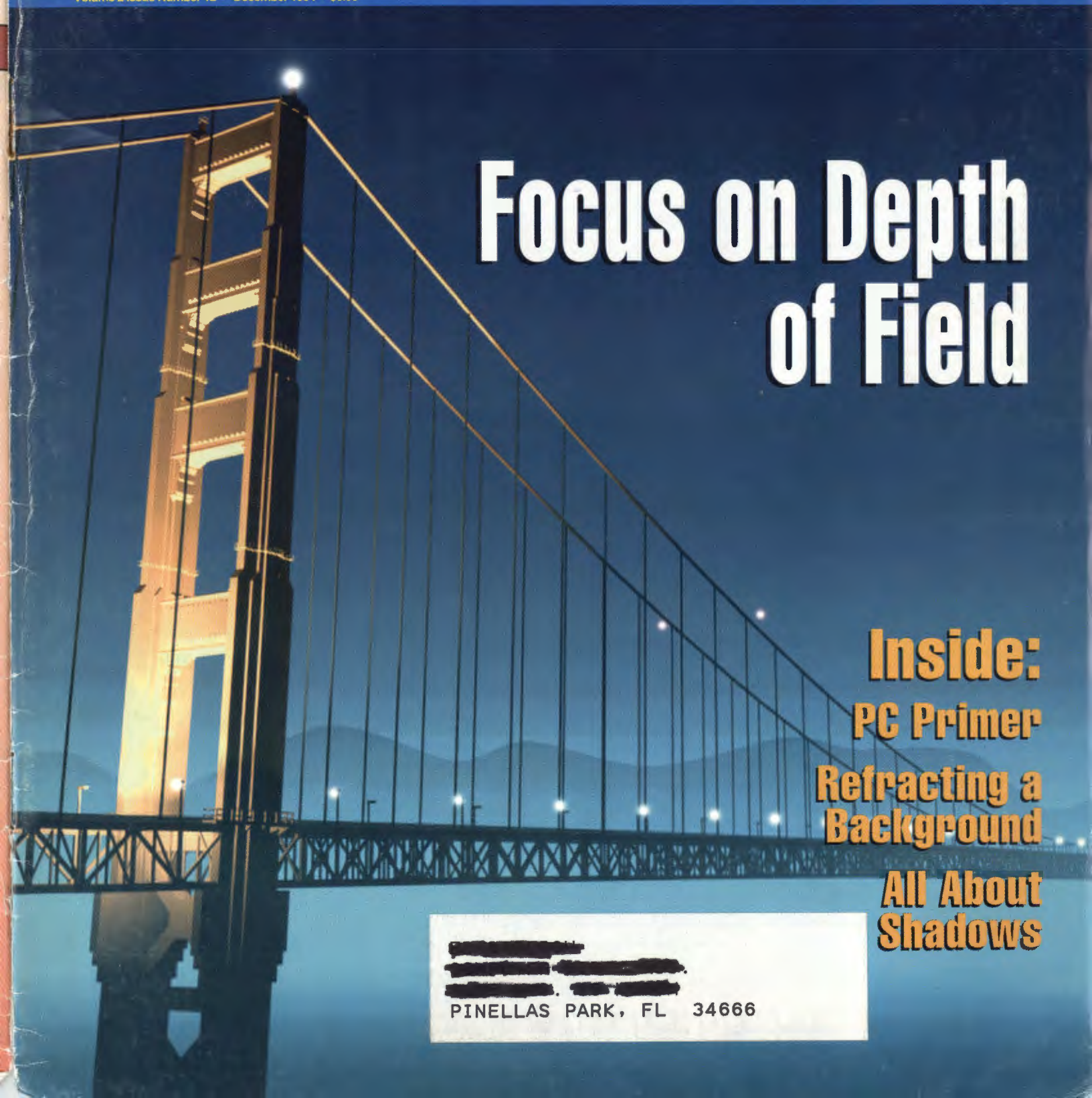
THE NEWSLETTER FOR LIGHTWAVE 3D ANIMATORS

LIGHTWAVEPRO

an Avid Media Group, Inc. newsletter

Volume 2 Issue Number 12 • December 1994 • $8.00



Focus on Depth of Field

Inside:

PC Primer

Refracting a Background

All About Shadows

The Abyss

Compositing refractive objects over live action is effective and surprisingly simple.

Copyright 1994 Colin Cunningham

Killer Blob

Holographic sidekick Diana (Andrea Roth) struggles to free herself from a syrupy demise in *RoboCop: The Series,* episode 16, "Sisters in Crime."

©1994 Skyvision Entertainment

Lifeguard Chair

An effective use of a fake shadow to create the illusion of a sunny day at the beach.

Copyright 1994 Mark Thompson.

EDITOR'S MESSAGE

by John Gross

Sometimes things don't go exactly as planned. Take last month's Editor's Message as an example. I had written an incredibly witty, uplifting and extremely important message for all of you, but when my issue arrived, what I read was the same message I had written a few months ago.

While it could be argued (but not by many) that it was such a great column that it deserved to be reprinted, the truth is that it was simply a mistake. Sorry about that. Don't worry, though—there wasn't any earth-shattering news (as far as you know).

Seriously, one topic mentioned was the fact that there has been a bug fix patch for Modeler made available on the NewTek BBS (913-271-9299), CompuServe and probably some other on-line services as well. This patch fixes problems with Modeler giving you an error message stating "Internal Buffers too small" when attempting certain procedures. There are also some other bug fixes and the patch is free. We also included the patch on last month's *LWPRO* disk.

NewTek's BBS? If you didn't know about this, you should give it a call sometime. There are plenty of great things posted there, including tutorials, questions/answers and information. Check it out.

For you Internet surfers, there are a couple of places to get LightWave information. There is a LightWave mailing list and a LightWave newsgroup. To subscribe to the mailing list, send a message to listserv@netcom.com. In the body of the message say "subscribe lightwave-l your address" (no quotation marks). In a few days you will start receiving messages from the list. The LightWave newsgroup is named comp.graphics.packages.lightwave.

Both the list and the newsgroup are great sources of information and are frequented by LightWave animators of all types. From *LWPRO*, both Mark Thompson and myself are frequent participants. Also, you can expect the occasional visit from Stuart Ferguson or Allen Hastings.

see Editor's Message, page 13

TABLE OF CONTENTS

LIGHTWAVEPRO

AN AVID Media Group, Inc. NEWSLETTER

Editor ..... John Gross
Managing Editor ..... Jim Plant
Editorial Coordinator ..... Douglas Carey
Art Director ..... Helga Nahapetian Taylor
Art/Production Coordinator ..... Kristin Fladager
Production ..... Sergio "Berimbau" Miller
Associate Editors ..... Joan Burke, Corey Cohen
Circulation ..... Debra Goldsworthy, Tracy Ann-Sparks
Contributing Writers ..... Colin Cunningham
..... Taylor Kurosaki
..... Mojo
..... John F.K. Parenteau
..... Mark Thompson
..... Ernie Wright
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)

Are you interested in writing for LIGHTWAVEPRO or submitting images? If so, contact us at our offices or electronically.
Avid Media Group, Inc. its employees or freelancers are not responsible for any injury or property damage resulting from the application of any information in LIGHTWAVEPRO.

LIGHTWAVEPRO (Vol. 2, No. 12) (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 $72 (U.S.); Canada/Mexico, $84 (U.S.); Overseas, $108 (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: This month's cover image was created by LightWave artist Stoney Runyon. It is from an upcoming animation entitled ***Stoney Left His Heart in San Francisco*** *and features over 10 minutes of LightWave animation. The Golden Gate Bridge object is built to exact specifications and took one week to model and texture (it even includes cracks and potholes in the road!).*

Depth of Field

Improving Your Image Through Imperfection

by Ernie Wright

Anyone who is routinely forced to watch home videos knows that objects photographed with real cameras and lenses don't have to be in focus. This lens effect, called depth of field, is ordinarily absent from CGI, and although home movies might make us wish otherwise, the unnaturally perfect focus of objects at every distance from the computer camera is a sometimes undesirable CGI signature that limits the artistry and realism of our images.

LightWave's depth of field controls, at the bottom of the **Camera** panel (Figure 1), allow us to simulate the focal properties of lenses. But before we can use these controls effectively, we have to overcome a minor problem: Unlike real cameras, LightWave doesn't have a viewfinder. It's difficult to compose the image, deciding which elements will be in or out of focus, without doing some math.

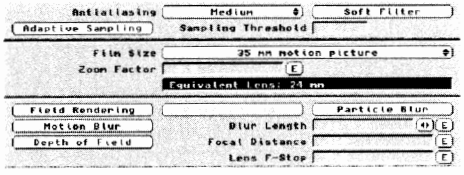


Figure 1: Controls affecting depth of field are in the bottom half of the camera panel.

Near and Far

Figure 2 is a diagram of the geometry involved in doing depth of field calculations. The labels "near" and "far" refer to the limits of the range within which objects appear to be in focus. The question is, assuming you already have a perfect focus distance in mind, how do you figure out where near and far are?

There are three simple formulas that answer this question. Before using them, we need to know where to find the values we'll be plugging in, and, as you'll see, we need to be careful about the units we use.

Three of the values are on the Camera panel:

F Equivalent lens (focal length)

f Lens f-Stop

S Focal Distance

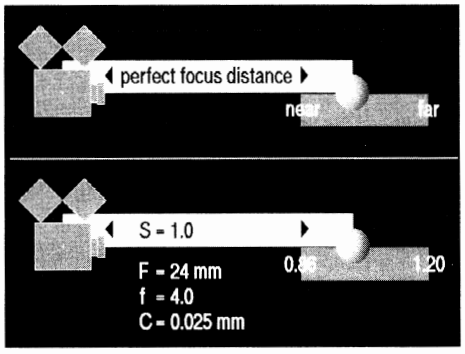


Figure 2: Objects within a range of distances from the camera appear to be in focus.

The focal length of a lens is the distance between the lens and the film plane. The Camera panel's **Equivalent Lens** readout is derived from the **Zoom Factor** and **Film Size** settings, so changing either of these will also change the focal length used to create depth of field.

Lens f-Stop sets the diameter of the lens aperture. In a real camera, the aperture is a circular window near the lens that can be widened or narrowed to control the amount of light that hits the film. The aperture diameter is usually expressed as a fraction of the focal length: an f-stop of f/4, which is the same as a setting of 4.0 in the Lens f-Stop control, indicates an aperture diameter that's 1/4 of the lens's focal length. (This can be a source of confusion: higher f-stop numbers correspond to smaller apertures because the number is the denominator in the fraction.) The aperture control on a real camera affects both the brightness and sharpness of the image, but LightWave uses the Lens f-Stop setting only in the context of depth of field, where it affects sharpness.

Focal Distance is just the distance from the camera to the point of perfect focus. This distance is measured along a line through the center of the lens and pointing in the same direction as the camera.

The fourth value we need is the diameter of the largest spot that still looks like a point on the image:

C circle of confusion diameter

This isn't a setting in LightWave. It's a number you'll use to define the fuzziness limit between in focus and out of focus (more on circles of confusion later).

