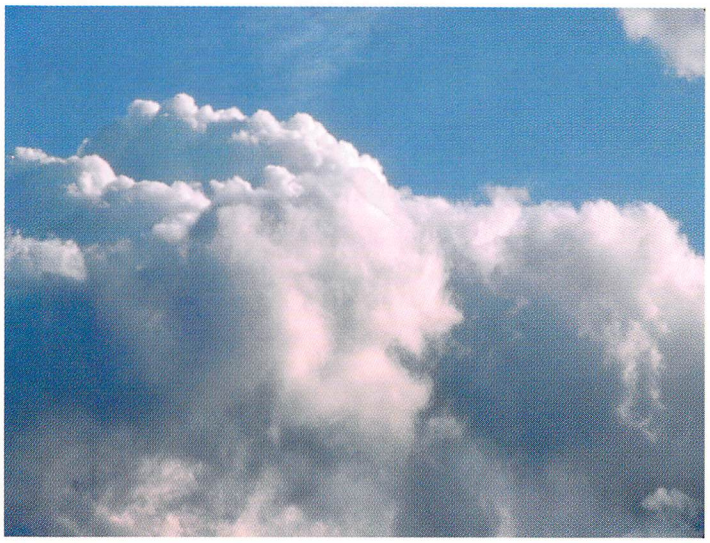
LIGHTWAVEPRO

273 North Mathilda Avenue
Sunnyvale, CA 94086-9313



TO ORDER WITH YOUR VISA OR MASTERCARD CALL TOLL FREE!

1(800) 322-2843



▲

Getting Cloudy

The original clouds brought in via PhotoCD. See "Cut It Out," page 12.

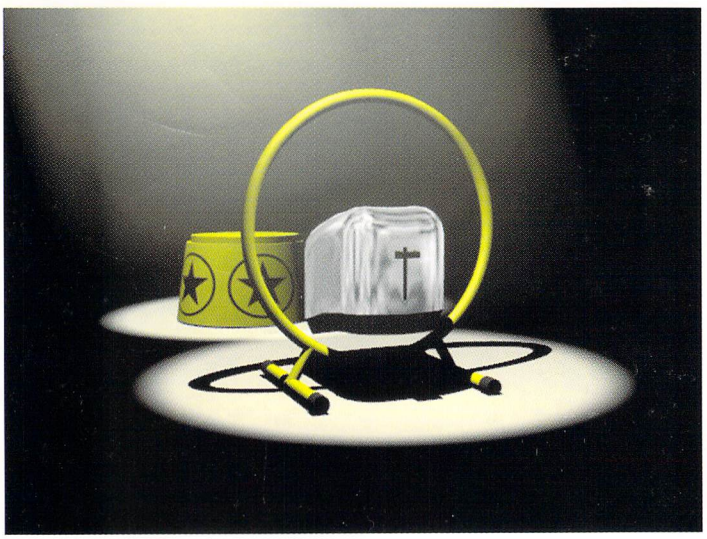
Copyright 1995 Dan Ablan

▼

Making Cloud 9

The LightWave scene, which has image-mapped polygons of the same cloud image. Transparency maps are placed on the polygons for soft edges, and a Front Projection Image Map is used to bring the logo out from behind the clouds. See "Cut It Out," page 12.

Copyright 1995 Dan Ablan



▲

Circus Toaster

Once you get the hang of how Bones work, you'll have objects jumping through hoops in no time! See "Taking That First Step," page 10.

Copyright 1995 Arnie Boedecker

▼

ToasterWalk

"Uh, Excuse me, is your toaster running? It is? Well, you better catch it!" Bones give life to inanimate objects. See "Taking That First Step," page 10.

Copyright 1995 Arnie Boedecker

See What Happens When Everyone Puts Their Heads Together.

Raptor 3 is twice as powerful as anything you've seen before.

Sink your teeth into Raptor 3 with Alpha 21164 microprocessor, the most powerful Windows NT workstation and CPU combination in the world. With Alpha 21164, Raptor 3 is now twice as fast as any Raptor to date. For Lightwave users, that means this one machine can take bigger bites than ever of your 3D rendering tasks. What's more, Raptor 3 is processor independent. In other words, you switch CPUs, not computers, when you upgrade. Or start out with another RISC microprocessor like MIPS R4600 or R4700 and upgrade later. Turn your attention to the motherboard and you'll find it delivers unparalleled I/O capability with 4 PCI slots, 2 ISA slots and twin SCSI ports, making it easy to use Windows NT's disk striping feature that can double hard disk performance. And 8 SIMM sockets provide capacity for up to one gigabyte of main memory. Best of all, Raptor 3 is from DeskStation Technology. The company that helped give birth to the 3D animation rendering industry and continues to reinvent it with an insatiable appetite for research, development and innovation. The result? For 3D rendering, Raptor 3 with Alpha 21164 is head and shoulders above anything else.

Raptor™ 3

For more on Raptor 3 at twice the speed, call (800) 793-3375

DeskStation Technology