HDRI 3D

LIGHTING THE WAY FOR DIGITAL ARTISTS

NOVEMBER/DECEMBER 2004

INSIGHTS & TECHNIQUES from Professional Digital Artists

TUTORIALS IN THIS ISSUE:
MAYA · LIGHTWAVE · XSI
PHOTOSHOP · MODO
MIRAGE · SASLITE

ISSN 1551-689X
01
0 74470 92596 5
Issue 1 $10.95 USD $16.95 CANADA

HDRI 3D
M • A • G • A • Z • I • N • E

Lighting the way for digital artists, HDRI 3D presents real world insights and emerging trends in digital production from the digital artist's perspective. Concepts and techniques are shared through production case studies and software specific tutorials by professionals in film production, architecture, advertising, medicine, and other fields. Topics are wide-ranging, covering areas such as Workflow, Visual Effects, Compositing, Lighting, Rendering, Modeling, Scripting, Animation, Character Rigging, Texturing, and more.

Go to www.hdri3d.com for more info!

ONLINE SPECIAL OFFER

☐ US $49 ☐ Rest of World $68

Subscribe to HDRI 3D Magazine
online before December 31, 2004
to get this great discount!

Six issues a year

4 EASY WAYS TO ORDER

1. In our online store at www.dmgpublishing.com
2. Fax to 1-702-616-9647
3. Phone 1-888-778-9283 or 1-702-990-8656
4. Mail to:

DMG Publishing
2756 N. Green Valley Pkwy #261
Henderson, NV 89014
USA

Name ____________________

Address ____________________

____

City/State ____________________

Zip/Postal Code ____________________

Country ____________________

Telephone ____________________

Email ____________________

PAYMENT

☐ I'm enclosing a check in US Dollars payable to DMG Publishing

☐ I wish to pay by credit/debit card

Card# ____________________ Exp Date __________

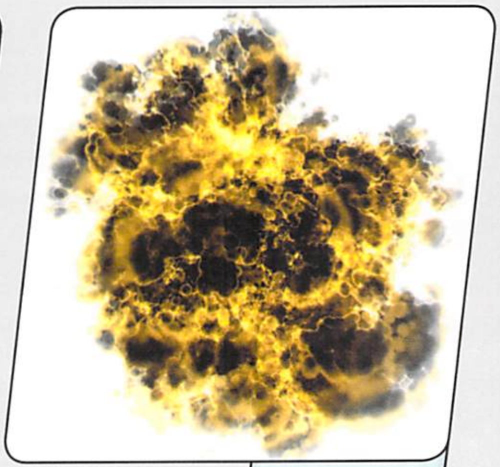
Signature ____________________ Date __________

Lighting the way for digital artists

IN THIS ISSUE...

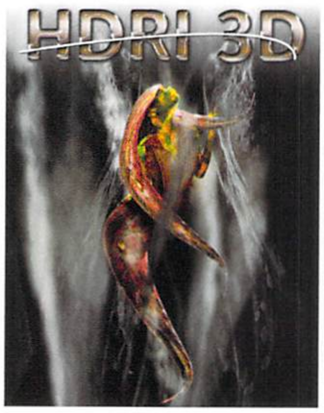
PAGE 24

PAGE 38


PAGE 68

SIPAPU – EMERGENCE by Stephen Burns

Since the objective of HDRI 3D is to address a variety of approaches and tools to digital artistry, Stephen looked for a vision that utilized a variety of media in the effort to create something that was not stylized by only one program. Inspired by American Indian mythology, Stephen created a scene showing a soul beginning to emerge from Sipapu, a place in the earth believed to harbor the souls of the dead, into the world of the living. He shares how he brought his vision to life in his tutorial in this issue.



ON THE COVER

tutorials

reviews

contents

HDRI 3D

Issue #1 • November / December 2004

PUBLISHER
Alice M. Edgin alice@dmgpublishing.com

EDITORIAL DIRECTOR/PUBLISHER
Charles A. Edgin charles@dmgpublishing.com

EDITOR-IN-CHIEF
Dan Ablan dan@hdri3d.com

TECHNICAL EDITOR
Brad Carvey bradcarvey@hotmail.com

MANAGING EDITOR
Amber Goddard amberg@hdri3d.com

CONTRIBUTORS
Eddie Alcazar eddie_alcazar@yahoo.com
Akiko Ashley akikohans@aol.com
Deuce Bennett deuce@ciamgineering.com
Nicholas Boughen nick@rainmaker.com
Stephen M. Burns chrome@ucsd.edu
Andrea Carvey andreacarvey@hotmail.com
Brad Carvey bradcarvey@hotmail.com
Peter Hartwig peter@hartwig.dk
Kenneth Ibrahim ken@shuri-ken.com
Eric Keller robotball@yahoo.com
Greg Lemon greglemon1@yahoo.com
Craig Monins csmonins@hotmail.com
Hunter Wolf hunterwolf@mac.com
Bruce G. Woloshyn bruce@rainmaker.com