Is your pencil sharpened? Let's begin. With **Zoom Factor** set to 3.2 and film size set to **35 mm motion picture** (the defaults), the focal length (Equivalent Lens) is 24 mm. The default Lens F-Stop is 4.0 and the default Focal Distance is 1.0 meter. For now, we'll use a fairly common value for circle of confusion diameter. So we have:

F = 24 mm
f = 4.0
S = 1.0 m
C = 0.025 mm

We then calculate H, the hyperfocal distance of the lens, or the distance at which "far" is infinity:

H = (F x F) / (f x C)
= 576 mm / 0.1 mm
= 5760 mm
= 5.76 meters

And now we're ready to find near and far:

near = (H x S) / (H + (S - F))
= (5.76 x 1.0) / (5.76 + (1.0 - 0.024))
= about 0.86 meters

far = (H x S) / (H - (S - F))
= (5.76 x 1.0) / (5.76 - (1.0 - 0.024))
= about 1.20 meters

Notice how units are handled here. H is calculated in millimeters, then converted to meters. When we need F again for near and far, we also convert that to meters (24 mm = 0.024 m). It's important to realize that even though it's convenient to think of LightWave's coordinate space as being measured in generic "units," the calculation we just performed requires that we think in meters. You might have a humanoid model that's five units tall, and you might think of the units as feet, but for purposes of depth of field calculation, the model is considered five meters tall. This is another good reason to make your models realistic sizes in metric units.

Focal Distance

So far we've pretended that you know what Focal Distance you'd like to use, but in fact you might not know. Let's assume now that you'd like a particular object to be in focus and you don't know how far away it is from the camera. The LightWave manual suggests you use the Layout grid to estimate the distance, but this advice is a little vague.

You can always use the Pythagorean theorem to calculate the exact distance (d), plugging in the position coordinates for the camera and the object:

dx = x2 - x1
dy = y2 - y1
dz = z2 - z1
d = square root of ((dx * dx) + (dy * dy) + (dz * dz))

If the object wasn't constructed near the origin in Modeler, you'll have to add the offset to the position reported by Layout before you do the distance calculation, because the coordinate location of objects in layout always refers to the origin of the object.

But what if you don't have the patience to find the square root of the sum of the squared differences of each coordinate? (Who does?)

Try this. Arrange the grid so that both the object and the camera are visible in the XZ View. Estimate dx, dy and dz by counting grid squares along X and along Z and then eyeballing dy in a different View, and write them down (or rearrange them in your head) in the following order: biggest, middle, smallest. A fair estimate of the camera to object distance is:

biggest + middle/4 + smallest/4

This shortcut (from Jack Ritter's paper in *Graphics Gems*, Academic Press, 1990) will at least prepare you to do test renders.

Circles of Confusion

For a better understanding of circles of confusion, take a look at Figure 3. When a point is at the Focal Distance, the light reflected (or emitted) from it focuses to a point on the image plane (the film). As the point is moved closer to or further from the lens, the image of the point becomes a circle, called a circle of confusion, whose diameter depends on how far from the Focal Distance the point has been moved.

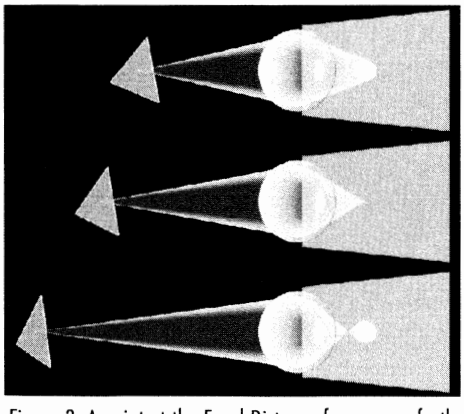


Figure 3: A point at the Focal Distance forms a perfectly sharp image. Moving it closer or further blurs its image into a circle.

Both the image medium (whatever it might be) and the human eye have limited spatial resolution — if a circle in an image is small enough, it might as well be a point. This is what is meant by a range in which things "appear to be" in focus. The circle of confusion diameter C that we used to calculate near and far is just the upper limit on the size of the circles. Over the limit, we say that objects are out of focus. Under it, they're in focus.

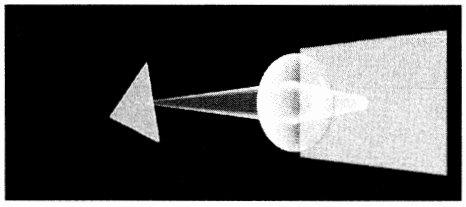


Figure 4: The effect of a smaller aperture (higher Lens f-Stop).

The actual value we choose to use for C is obviously somewhat subjective, but it's related to the resolution of the image medium. The value 0.025mm often used for 35mm motion picture film is about 1/1500 of the width of the image.

For LightWave rendering, you won't notice circles of confusion smaller than a pixel of course. A single pixel in a Medium-Res Overscan image is about 1/750 of the image width, or twice 1/1500, so with Film Type set to 35mm motion picture, you should probably use a value for C no smaller than 0.050mm. In general, let C be a fudge factor that makes the equations for near and far give you what you think are the right answers, based on what you've actually seen LightWave produce for various depth of field settings.

Antialiasing

You may have noticed that the depth of field controls aren't available when you first enter the **Camera** panel. This is because depth of field requires Antialiasing to be set at **Medium** or **High**. LightWave's depth of field simulation uses the same supersampling algorithm that's used to antialias images and create motion blur. It's important to realize that this puts (reasonable) limits on what you can do.

Figure 5 is an image of a 1-meter box rendered with Depth of Field. The camera is 2.5 meters away, and antialiasing is set to Medium. All of the Depth of Field settings were left at their defaults except Focal Distance, which was made a ridiculously small 0.05 meters. As you can see, the box is so far out of focus that it can no longer be rendered coherently. Keep this in mind when composing scenes that will use Depth of Field.

If you're comfortable with the math we've used so far, you can easily use the near and far formulas to tell you where "too near" and "too far" are. Increase the value of C so that it corresponds to a circle of confusion that's at the size limit of what LightWave appears able to handle. A few test images with a 1-meter box will help you converge on the right numbers.

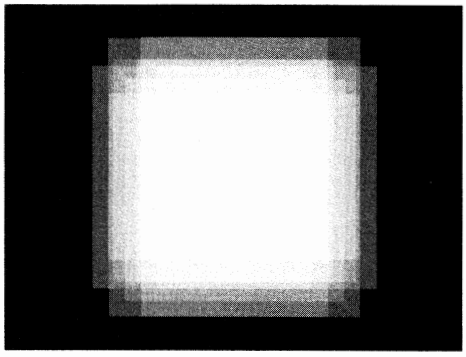


Figure 5: A box that's so out of focus it can't be rendered properly.

When to Use It

Depth of Field isn't something you're likely to use every day. As a practical matter, the Medium to High antialiasing it requires has a significant impact on rendering time, but more importantly, DOF is a powerful tool that you'll want to use sparingly, when a shallow depth of field (a small near-to-far distance) will enhance the composition of an image.

Shallow depth of field gives an intimate feeling to images containing small or nearby subjects because it doesn't allow the background to intrude on the viewer's attention. My wife uses a 50mm macro lens on her SLR to photograph Christmas ornaments (on the tree) from a few inches away. The only object in focus is the ornament itself. The surrounding elements—needles, small branches and a few lights—are blurred into a soft backdrop that halos the ornament and provides context without drawing attention away from the subject.

Changing focal distance is a way to gently push or pull the viewer's attention between the foreground and the background. You'll see this technique used fairly often in soap operas, where it complements the viewers' intimate relationships with the characters and, not incidentally, offers a little visual variety when the "action" is limited to two people in a room talking.

Shallow depth of field also enhances the 3D illusion. The naked eye doesn't ordinarily rely much on focal distance as a depth cue, but we easily recognize the effect in photographs. The sharply focused, high-contrast subject appears to float well in front of the more diffuse background. Just remember that this difference in focus seems most natural for subjects that are small or quite close to the camera.

LWP

For the past four years, Ernie Wright has been a programmer doing scientific visualization for several defense-related agencies (he won't say which ones). He also wrote the HAM routines for LightWave 3.5. He works from his home in Maryland while watching his 2-year-old daughter.

Digital Cinematography

by John F.K. Parenteau

I am, of course, prejudiced, but I feel the most important job on a production set is that of the cinematographer. Yes, many other positions such as director, actors and even production designers play an equally important role in bringing a script to life, but the camera person is the last bastion of truth before all the work of those other departments is recorded on film stock. It is the director of photography (another term for cinematographer) who designs, with the director, the method each shot is to be photographed, how it will be lit and in what method the camera will record it. And this brings us to the point of this month's column. I hear two complaints from young animators, and young camera people, for that matter: "Lighting seems so simple, but I can never make it look right" and "How do you move the camera?" The former we have dealt with and will continue to deal with in the future. The latter is a complex question that requires immediate attention.

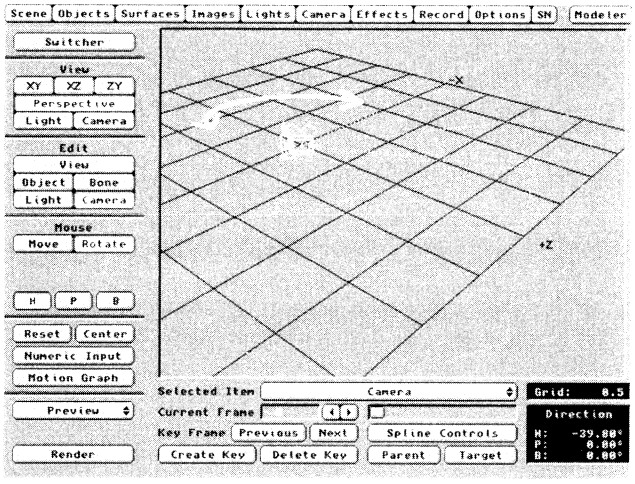


Figure 1

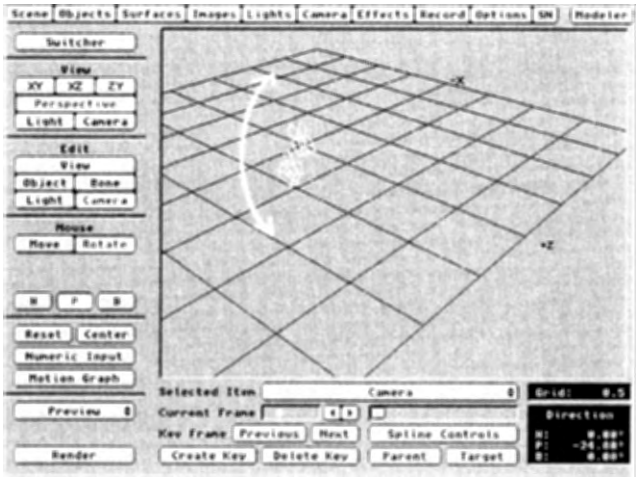


Figure 2

A Work of Art

Moving the camera is actually an art form. Watching some of the masters' greatest works, and how each shot seems to elicit the most appropriate response, makes you wonder if it was all planned. Believe me, it was. The best cinematographers pay as much attention to how the camera moves, or doesn't move, as they do to how it is lit. The process not only requires the knowledge of the rules of motion, but also an emotional understanding of what effect you are achieving by moving the frame.

Think of the camera as another actor. Just as an actor on screen chooses a method of approaching a performance, a good cameraperson makes a similar choice. An actor prepares for a scene by carefully studying the dialogue, and the situation in which the character has been placed. Picture the following scene from *The Shining*:

Jack Nicholson smashes a door with an ax as he chases his family through a deserted lodge. He is confused, angry and seemingly possessed as he hunts his wife and children. As the door smashes apart from the ax, Jack sticks his head through the door and exclaims, "Here's Johnny!"

Nicholson chose to deliver the line with a twisted smile rather than a more serious "I'm going to kill you" face. By contrasting his intent with a smile, the moment is even more disturbing. Choices such as these are made every day. The best actors make the best choices and thus achieve the most convincing performances. As a cameraperson, it is important to approach each shot with the same mindset. The camera is an actor as well, and as you choose how to reveal the scene through the lens, you are choosing what the audience will or won't see. Most horror films have a scene as follows:

Our hero (or heroine) has stupidly decided to go back in the house, even after he has found all his friends dead by mysterious methods. The camera starts in close-up, probably a handheld shot, as we move with our character through the hallways. The scene cuts to an over-the-shoulder shot (behind the actor, looking over a shoulder, called OTS), as we now follow the same character. Because of the strange, voyeuristic feel of the over-the-shoulder shot, we feel like something will come up behind him at any moment. Finally, as our OTS shot has become closer and closer, the character screams and turns to camera. We cut around to his point of view (the view from the actor's position, as if looking through his eyes) and see nothing. Cutting back to the OTS, the actor and we, the audience relax and he turns back to the hallway... and a demonic creature is waiting to pounce!

As you can imagine, the camera positions and movement had as much to do with creating a spooky mood as did the performance of our ridiculously stupid character. But this is an obvious example. Sometimes the most gross and obvious moves are the most simple to understand. Consider, however, the subtle pan, a short dolly move or even no motion at all. Each choice expresses its own emotion, and carries its own performance.

All the Right Moves

First, let's consider the basic moves possible with a camera, both in CGI and live action. The PAN is a pivoting motion left to right or right to left (Figure 1). The TILT is a pivoting motion up and down (Figure 2). A camera TRUCK or DOLLY is a physical movement of the camera in any direction on the X or Z axes (Figure 3). A BOOM is a physical movement of the camera in an up or down motion (Figure 4). These four basic motions comprise the simplistic forms of all camera movement. To understand how to manipulate these motions requires some understanding of basic framing.

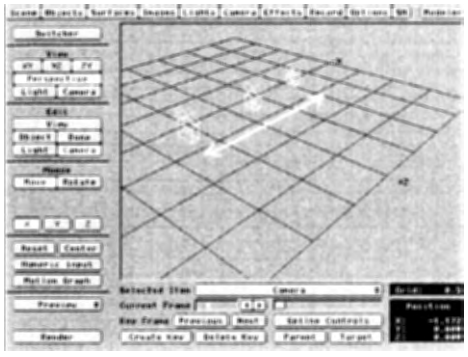


Figure 3

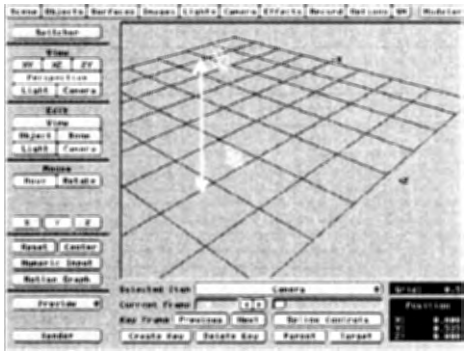


Figure 4

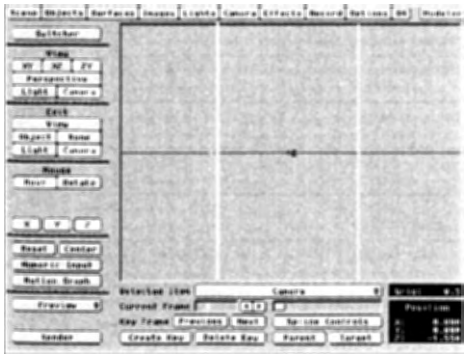


Figure 5

A Perfect Frame of Mind

When you photograph your friends or family with a still camera, whether you are conscious of it or not, you are considering the rules of framing. Though you may not realize or even understand these rules, natural

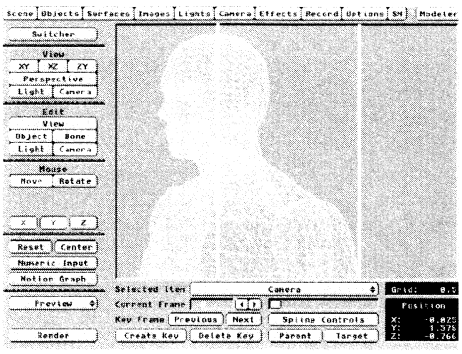


Figure 6

skill can come into play. As you place your friend on the edge of a cliff overlooking a beautiful panorama, you automatically frame your friend to the side to see both the view and the person. To consciously understand why this is the most pleasing method of framing, it is important to understand the theory of thirds.

Consider the visible frame through the eyepiece, or on the LightWave layout screen. Break this frame into even thirds, vertically across the frame (Figure 5). Placing the object so that one of the borders between the thirds runs through the object creates the most desirable framing. If the object were placed in one of the side thirds, the frame would be unbalanced and would feel wrong. If the object were placed in the center, the frame would feel much more natural, but very uninteresting.

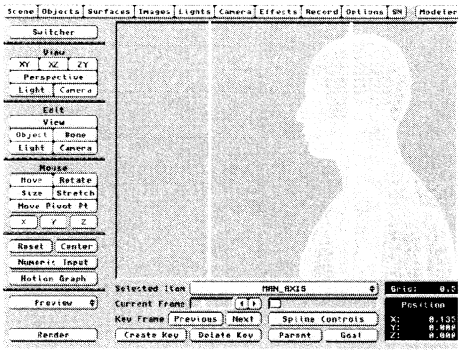


Figure 7

Extending this theory into the photography of people, it is important to consider "nose room." In a close-up of a person, looking off frame left, it is necessary to allow the subject room to look off. If the subject were framed center, or the nose was placed close to the left edge of the frame, the balance of the frame would be off, and the character would have no room to look (Figure 6). "Nose room" gives the character a place to look. Consequently, if the character is looking directly at camera, it is appropriate to frame him or her center. As the angle of the look moves further and further away from the camera, more room should be allowed as lead space. If our character is looking directly off screen, showing us a profile, we would tend to place him or her close to the edge of the frame (Figure 7). In addition to breaking the frame in thirds vertically, it is also important to consider horizontal separation. The same rules apply here, and must be considered as well.

As you transition, you're thinking from a still frame to motion, be aware of live and dead spaces in your shot. By limiting the action to a small area of the frame, the remaining areas may seem dead and useless. Try to utilize all areas in some way to prevent the audience from losing interest.

Remember that action does not have to take place from the sides or top and bottom only. Use the angles of the frame to break up these dead spaces. For example, a spaceship entering the frame from the left and exiting to the right may be appropriate in telling your story, but it may also be the most boring way to tell it. To add some excitement, have the ship enter from the top left corner and exit the bottom right. This utilizes all thirds of the frame and makes for a much more interesting angle.

With an understanding of the theory of thirds and the basic camera moves, it is now time to study more complex motions and the reasons why you might want to utilize them.

You're the Director

A crane shot in live action is a very popular move. Not only is it a great way to establish a scene, but it's also fun to ride. A crane is the combination of a boom and a tilt, with a little dolly often thrown in for good measure. Though it is a simple motion of dropping the camera from some height to the ground, there are many considerations. Though the boom may be dramatic, it will be relatively invisible unless some foreground elements are utilized.

Let's create a large monolithic building in an open field. The building is 1,000 feet high and a mile away from the camera. If we were to start up high and boom down to the ground we would see little or no effect. By placing a tree in the foreground and booming down past the branches, we can now see the effect of the crane. Add a few cars on the road approaching the building, or perhaps other, smaller buildings on the road approaching the camera. Now the parallax will be increased by adding additional objects at increasing distances from the camera. Remember, a crane is not effective unless you are craning past something.

Now our scene is of two cars chasing each other down the road. Our camera position is on the side of the road and the cars are approaching quickly. Utilizing a "Whip Pan" in this situation would be most effective.

As the first car approaches, the camera pans quickly with it until we see the car receding away from us. Suddenly the camera whips back down the road to catch the other car in hot pursuit, and again the camera pans with it as the car continues down the road. This shot actually uses a whip pan in two ways. First, as the car moves past the camera, it is necessary

see Digital Cinematography, page 18

Interactive Refraction

by Colin Cunningham

Computer-generated imagery has come a long way in the past few years, from the watery pseudopod in *The Abyss* to the Tex Avery-inspired nuttiness in *The Mask*.

The one element these films share is the seamless integration of computer graphics (CG) and live action. CG compositing is, in itself, an art form requiring a good eye and loads of patience.

With LightWave, animators have at their disposal an arsenal of tools to make this task less daunting. Shows like *seaQuest DSV* and *Babylon 5* have often made good use of LightWave's compositing tools to blend CG elements into a real environment, but generally remain in the 3D realm. On *RoboCop: The Series*, however, marrying LightWave elements with live action was a daily task involving 80 cans of Coca-Cola and a ton of raw meat (I won't go into details).

You've heard it all before, I'm sure: "Match-lighting, THIS..." and "Front projection mapping, THAT..." There are many articles available on CG compositing and you're probably all experts in the field by now, so before you think I've gone a bit screwy, read on.

The truth is, basic compositing is pretty much straightforward: use your video or film footage as a background image sequence and render your objects directly on top. Simple, right? Of course you'll probably have to match the lighting or cut a matte for your CG elements to move behind, but in the hands of a talented artist, there shouldn't be much to worry about aside from being time-consuming.

LightWave made it easy for the effects team at *RoboCop: The Series* to add 3D choppers and sometimes people into scenes convincingly; even my mom couldn't tell the difference (maybe she was just being nice). We did, however, run into problems trying to composite refractive elements over live action. Let me explain.

Know Your Physics

In episode No. 16, "Sisters in Crime," RoboCop battles evil and injustice within Delta City's corporate sector. More specifically, this episode called for Robo's holographic sidekick, Diana (played by Andrea Roth), to become engulfed in a syrupy, amber goop that would eventually harden into a crystalline tomb. Not only did the blob have to refract its surroundings, it also had to bulge and twist convincingly as Diana struggled to free herself from its sticky grasp.

Now some of you already know that refracting a background image/sequence is not as easy as it sounds, and I'll be the first to agree with you: it's not. For the other 90 percent who are probably scratching their heads right about now, here's a little experiment you may want to try when you have some free time on your hands.

Upon entering Layout, load your favorite framestore and select it as a background image through the Effects panel. Next, load a simple object like a sphere and position it centrally within the camera's view so we get a nice look at it. For our experiment, we'll need to give this object some refractive qualities; the following is a nice glass surface I've used from time to time:

GLASS	
Color	225,225,225
Specularity	95%
Glossiness	High
Reflectivity	15%
Transparency	98%
Color Filter	ON
Refractive Index	1.51

It would be a good idea to save this surface, as you'll need it later. The default lighting will do just fine for this scene, but turn on **Trace Refraction** (Camera panel) and hit **Render**. Notice anything interesting? You shouldn't have to look very closely to see that our glass sphere isn't refracting the background. The background may be tinted slightly through the sphere, but the image hasn't been distorted in any way.