ART DIRECTOR
David S. Wilkinson david@sabredesign.net

LAYOUT/DESIGN
Sabre Design artists@sabredesign.net

ADVERTISING SALES
Charles A. Edgin charles@dmgpublishing.com
1-702-990-8656

SUBSCRIPTIONS & BACK ISSUES
Order online at www.dmgpublishing.com
Email: info@dmgpublishing.com
1-702-990-8656 or 1-888-778-9283

FOR ADDRESS CHANGES
Email: subscriptions@dmgpublishing.com
Via Mail: DMG Publishing
2756 North Green Valley Pkwy #261
Henderson NV 89014

HDRI 3D MAGAZINE
(ISSN 1551-689X)
is published bimonthly by
DMG Publishing
2756 North Green Valley Pkwy #261
Henderson NV 89014
USA

DMG Publishing specializes in the publication of high-quality magazines for the digital visual effects community. As a 2002, 2003 and 2004 Graphics Excellence Award winner, DMG's mission is to be responsive to community needs and to draw together the finest to create relevant, superior, and vibrant publications using the latest printing technologies. With a worldwide distribution of its magazines, DMG prides itself on high standards of customer service and subscriber support.

Welcome to the first issue of HDRI 3D magazine!

It's been said that change is hard because people tend to overestimate the value of what they have and underestimate what they might gain. I believe this, and over the past few months as we've geared up for the first issue of HDRI 3D, I've had to convince people that change is good.

This magazine is something I've wanted to see for some time. When DMG Publishing approached me with the idea, I was all for it. My motives were not fueled by business sense, but rather by the benefits we can all gain from such a magazine. There's nothing wrong with a magazine dedicated to just one software application, but if you can learn what others are doing, and how other applications handle various projects, you're better off for it. But you're probably thinking, "Dan, I only use Maya!" or "I only use LightWave," which is great. But one of the ways I first learned LightWave, for instance, was by reading 3D Studio tutorials. At the time, there were no LightWave publications, nor were there any books. By learning enough about my application, I could apply the other program's tutorials. The transition to other 3D tools, such as XSI, was just as easy. Today, every major (and even not so major) 3D application has very similar tools, and it's easier than ever to cross-reference tutorials.

Beyond that, we want to assure you that HDRI 3D will be unlike any other magazine, covering applications such as LightWave, Maya, XSI, Photoshop, and much more, depending on your responses to our reader surveys. And then we'll take it one step further by introducing you to the emerging trends and cool third-party tools on the market, not as reviews, but as tutorials.

Digital content creation ranges far beyond 3D modeling and animation, and this magazine will be right there covering it for you. HDRI 3D welcomes your comments and suggestions, and we thank you for your support.

[illegible]

TSUNAMI

EDIT. ANIMATE. FAST.

TSUNAMI VT[4] Workstations

Our Tsunami VT[4] workstations offer industry-leading performance and proven reliability, ideal for real-time video editing, live switching, and 3D animation. Available in rack-mount or desktop configurations, we use only the highest-quality components, including genuine Intel® processors with Hyper-Threading Technology and your choice of serial ATA or SCSI Ultra320 RAID. Visit www.sharbor.com/build-yours/ and use our super-easy configuration system to get an instant custom quote, or call us to talk with a NewTek-trained VT expert who will help you determine your unique solution.

YOUR NUMBER ONE SOURCE FOR ANIMATION AND VIDEO EDITING

Cinema 4D R9 XL Bundle

Extensive character animation tools, advanced rendering features and amazing particle effects launch your 3D possibilities into the next dimension. An integrated solution for your 3D animation and rendering needs 1669.99
Cinema 4D R9 . . . . .639.99

SOFTIMAGE|XSI 4 Foundation with FREE Training

Take your work to the next level of realism! This robust and cost-effective solution for artists has extensive polygon modeling tools, industry leading Subdivision Surfaces and the revolutionary SOFTIMAGE|XSI toolset. A wealth of standard scripting languages, unmatched realtime shaders, and a workgroup-aware structure make XSI Foundation the flexible choice for your production 485.00
XSI Essentials with FREE Training 1895.00
XSI Foundation Competitive Upgrade 459.00

LightWave™ 3D [8]

This major upgrade to the powerful, Emmy award-winning animation program includes enhancements in character animation, faster IK and FK, new modeling and texturing tools, bone animation tools, and more 1289.99
Upgrade to [8]. 489.99