The reason for this is that, simply put, the background doesn't exist. Sure, you can see it, but it isn't really there (now I've completely lost it, right?). No matter how many objects you place in your scene or how far you position them from the camera, they will always appear in front of the background image. This is because LightWave places the background layer at an infinite distance from the camera so that objects can simply be rendered on top in one step without having to composite.

The one thing I do remember from Mr. McSharry's physics class (aside from the fact that some things don't react well to open flame) is that calculations for refraction rely on the precise locations of objects, and since there is no measurable distance for our background image, LightWave does what it should and renders the sphere without any refractive qualities. It would appear as if we're in a bit of a rut, but then things are not always as they appear.

Building a Virtual Screen

In order to make the sphere refract our background, we must give the image physical qualities that LightWave can work with. The simplest way of doing this is to map the image onto a 3D object. Not any old polygon will do, mind you—we need a polygon perfectly sized to fit the camera's viewing area. This way, our rendered image will be indistinguishable from the original framestore.

- Enter Modeler and select the Box button in the Objects menu. Select Numeric.

The aspect ratio of a Toaster framestore is 1.346. This means the screen is 1.346 units wide by 1 unit high; the units aren't really important, only the ratio. Knowing this:

- Make a box that is 2.692 meters wide (1.346 x 2) by two meters high (it's important for this tutorial that we use meters) by entering the following values in the Box Numeric requester:

  Low
  X= -1.346
  Y= -1.0
  Z= 0.0

  High
  X= 1.346
  Y= 1.0
  Z= 0.0

Leave the number of segments set to 1 for X, Y and Z. Since we'll only be seeing this polygon from the front, it doesn't need depth, which is why there is 0 for the Z values.

- Press **OK** and then **Make** or return to make the box.
- With our "screen" hot off the assembly line, it

would be a good idea to name its surface to avoid confusion later on. To do this, hit q for **Surface** (**Polygon** menu) and enter "SCREEN". Press return to change the surface name.

- Save the object as "Screen.obj".

While we're in Modeler, create a simple cube using the Box tool again. Dimensions aren't important right now (we'll be re-sizing it later); just make sure to press f to flip the polygons so that they face inward. Next, change the surface name to "Box" as mentioned above and save the object. Don't worry if you're confused—I'll explain what this is all about later. Now go back to the Layout. Pretty simple so far, wouldn't you say?

STEP-BY-STEP

- Build the screen & Box
- Name the surfaces and save each one
- Position the screen in Layout
- Map background image onto screen
- Position objects and lights
- Position box around all objects
- Map image onto sides of box
- RENDER AWAY

Positioning our Screen

- Load the screen object just created.

This next step is crucial. If the screen is not positioned exactly right, our background image may appear slightly larger or smaller than it should be. It's doubtful that even those with a keen eye will ever notice that your background image is sized incorrectly, mainly because they don't have the original image for comparison. Because of this, you have some room for error, but for me, this wasn't an option, as I had to precisely rotoscope my blob to the actress' movements.

Positioning the screen can be done two ways: the hard way and the easy way. The hard way involves rendering your screen object with a framestore mapped onto it and comparing the render to the original framestore. By switching between the two very quickly, you'll notice any discrepancies and reposition your screen accordingly. Although somewhat accurate, this technique is nothing more than a time waster, and the last thing you have to waste in this industry is time.

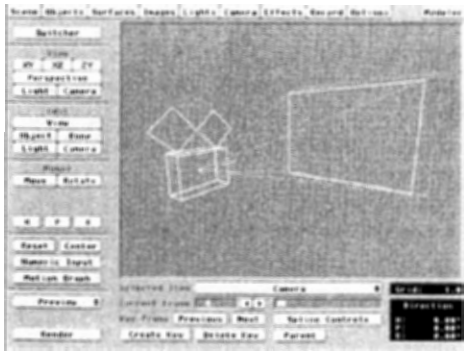


Figure 1: By following a few simple steps, the screen object can be positioned perfectly within the camera's view.

I bet you're just itching to know the easy way, so I won't keep you in suspense.

- Load the screen object into layout and position the camera at 0,0,0.
- Next, enter the **Camera** menu and note the camera **Zoom Factor** (the default is 3.2). Enter that value as the screen object's Z position and make a keyframe for the screen by hitting the return key twice. Presto! Our screen is now positioned so that it will precisely fit the boundaries of our camera view (Figure 1).

You can run the comparison test I mentioned above but I'll save you time and tell you not to bother, it's 100 percent accurate. It would be a really good idea to save this scene now so it can be used as a template for any future projects.

Now that the screen is in the right place, it's time to map something onto it (you wouldn't want to refract default gray, would you?).

- Load your favorite framestore and enter the **Surfaces** panel.
- Select the SCREEN surface that was named earlier and select the **T** button next to Surface Color.
- Use the **Planar Image Map** texture to map the framestore onto the screen along the **Z Axis**. Don't forget to click on **Automatic Sizing** before returning to the Surfaces panel.
- Set the Luminosity to 100 percent and the Diffuse Level value to 0 percent so it won't be affected by any lights in the scene.

Now let's retry our original experiment to make

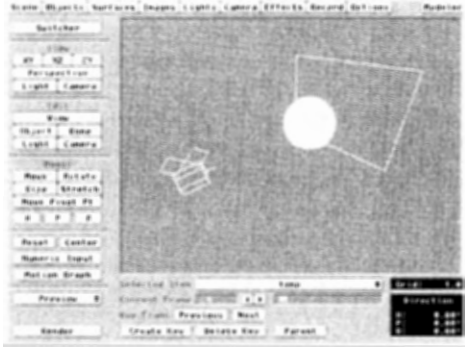


Figure 2: Because of the limited scene area, objects must be placed in front of the screen at all times.

sure we've done things right.

Getting Interactive

Load another sphere and give it the glass surface we created earlier (you did remember to save it, didn't you?). We can't just position the sphere anywhere within the camera's view as we did before; we must keep in mind that there is now a screen object within our virtual soundstage.

- Select the ZY (side) view and make sure that all objects are positioned in front of the screen object and not overlapping or behind it (Figure 2).

Because you now have a limited "scene area" in which to move the objects around, you may have to fake distance by scaling your objects down if they have

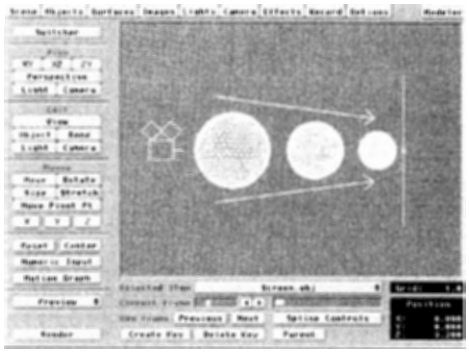


Figure 3: Scaling an object while moving it along the Z axis helps creates the illusion that it is moving a farther distance than it really is.

to move a great deal along the Z axis (Figure 3).

One more rule to remember when positioning refractive objects in relation to the screen: The closer the object is to the screen, the more discernible the background image will be. If you position your object closer to the camera, the background image will be distorted quite a bit; this may or may not be the effect you're after, so a judgment call is required.

With this in mind, place your refractive sphere within the scene area, make a key frame for it and hit **Render** once again (don't forget **Trace Refraction**).

If you've followed the steps correctly, you should now have a realistic glass sphere on your screen refracting the background image.

The Fine Art of Boxing

After fine-tuning this technique, sticking a refractive amber blob on our actress was a snap. We created a spline blob (this was pre-metaform) to match Diana's general form and bones were added to match the movements of her head and arms. As the bones twisted and stretched the blob, it looked as if Diana was really trying to escape from her CG prison; there were only a few minor things left to completely sell the effect.

Depending on how your refractive sphere was positioned in the last experiment, you may have noticed a black edge or black patches within the sphere. The same thing happened with our blob even though there were no black areas in our background footage. There is a simple explanation: the blob was bending light so much that it was beginning to refract the empty, black environment around it. This was a problem because the blob had to look like it was in a 3D environment, surrounded on all six sides by walls and furniture. We were simply faking 3D by placing a flat screen behind the blob; other than that, there was nothing else in the scene to refract except black. If you ever run across this problem in your refractive exploits, there is a simple but effective way around it.

Remember that box we made earlier?

- Load the box into your sphere scene and stretch it until it completely surrounds all the objects in your scene, including the camera and lights (Figure 4).
- Enter the **Surfaces** panel and find the BOX surface. Select T next to Surface Color and use a **Cubic**

continues on page 18

Dark Shadows

by Mark Thompson

You would probably be surprised by just how many algorithms and methods there are in computer graphics for generating shadows in 3D imagery.

LightWave offers a choice of two different methods, traced and shadow mapped. Each has its strengths and weaknesses, and one may be more appropriate than the other depending on the application.

Traced Shadows

Shadow algorithms are much like hidden surface removal algorithms, except the idea is to find out what surface is hidden from the light source. This is so because anything that is hidden from the light's view by foreground objects should be in shadow. Hence the similarity in methods.

Traced shadows in LightWave use a technique known as ray casting. It is essentially the same as ray tracing except that the light's view of the scene is being calculated, not the camera's. In addition, reflective and refractive surfaces do not bounce or bend the cast rays because they would not appreciably affect how the shadow is formed. However, like ray tracing, it is a slow brute force method that can have a major impact on render times.

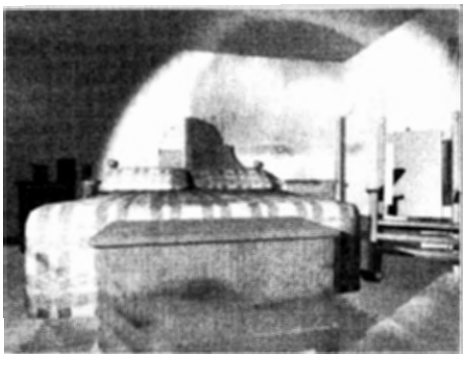


Figure 1

Slow as it is, there are several advantages to tracing shadows. First of all, although the characteristic hard edges may not always be the most photorealistic, traced shadows are typically more accurate than shadow mapping. Also, tracing works with all three LightWave light sources, unlike shadow mapping, which can only be used with spotlights. And finally, traced shadows respond to transparency and object dissolve properly. In fact, transparent surfaces that have the **Color Filter** option enabled will even cast colored shadows based on the color of the surface.

For example, you could create a stained glass window by color mapping an image onto a transparent window surface and turning on **Color Filter**. When a light is directed through that window with traced shadows enabled, the window will project the mapped image in the shadow (Figure 1).

In this picture, a scanned photo was mapped onto a transparent polygon and placed directly in front of a spotlight. The image is projected on the furniture and across the wall. Some software packages produce this effect through a feature called projection lights, but LightWave does not have this at the moment.

Riiight Face!

By now you are probably painfully aware of the difference between front-facing and back-facing polygons. It is frequently a source of confusion for anyone just starting out with LightWave. Basically, the side of a polygon that has the surface normal pointing outward is the side that is visible to the camera. Ironically, the opposite is true of light sources and traced shadows.

If you have a single-sided polygon with the visible side facing the camera, the only way to achieve a traced shadow is to illuminate it from the opposite side or select **Double Sided**. On the other hand, shadow maps ignore the direction of the polygon surface normal and cast a shadow from either side. And if you ever encounter slivers of light passing through your shadow-casting object, check it for non-planar polygons. An object containing non-planar polygons may visibly render just fine, but traced shadows can leak through it, exposing the imperfect geometry.

Shadow mapping

Shadow mapping was introduced in LightWave 3.0 to provide an alternative to the brute force ray-casting algorithm. Depending on how they are used, shadow maps can make a frame render many times faster than it had with traced shadows. Shadow mapping is a derivative of the z-buffer hidden surface removal algorithm. Imagine calculating a z-buffer from the light source's perspective rather than the camera's and then using that depth information to determine which areas are visible and which are not. Those that are not visible are in shadow.

The **Shadow Map Size** setting determines what resolution to use when calculating that "z-buffer." The number represents one side of a square view so the default of 512 generates a 512x512 shadow map. Each entry into that map is a 32-bit floating point number representing the distance from the light to a particular surface. This means that the memory consumed by a shadow map in bytes is equal to four times the square of the size you entered. For 512, that is 1MB and a size of 2048 would consume 16MB. As you can see, shadow maps can quickly use all your available RAM.

On the flip side, if you select a map size that is too small, the generated shadow can suffer from all sorts of artifacts and possibly even disappear.

Remember that the map is a view from the light's perspective and if the size is set so small that it can't adequately resolve the objects in view, the shadow will suffer. A common artifact is pixelated-looking shadows. Another is shadows that "jump" when animated. The **Shadow Fuzziness** setting helps smooth the shadow silhouette and can go a long way toward relieving edge pixelization. It has the added benefit of making the shadow more realistic.

Real-world shadows have an umbra and penumbra. The umbra is the dark inner area of a shadow while the penumbra is the light softer area around the edge. This effect is due to the fact that real light sources are bigger than just a point and are within a finite distance of shadow-casting objects. The larger the light source and the closer the distance to the object, the more the shadow will be dominated by the soft penumbra. A large **Shadow Fuzziness** setting with a small **Shadow Map Size** can create the appearance of soft penumbra dominant shadows.

Shadow maps have one other major advantage over traced shadows. They can cast shadows from single-point particles and two-point lines. Ray casting cannot do this because it must have a surface or volume for the cast ray to intersect with. One- and two-point particles essentially have no geometric volume or surface so they are invisible to the cast rays. But shadow mapping works by rendering a picture from the light's perspective, and since the particles render as visible enti-

ties, they can cast a shadow in the shadow map. If you remember my tutorial on creating a field of tall grass in the premier issue of *LWPRO*, you will note that I used shadow maps for this reason.

One of the most frequently asked questions about shadow maps is why do they only work with spotlights. Well, since the algorithm creates a rendered view of the scene from the light source, that view must be finite. Both distant lights and point lights have an infinite view, so a map image cannot be generated. However, spotlights are bounded by the **Spotlight Cone Angle** keeping their view of a scene confined to a finite area. It is conceivable that a feature could be added to bound the shadow map view of a distant light source to a finite volume, but there is not much difference between this and a spotlight placed very far away with a reduced **Shadow Map Angle**.

Shadow Map Options

One often overlooked method for keeping shadow map sizes small while not sacrificing shadow resolution is to use a **Shadow Map Angle** that is smaller than the **Spotlight Cone Angle**. This is particularly useful when lighting a large area, but only a small portion of that area actually needs shadows. Figure 2 has four examples of shadow mapping with the same light placement, a **Spotlight Cone Angle** of 30 and a **Shadow Map Size** of 200. The top two use a much smaller **Shadow Map Angle**, making the small map resolution more effective. You can, in effect, mimic an infinite light source this way by placing the spotlight far from the target, but limit the shadow angle to just the area of interest. Note that when using a **Shadow Map Angle** that is smaller than the **Spotlight Cone Angle**, the area the map will cover is represented by a square when using the **Light** View in Layout.

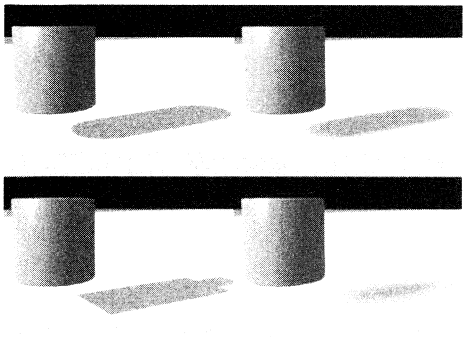


Figure 2

One of the big drawbacks of shadow maps is that they do not respond to transparency or object dissolve. However, in some cases, this can be used to your advantage. There are often applications where you need to cast a shadow, but you don't want to have a visible casting object. This is trivial with shadow maps. Simply set the object to cast shadows, dissolve it 100 percent, and shine a shadow-mapped spotlight on it. Voila, instant shadow but no object. This can be handy in special effects where you need to create a CGI shadow for an existing live action object.

While shadow maps ignore transparency, they do respond properly to clip maps. There is a minor glitch here that you should know about. When using clip maps, turn off antialiasing because it can interfere with proper clip mapping and shadow map operation.

Although shadow maps in LightWave have three adjustable parameters, there is a fourth hidden from the user. It is called "bias" and it is used to prevent a surface from incorrectly shadowing itself. In effect, it offsets the shadow from the source object. Some 3D packages allow the user control over this bias value but LightWave attempts to calculate the appropriate value internally. This is advantageous since most users would require a great deal of trial and error to select a value that works well. And according to Allen Hastings, "This has the added advantage that the bias value is customized for each pixel being rendered, based on its distance from the light, its angle to the light, the cone angle, and the shadow map resolution." Unfortunately, the current algorithm for selecting the bias is a little too conservative. This can cause objects on the ground or abutting a wall to appear to be floating in air rather than up against the surface.

The way to combat this effect is to increase the Shadow Map Size. LightWave 4.0 will have an improved bias calculation algorithm.

Further details on how shadow mapping works can be found in the paper "Rendering Antialiased Shadows with Depth Maps" by Reeves, Salesin, and Cook in the 1987 SIGGRAPH Proceedings.

Before continuing, I cannot emphasize enough the importance of discriminately selecting which objects in your scene will **Cast**, **Receive**, and/or **Self Shadow**.

The render time savings can be tremendous, especially with traced shadows. This is likely common knowledge, but it is worth repeating.

Cheating Shadows

Although traced shadows and shadow maps are the only two explicit methods of shadow generation in LightWave, there are certainly other creative ways to produce them. One common method is the use of dark translucent polygons shaped to mimic a shadow. In fact, this was the primary method of creating shadows in VideoScape, LightWave's predecessor. The technique is particularly well suited to objects that would cast relatively simple shadows, like a ball. But with a little work, even complex shadows can be created with this "cheat."

Start by placing your shadow casting object on a plain white ground object. Then place a light in the scene as it will be relative to the object that is casting the shadow. Now place the camera directly over where the shadow will be, pointing the camera straight down with a pitch of 90 degrees. To reduce perspective distortion, you might want to increase the camera zoom while placing it further away from the ground plane. Then dissolve the object 100 percent and render with shadow mapping enabled. This should yield a clean unobscured shadow. Adjusting the map size and fuzziness, you can control how hard or soft your shadow edges are.

The saved image can be used as a template for creating a shadow object in Pixel3D or Modeler, or can be simply transparency mapped onto a single polygon.

However, actually creating the shadow object will allow for easier placement relative to the object casting the shadow. Also, you may still want to use the image for a transparency map on the shadow object in order to provide the soft edges. Of course, this technique is really best suited for shadows against a flat plane. Figures 3 and 4 show a shadow object placed in layout and the rendered result.

Another method to simulate shadows is through diffuse maps. This technique is similar in execution to the previously described method, but has the added capa-

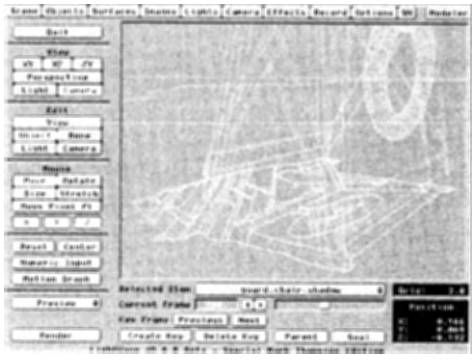


Figure 3

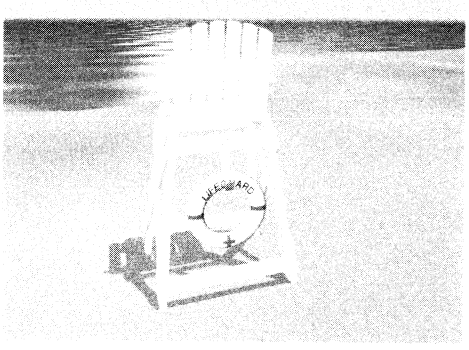


Figure 4

bility of being usable on irregular surfaces. A typical application for diffuse-mapped shadows is in the simulation of the lighting variation caused by overhead clouds in an outdoor scene. A little bit of diffuse animated fractal noise over the ground works very well to accomplish this effect.

Of course, the easiest shadow cheat, and the one I use most frequently, is to pre-render static portions of a shadowed scene and create foreground and/or background images from them. This can be tricky because with shadows, even small amounts of motion can cause a great deal of change across the image. For a foreground image, render a typical frame from your sequence, load it into your favorite paint program, and slosh full intensity magenta (255, 0, 255) over any non static areas. Then load it into LightWave as a **Foreground Image** with **Foreground Key** enabled (**Effects** panel) and set both high and low clip values to magenta.

continues on page 17

LightWave 101

Methods of Modeling, Part 1

by Taylor Kurosaki

There is one very important concept one must learn early on with regard to LightWave Modeler: For the most part, Modeler is not a conceptual tool, it is an execution tool. You will be much more effective if you first decide precisely what it is you want to model before running the program. If you are replicating a physical object, look all around it and visually break it down into its basic components. If you are modeling from a drawing or photograph, try sketching it from different perspectives to grasp its whole shape. Once you have done this, you must next decide on a modeling approach.

Primitives-based modeling utilizes the basic shapes of boxes, discs, cones and balls. These shapes can then be used as they are, or modified with any combination of Modeler's tools. The following tutorial illustrates the process of building a model entirely from primitives.

Primitives Tutorial:

Lamp Table

In order to build a convincing table lamp, we need a table to set it on, so let's begin there:

Enter Modeler, and set the grid size to 200mm (use the < and > keys to zoom in or out). Select **Disc** under the **Objects** menu. Click on the **Numeric** button (n) and enter the values in Figure 1. Select **OK** and hit the Enter key to make the Disc.

Figure 1

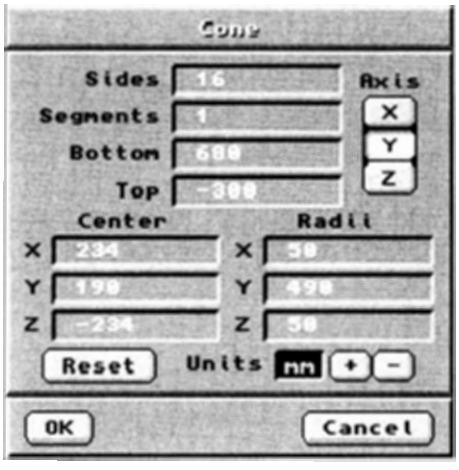


Figure 2

The table body will be made from the box primitive. Press the 2 key to enter the second layer in Modeler. Select Box from the Objects menu. From the Numeric requester, enter these values: Low: -250, 350, -250, High: 250, 680, 250 Segments: 1, 1, 1. Click OK and hit Enter to make the box.

The Cone tool will be used to make the table legs. Press 3 to make that layer active. Select **Cone**, open the **Numeric** requester and enter the values in Figure 2. Notice how the top value is less than the bottom value. This inverts the orientation of the cone such that it tapers from top to bottom. Click **OK** and hit enter to make the first table leg.

To construct the remaining three legs, select **Clone** from within the **Multiply** menu. Set the Number of clones to 3, with a Rotation of 90 degrees along the Y Axis. Click **OK** to clone the leg.

The table will look more conventional if the legs don't taper so much that they end at a sharp point (unless you would like holes in your floor). They have been constructed extra-long to begin with so we can go back and cut off the tips. The best operation for this purpose is a **Boolean** Subtract (**Tools** menu). A Boolean operation works by affecting the contents of the foreground layer with the contents of the background layer. More specifically, a Boolean Subtract works by cutting from the foreground layer any geometry that overlaps with geometry in the background layer. Additionally, Boolean is a Constructive Solid Geometry (CSG) operation. It adds polygons to any affected areas in the foreground layer, making the object appear solid where it was cut.

Press 4 to enter that layer. Press Alt-3 to put the table legs in the background layer. Select **Box** from the **Objects** menu. Draw out a box such that its top face has a Y value of 0, its bottom extends below the bottom of the table legs, and its sides extend beyond the table legs, as shown in Figure 3. Make the box by hitting the enter key.

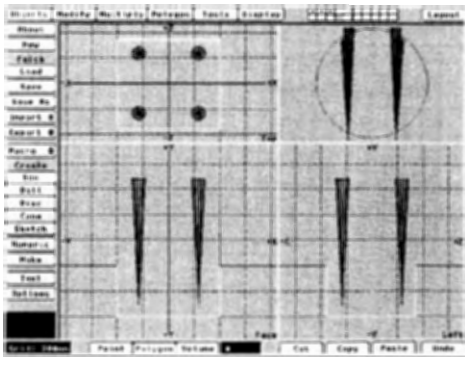


Figure 3

We have just created the template from which we will perform the Boolean operation on the table legs. Click the (`) key to swap the foreground and background layers. Now select **Boolean** (**Tools**) and then Subtract from Boolean requester. As stated earlier, and true to its CSG designation, the Boolean operation automatically added polygons to the bottom of the table legs after cutting off the ends.

Finally, two false drawer handles are made simply by using the Ball primitive. **Cut** the Boolean template from layer 4. Select **Ball** from within the **Objects** menu, press n to bring up the numeric requester, and enter these values:

Ball Type:	Globe
Sides:	16
Segments:	8

Center:	0,570,-265
Radii:	15,15,15
Units:	mm

Click **OK**, and hit Enter to make the ball. Select **Copy** (c) to copy the ball. Next, press (t) (for **Move**), and move the ball -120mm along the Y axis. You may wish to hold the Ctrl key while moving the ball to constrain the movement to the Y axis only. Finally press the (v) key to Paste the copy you made of the ball to its original spot.

To save the table as one object, hold the shift key while hitting the 1, 2 and 3 keys to bring all four layers to the foreground (make sure layer 4 is also active). Hit (s) to save all four layers as "LampTable."

Another major modeling technique is freehand modeling. This approach gets its name from the process of creating polygons comprised of hand-placed points. These freehand polygons, or primitives, are then affected primarily by the Extrude, Lathe, and Mirror tools of the Multiply menu, creating the desired shapes.

Freehand Tutorial:

Lamp

Several freehand modeling techniques can be illustrated by the modeling of a lamp. The look we are trying to achieve is similar to a spiraling wrought iron design, with a pleated lampshade. We will begin by building the main body of the lamp:

Click on the **New** button under the **Objects** menu, or press (n). Go to the **Polygon** menu and select **Points.** Use the left mouse button to lineup the cursor, and the right mouse button or the enter key to place the points. To create the outline of the lamp, place points in the Face view at these XY coordinates: 0, 1225; 15, 1225; 15, 1205; 70, 1205; 70, 1195; 60, 1195; 20, 1180; 20, 735; 50, 735; 190, 710; 190, 700, 0, 700mm. [*Editor's note*: To quickly enter these points, use the Enter Points macro.]

De-select all points and reselect them in clockwise order. Once the points are selected, hit (p) to make a polygon containing them. This polygon represents the radius of the lamp.

Click the **Polygon** selection mode button at the bottom of the screen and select the newly made polygon. Go to the **Multiply** menu and select **Lathe**. Open the **Numeric** requester. Change the rotation of the Lathe to the **Y** axis. Otherwise, use the default settings. Click **OK**, and press enter to lathe the lamp as shown in Figure 4.

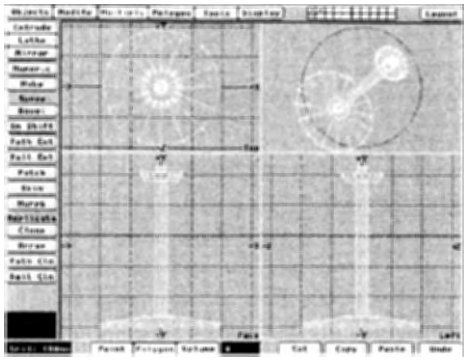


Figure 4

Next to build is the spiral section of the lamp. Whereas the main body of the lamp combined placing points with the **Lathe** feature, this piece of the lamp will utilize primitives, combined with lathing.

Press 2 to make the second Modeler layer active. Select **Disc** from the **Objects** menu. Open the **Numeric** requester and enter these values:

Sides:	16
Segments:	1
Bottom:	700
Top:	700
Axis:	Y
Center:	0,700,-40
Radii:	10,0,10
Units:	mm

Click **OK** and hit enter to make the disc.

Now select **Clone** from the **Multiply** menu. Enter 2 as the number of clones, with a rotation of 120 degrees along the Y axis. Click on **OK** to clone. The resulting three discs represent a cross-section of the spiral section of the lamp.

Select **Lathe** from the **Multiply** menu. From within the **Numeric** requester, change the lathe axis to **Y**, and enter an Offset of 500mm. The offset will cause the discs to spiral upward 500mm to the end of the lathe. Click **OK** and hit enter to lathe the three discs. Your object should look like Figure 5.

Now to build the supports for the lampshade. Press 3 to enter the third layer. Select **Disc** from the **Objects** menu and enter these values in the Numeric requester:

Sides:	16
Segments:	1
Bottom:	1.2
Top:	1.4
Axis:	Y
Center:	0.06,1.3,0
Radii:	0.002,0.0.002
Units:	m

Click **OK** and hit enter to make the disc.

Select **Mirror** from the **Multiply** menu. Center the cursor along the Y axis and mirror across the Y Plane, or select Y Plane with a position of 0 from within the **Numeric** requester. Hit the enter key to mirror the support.

Finally, to complete the lamp, we must build a lampshade. Move to the fourth layer in Modeler and make a Disc with these values:

Sides:	64
Segments:	1
Bottom:	1.1
Top:	1.4
Axis:	Y
Center:	0,1.25,0
Radii:	0.07,0.15,0.07
Units:	m

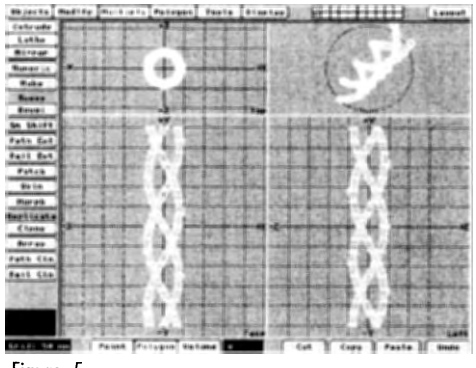


Figure 5

From the Top view, select every other pair of points along the Y axis, beginning with the pair at either 3, 6, 9 or 12 o'clock. Hit the (h) key to select **Stretch** from the **Modify** menu. From the **Numeric** requester, enter stretch Factors of 0.885 for the X and Z axes. Click **OK** to stretch the points inward.

Now we want to give the lampshade the correct shape. De-select all points. Click **Taper 1** from within

continues on page 17

EDITOR'S MESSAGE

continued from page 3

Speaking of things going as planned, the port of LightWave 3D to Windows and the SGI is going extremely well (at least as of this writing). I've seen both versions, and I can say I am extremely impressed with the speed with which Allen and Stuart are getting it ported. (They're porting faster than I can write the manual!) Shipping is scheduled for early 1995.

I never thought I'd say this, but I'm getting kind of excited about buying a PC (read Mojo's PC Primer for more info). I even went out and bought (gasp!) PC magazines with which to start boning up. I'm not sure if I'm excited because it's a new techno purchase or because LightWave is going to run on it. Actually, it's a little of both.

I still run LightWave on a stock Amiga 3000/030, and I want to see it scream! I'm thinking 100MHz Pentium here with a 17-inch (or larger) monitor and a 1GB drive. Of course, I'll have to take a giant leap backward in order to work with the Windows operating system, but it's worth it for LightWave.

Who knows? Maybe someday Windows will catch up with the ease of use of AmigaDOS.

LWP

John Gross
LWPRO *Editor and soon to-be PC owner.*

The PC Primer

by Mojo (with regrets)

OK, kid gloves off.

LightWave is going to be released for the PC market within a few months and, like it or not, it's something we all have to deal with. If you like, you can scream "Amiga forever!" and bury your head in the sand, or you can put aside years of platform bias and see what Big Blue and its clones have to offer.

At the risk of sounding hypocritical, let me start by saying that I hate the PC. I'm as much of an Amiga flag-waver as anyone else and the idea of using one of those outdated, clunky, slow, frankly ugly hunks of silicon trash repulses me. But guess what I found out? One of those pieces of trash will work at four times faster than the most souped-up Amiga and cost half as much. It seems like those techno-doorstops aren't so slow anymore. You know what else? That hunk of junk runs Photoshop, the best paint package/image processor known to man. Owning a PC also means opening up the door to a world of cheaper-than-dirt hardware, bug-free, super-developed software and a million places to buy it all in.

After coming to terms with these facts, I decided to peer over the fence and sneak a peek at the PC universe. For those of you, like me, who have lived exclusively inside the checkered ball, here's what I discovered.

The Suspects

There are basically three types of PCs that will run LightWave. First and foremost are **Intel**-based machines, which revolve around Intel's 486 and Pentium processors (the Pentium chip is actually a 586, but Intel discovered they could not copyright a number, hence the chip name). The Intel processors are more or less the equivalent of Motorola's 68000 series, the heart of our Amiga. Intel-based machines make up the vast majority of PCs on the market and are the same ones you see advertised in so many catalogs and magazines.

Next are **MIPS** and **DEC**. These are two companies who manufacture their own machines based around a custom processor. MIPS manufactures the R4000 series, the same type used by the Screamer and now the Raptor. Digital Equipment Corporation (DEC) is the maker of the new Alpha chip, a super-fast processor at the heart of the machine commonly referred to as (surprise) the DEC Alpha. Both utilize the relatively new 32-bit Microsoft operating system, Windows NT. No bones about it, these two RISC-based workstations are LightWave dream machines. Unfortunately, they are quite expensive and only run a small number of software packages. Although this will undoubtedly change, at the moment an Intel machine is the way to go for the cheap, all-purpose LightWave PC. [See the sidebar for more on the DEC Alpha.]

The Lowdown

PC LightWave is very flexible. Although it is designed to work best under the superior Windows NT, a patch program called Win32-S (included) will allow it to run under the current version of Windows on just about any Intel-based machine. Although most PCs now contain full 32-bit architecture, Windows was written during the heyday of the 16-bit computer, forcing new, 32-bit software to run in a 16-bit mode. The new Windows 95 operating system (due in the coming months) will also feature full 32-bit operation, as well as improved overall performance.

Which OS will prevail? It's hard to say. Windows NT sounds pretty darn good. It offers pre-emptive (true) multitasking and protected mode memory, which means that if a program under NT crashes, it will not affect any others running. In fact, you can kill the crashed task and recover the memory it was using. NT also allows for 32-character filenames (finally!), has built-in networking capabilities and a built-in virtual memory system, allowing you to render scenes beyond the limits of your RAM (although more slowly). It also supports Symmetrical Multi-Processing (SMP), which allows more than one main processor to be installed in a single machine (the speed!). Windows 95 may contain many of these same features, although an assessment of the system is impossible since it is not currently available. At the moment, NT is here and is quickly gaining support.

Fortunately, most Windows software will run under NT using the Win32-S patch (on Intel machines only—MIPS and DEC machines have a different version of NT for their unique processors and can run some Windows software under emulation). Programs supporting full 32-bit operation (like LightWave and Photoshop) will take advantage of NT and run faster than under normal Windows. Applications that will not run at all under NT (games, most likely) will need regular Windows, so the recommended modus operandi would be to install them both on your machine. Since operating systems for the PC are 100 percent software-based (unlike the Amiga's hardwired Kickstart), you can boot into whichever OS suits your particular needs at the moment. This would allow you to run NT specific applications in full 32-bit glory while maintaining full compatibility with the monstrously diverse general PC market (spreadsheets, paint programs, organizers, those darn games, etc.).

The Brain

So now you have the software and the operating system. What kind of machine do you put all this in? Unlike the Amiga market, where the choice is, um... well, at the moment, none, there are literally hundreds of different PCs out there. Some fast, some slow, some big, some small, some expensive, some cheap, ad infinitum. However, this reflects the diversity of applications in the PC universe. We could get one designed to do our taxes, or help us with homework, or publish a newsletter. Alas, our desires are simple: A box that renders LightWave faster than an Amiga and also runs Photoshop.

It just so happens that these requirements lead us to the realm of graphically advanced PCs, which between the past year and the coming one will actually surpass the Amiga. Render speed is probably our number one priority, so we should limit ourselves to two types of base machines with fast processors.

At the "low" end of the spectrum, the older yet mature Intel 486 (66 MHz) is roughly twice the speed of the 40MHz 040 Warp engine, the fastest Amiga available. Given the relative affordability of a system based around this chip (you can get them up to 100MHz), I would recommend this as the minimum for PC LightWave. For approximately $800 to $1,000 more, you can get the latest 90MHz Pentium, which runs at least twice as fast as a 486/66. This is probably somewhere in the vicinity of eight times faster than a stock Amiga 4000, yet the base machine is close to half the price. Since NT alone requires around 12MB of RAM, a base PC system for running LightWave should have no

less than 16M, preferably 32 (although most machines are upgradable to at least 64, usually more).

The Face

The heart of any graphics machine is its display. This is where the Amiga always soared, peaking with a motherboard that could display and animate screens with up to 256,000 colors in a resolution of 768x480. This was, and still is, very impressive. However, true 24-bit image manipulation became the standard and the native Amiga never caught up, relying on third-party graphics boards to pick up the slack. Although there were (and still are?) maybe a dozen 24-bit display boards, with the exception of the Video Toaster, none of them ever found much support. The programs that made use of them were few and far between and the relatively small Amiga market meant a small user base and therefore a slow, sometimes non-existent upgrade path. In the end, we all paid a lot more for a graphically advanced motherboard that had few uses in a 24-bit world.

On the other hand, the PC was always a graphics misfit. In fact, the machine itself was never really capable of displaying true graphics of any kind—this was always left up to third-party developers. The result? A more affordable base machine and—at last count—no less than 54 inexpensive graphics boards, ranging in price from $200 to more than $2,000, all with 24-bit color. This highly competitive market has produced many low cost, high quality boards with capabilities that far outreach the Amiga, and their high user bases have insured regular upgrades and professional support.

Most of these boards fall under the latest standard of SVGA, offering resolutions of 640x480, 800x600 and 1024x768, some as high as 1600x1200. Each of these resolutions is capable of displaying up to 16 million colors (known as 24-bit or 'true color' in the PC world). The number of colors that can be displayed in any given resolution depends on how much graphics RAM you have installed on your board (several Amiga boards also have installed RAM, resulting in much faster display updates and graphics manipulation—the same principal applies here).

Most boards come with 2MB of Dynamic RAM (DRAM), sufficient for 24-bit color in resolutions up to 800x600—higher resolutions get bumped down to 65,000 or fewer colors unless the DRAM is upgraded to 4MB. For faster graphics and better performance in higher resolutions, the quicker (and more expensive) Video RAM (VRAM) is recommended, although for LightWave work and basic paint operations, 2MB of DRAM is more than sufficient. If constant 24-bit image manipulation is your goal, go with VRAM. A good card like this will set you back all of $400; half of that for a DRAM card. The manufacturers to look for include Diamond and Number Nine.

Animation and video manipulation is making great strides in the PC market, and many new graphics boards come with features that facilitate this. One area to look to for support in is DCI (Display Control Interface), a new Windows driver that will speed up video playback and benefit other animation tasks. Many graphics boards are implementing DCI technology, so make sure your board includes this if you're going to buy one anytime soon.

A little further into the future will be support for Open GL, a Silicon Graphics-based library of instructions that will improve display updates a thousandfold (real-time shaded previews in Layout and Modeler will be possible). Although this technology is very new, it won't be long before graphics cards include Open GL support, so watch for it.

In addition, Microsoft is just releasing Win-G, an animation speed-up language that will improve PC animation in any Windows application. The improvement is tantamount to the playback increase Amiga Anim7 made over Anim5, if not greater.

Yes, folks, the PC has caught up. Deal with it.

The Slots

Our Amigas have standard Amiga (Zorro) bus slots and a few specialized ones, such as the CPU and Video slots. The PC standard are called ISA slots, which hold modems, serial cards, sound boards and other basic add-ons. More advanced hardware (including the graphics boards mentioned above) use the more advanced PCI or VESA bus (the VESA bus is also called a VL-Bus, or VESA Local Bus). Most cards out there support the VLB, although the newer and better PCI bus is fast becoming the slot of choice. You should try to get a machine that supports both, although PCI is the way to go if forced to pick one. As with Amiga slots, the more ISA slots your machine has, the better.

The Bottom Line

OK, so we've got a 90MHz Pentium, 32MB of RAM and a 24-bit graphics card with 2MB of DRAM; the basic requirements of our LightWave box that also runs Photoshop. Throw in Windows NT, a 1GB hard disk, double-speed CD-ROM drive (a necessity), a 17-inch monitor (flicker-free, of course), fax-modem, mouse, keyboard, assorted software and a 3-year warranty. Total price? Around $3,500, if you shop around. Close to a grand less if you want to settle for the 66MHz 486 (still much faster than an Amiga). Not too shabby, eh? An Amiga 4000 with a Warp Engine and all these extras would cost, well... a lot more. If you could find one.

As I said before, I hate the PC. OK, so maybe it's not so slow, even outperforming the Amiga in most areas. OK, OK, and it's cheaper and more reliable and easier to shop for, but... well... I still don't like them.

So I try not to think about the dumb commercials and the fact that everybody has one and how ugly they are.

I just think of a cheap box that runs LightWave eight times faster than my 4000.

And Photoshop, too!

LWP

It looks like Mojo is going to be buying a PC.

The DEC Alpha:

FASTER THAN A SPEEDING BULLET

OK, hotshot. You have money to burn and couldn't care less about games and spreadsheets—you just want the fastest LightWave machine in the west. Look no further, partner—you found it: the DEC Alpha. Relatively new, the heart of this speed-demon is a 275MHz RISC-based Alpha processor, literally the fastest in the world. One of these machines beats the pants off the Screamer and Raptor, and is calculated at 18 times faster than a 33MHz Amiga 040. Apart from its speed, what sets this machine apart from previous LightWave renderers is its ability to perform as a fully functional computer, complete with a mouse, a keyboard and everything! Running a custom-compiled version of Windows NT, this machine is probably the best workstation money can buy, although the applications it can run are limited. Since it sports a custom OS, software must be compiled to specifically run on the Alpha. At the moment, this means LightWave and a few others, although programs such as Photoshop can run under a Windows emulation mode (slowing the system down). However, at the rate these things are catching on (Amblin Imaging and Foundation Imaging now use them), it's a safe bet that more native software is on the horizon.

What exactly do you get for the $11,500 this silicon sex-machine will set you back? Plenty—the world's fastest processor (which will soon be faster), 64MB of RAM, 1GB hard disk, 64-bit graphics board with 4MB of VRAM, double speed CD-ROM drive, a top-of-the-line 17-inch monitor, Windows NT, mouse, keyboard and a huge ego.

Expensive? Not for what you get. Remember, the Raptor costs around $12,000, comes with no memory, none of the cool extras and doesn't render nearly as fast.

Be still my beating heart!

In the January Issue

A new season of *LWPRO* gets off to a roaring start, featuring a how-to on creating hair in LightWave and a rail extrude tutorial.

End-of-the-Year Index

LIGHTWAVE 101

continued from page 13

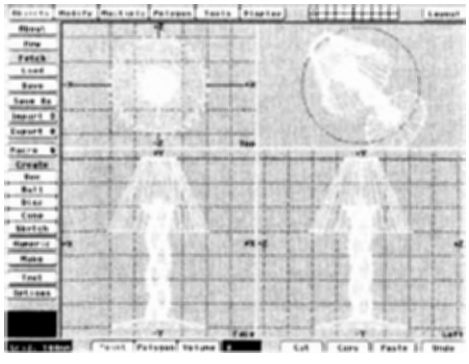


Figure 6

the **Modify** menu. Adjust the **Numeric** settings so they match these:

Axis:	Y
Range:	Automatic
Sense:	-
Factor:	.5
Center:	0,0,0

Click on the **Apply** button to taper the lampshade.

To complete the lampshade, click the **Polygon** selection button, and hit (w) to open the Polygon Statistics window. From this window, select the two polygons with greater than four vertices. These are the polygons at the top and bottom of our lampshade. Press (x) to **Cut** them. Finally, hit (c) to **Copy** all the polygons of the lampshade, (f) to **Flip** the originals, and (v) to **Paste** the copies back down. Under the **Tools** menu, select **Merge** Points (m) to merge duplicate points. Your lampshade is now double-sided.

The lamp is now complete, save for a cord and light bulb (I'll leave those to you). Make all four of the lamp's layers active, and save the lamp as one object. Your lamp should match the one in Figure 6.

Next month we will cover the basics of the other two modeling disciplines: Splines and Metaform. Spline curves and patches have long been the only way to model complex, curved surfaces. That is, until LightWave version 3.5. Now there are two distinctly different approaches one can take when modeling difficult shapes. As we will see, Metaform literally blows away splines for ease of use and versatility in many cases. However, there are still situations where splines and patches are the easiest way to go.

LWP

Taylor Kurosaki is a visual effects artist at Amblin Imaging. He can be reached at 100 Universal City Plaza, Bldg. 447, Universal City, CA 91608, or through John Gross on-line.

Dark Shadows

continued from page 11

I hope I've covered just about everything you ever cared to know about shadows in LightWave. Clearly, the possibilities are numerous. But what makes LightWave truly flexible is the fact that all the above mentioned techniques for shadow generation can be freely combined in the same scene or image. And fortunately, gone are the days when adding shadows to a scene was too costly in render time.

LWP

Mark Thompson is president of Radiant Image Productions, a 3D animation and special effects production house. Send questions or comments to:

Radiant Image Productions 51 Derry Street, Merrimack, NH 03054 or e-mail to mark@westford.ccur.com.

Interactive Refraction

continued from page 9

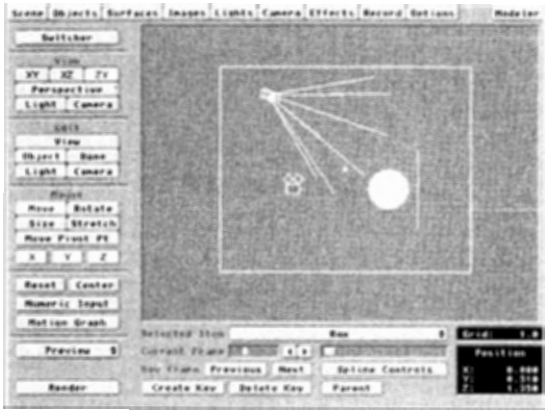


Figure 4: Stretch the box until it completely surrounds all the objects in your scene, including the camera and lights.

Image Map to map the current background framestore onto all six of its inward-facing sides (don't forget to select **Automatic Sizing**).

- Once again set the Luminosity to 100 percent and the Diffuse Level to 0 percent.

Re-render the scene and you'll see that the black areas are now gone. Because our blob now had something to refract from all six directions, it blended seamlessly with the background footage and didn't have a cartoony, black outline. For added realism, we made the blob 15 percent reflective using our background footage as a reflection map. This made the blob look like it was reflecting the surrounding room and completed the illusion.

Time for Coca-Cola

We managed to complete the effects on time and the sequence was a success. I eventually got some sleep, and a few caffeine-laden products later, I was up and running once more. (Of course, I missed the show when it aired, but my mom said it was good.)

LWP

Colin Cunningham served as an FX animator on RoboCop: The Series *and is currently enrolled in the Classical Animation program at Sheridan College, where he wreaks havoc on a daily basis. He likes Coca-Cola and is a MUCH bigger Elvis fan than Mojo (ah, thank yuh). He can be reached at (905) 338-8033.*

Digital Cinematography

continued from page 7

to pan quickly to keep the car in frame. As the first car moves away from us, we whip pan again, panning quickly back to the chasing car.

A more traditional method of shooting this scene might still convey the same information, but not in the same exciting fashion. If the camera were to just allow the cars to shoot through the frame without panning at all, the shot would be relying on the motion of the cars alone. The camera could also whip pan with the first car, then let the other car enter frame in pursuit. Again, the scene is exciting but not to the best of our ability. By whipping back to the chasing car, we are showing how close the two cars are to each other, and implying the speed at which they are traveling.

An age-old technique that has found its way back into mainstream use is the handheld shot, simply described as holding the camera in the hand rather than on any camera support system such as a tripod or dolly. Though it is a common technique used for point of view shots, shows such as *NYPD Blue* have recently began using handheld to create a "you are there" feel for the viewer. By removing the smooth look of a conventional camera head and dolly, the audience feels as if they are actually in the scene, viewing it in close-up. The handheld look should be used sparingly in most cases, as it can be distracting.

In a LightWave shot, however, this type of shot can help remove the fluidity associated with most CGI work. By nature of its definition, handheld removes the perfect framing or the exact steadiness found commonly in photography. By holding the camera on your shoulder, the frame tends to jump around and appear unsteady.

Though it is used to imply the view through someone's eyes, unless we are using some chemical assistance, the view through our eyes is usually much more steady. Yet, for some reason, it is this unsteadiness that we accept as our view on the world.

In the CGI world, utilizing handheld is somewhat difficult since the feel established by a live action camera is somewhat random. The best circumstances to make use of this technique is in a turbulent environment or, for example, near an explosion, essentially shaking the camera as if the shock wave hit. Sometimes it is much simpler to shake the camera, rather than shake the objects in the scene to imply a rough environment.

In live action, in an effort to maintain the versatility and mobility of handheld without the shakiness, the Steadicam was invented. A device that, with the camera mounted on top, counters the unsteady motions of handheld, has become quite popular. Many directors prefer to use Steadicam exclusively, thus removing the need for a dolly all together. In CGI, we cannot avoid achieving a Steadicam feel in all our shots. By nature, the computer camera is steady, unless we choose, by force, to make it otherwise. By adding a little bank to any CGI dolly shot, you will have successfully achieved a Steadicam feel without much effort.

Settle Down, Beavis!

Basic principles of camera movement can be applied to any situation with very desirable results. A common mistake among young camera people (especially CGI camera people), however, is moving the camera too much. It is important to simplify your shot to avoid doing too much. Though it may be cool to boom down, dolly back and tilt to reveal an object, it could be confusing and unnecessary to use such a complex move. It might be just as effective to simply tilt down to the object.

Though shots such as the whip pan example above are fun to create, these are specialized shots and should be used sparingly. Though it may be difficult to fathom, camera motion requires a certain emotional commitment. You aren't just panning and tilting the camera around. You, as the eyes through the camera, are revealing, or not revealing, information to the audience. What you choose to show, or choose not to show makes all the difference in the world. As mentioned earlier, the camera is also an actor. Try to perform your moves to help complement the scene and the other performers. Sometimes you might want to softly pan the camera across a shot to imply a quiet moment, or dolly quickly to a low angle of a creature to imply power. Every move the camera makes is important, and every frame should be considered.

Naturally Done

On a side note, I was recently watching *The Natural* and, as usual, marveling at Caleb Deschenel's simplistic approach to shooting a scene while still achieving a beautiful image. If you get a chance, rent this one; it's worth studying.

I was hoping to get into specifics of various ways to move the camera in LightWave, but I think this column is running on. Next month, we can examine how to use null objects to track a complex move to camera, how to keyframe complex moves that combine the camera and another object, and much more. Until then, keep up the good work.

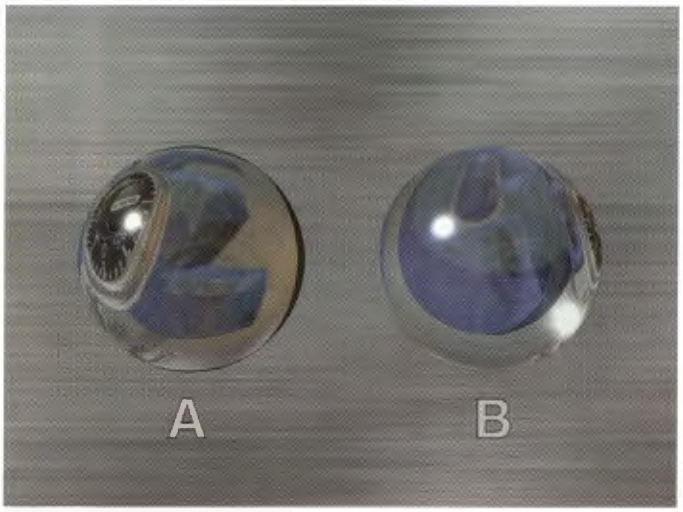
LWP

John E.K. Parenteau is a vice-president and general manager of Amblin Imaging, whose CGI work can be seen regularly on seaQuest DSV.

Projected Shadows

By mapping an image as a surface map onto a transparent plane and selecting Color Filter, you can project shadows onto your objects using the color of the image.

Copyright 1994 Mark Thompson



Glass Spheres

By enclosing our objects in an unseen box the cartoony black outline around sphere A disappears and creates a more realistic-looking glass sphere (sphere B).

Copyright 1994 Colin Cunningham

Depth of Field

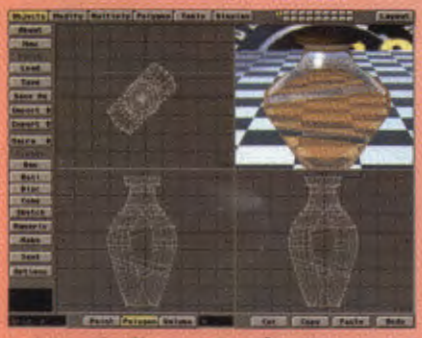
Changing depth of field shifts the viewer's attention from the foreground to the background.

Image copyright Ernie Wright 1994.

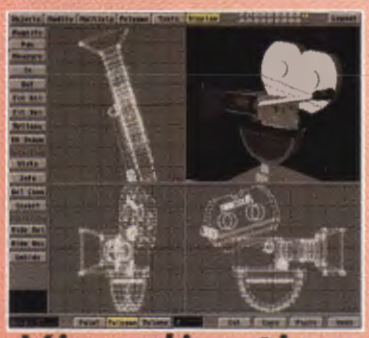
NEW ERA PRESS Announces

LightWave on Location!

The Digital Directors Guide to Desktop Animation!



Modeling Techniques



Visualization

Animation Concept & Design

Begins Shipping Jan 15th!!

This complete reference and training manual is geared for all LightWave enthusiasts from beginner to advanced.

Covered topics include:

- **Quickstart** - basic tutorials designed to help novice modelers and animators acquire basic LightWave skills.
- **Exploring LightWave!** - an advanced section includes complete explanations and tutorials on all of LightWave's functions.
- **LightWave Applied**- complete production examples from storyboard design to client meetings.
- **Theories of 3D Fundamentals** - including lighting, color, motion, scene design and directing techniques as they relate to LightWave.
- **And much, much more!!**

ORDER *LightWave on Location!* NOW

Please send me ___copies of LOL for just $59.95 each. All orders shipped C.O.D. personal check. Please make checks payable to New Era Press. Add $5.00 per order for Shipping & Handling + $1.00 for each additional copy. Add $15.00 S & H on all orders shipped outside of the continental U.S.

Ordering is Fast & EASY!

Send Check or Money Order to:
New Era Press
23120 West Lyons Ave. #143
Santa Clarita CA. 91321

-or-

Call 1-818-892-9595
to place a phone order

-or-

Fax your order to:
1-805-259-1821

30 Day Money-Back-Guarantee