Class on Demand: Complete Compression for ProCoder 2

Whether you need to deliver content for the Web, Flash, DVD, DivX, or HD, these tutorials will assist you in achieving the best results for your next project. Under the expert guidance of Ben Waggoner, you'll learn the ins and outs of Canopus' ProCoder 2 and so much more 119.99

MeshPaint

Create beautiful surfaces with ease! Fur, hair, grass - it's all simple with MeshPaint for LightWave. Features include surface-aligned modeling operations, interactive mesh cloning, automatic population, curve-based extrusions, and curve-based deformations. Utilize textures, envelopes, and custom-built gradients . .199.00

Vehicle Modeling Volume 1

This comprehensive video will teach you how to spline model a Chevrolet SSR. Gerald Abraham takes you through this process in several easy to follow 15 - 40 minute sections, so you can learn at your own pace. Volume I covers the exterior of the car in detail with more than 8 hours of video training 30.99
Volume 2 . . . . .30.99

Intuos3 Graphic Tablets

Programmable ExpressKeys™ and a Touch Strip are right at your fingertips for convenient keyboard shortcuts, scrolling, zooming, brush size control, and more. Cordless and batteryless mouse and pen included.

4x5	215.99
6x8 *(shown)*	319.99
9x12	449.99

www.sharbor.com

800-544-6599

SAFE HARBOR COMPUTERS

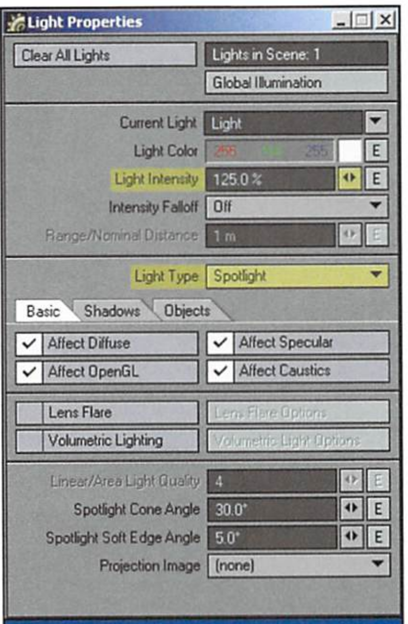
Information 262-548-8120 • Tech Support 262-548-8157 • Fax 262-548-8130 • Hours Mon-Fri 8:30am-5:00pm CST.

All checks require 7 days to clear. Defective products replaced promptly. RMA number required for all returns. Returns accepted within 15 days, in original packaging, postage prepaid, undamaged. Opened software not returnable. Shipping charges not refundable. Returns subject to a 15% restocking fee. Prices subject to change. Not responsible for typos.

BY BRAD CARVEY

LW ESSENTIALS — SASLITE

SasLite is a hair and fur plug-in for LightWave that is easy and fun to use. SasLite is included free with LightWave and is the Lite version of a commercial product, from Worley Labs (www.worley.com), called Sasquatch. The SasLite plug-in can be added to all surfaces or to individual surfaces. Multiple layers of SasLite textures can be added to surfaces. By layering SasLite textures, interesting effects can be achieved. There are lots of ways to use SasLite, besides hair. In this introduction to SasLite, we will texture the billiard 8-ball, which is included with LightWave.


Light Properties

Clear All Lights | Lights in Scene: 1 | Global Illumination

Current Light: Light
Light Color: 255 255 255
Light Intensity: 125.0 %
Intensity Falloff: Off
Range/Nominal Distance: 1 m
Light Type: Spotlight

Basic | Shadows | Objects

✓ Affect Diffuse | ✓ Affect Specular
✓ Affect OpenGL | ✓ Affect Caustics
Lens Flare | Lens Flare Options
Volumetric Lighting | Volumetric Light Options

Linear/Area Light Quality: 4
Spotlight Cone Angle: 30.0°
Spotlight Soft Edge Angle: 5.0°
Projection Image: (none)

FIGURE 1– SET THE LIGHT INTENSITY

SET UP SCENE

1. Load 8-Ball from the LightWave directory (Objects\Games\Billard_Balls\Ball_8.lwo), and rotate it so the "8" faces the camera.

2. Open the Backdrop Dialog Box (Ctrl-F5) and change the Backdrop Color to White.

3. Select Lights (L), Properties (P), and then change the Light Type from Distant to Spotlight and the Light Intensity to 125%. You must manually enter the number 125, because the arrows default to 0 - 100%. SEE FIGURE 1.

ADD SASLITE PLUG-IN

4. Open the Image Processing Window (Ctrl-F8), Click on Add Pixel Filter, and select "SasLite" from the popup menu.

5. Double Click on the text "SasLite: Low Antialiasing" to open SasLite, and select Self Shadows, then "OK." SEE FIGURE 2.

6. Render the 8-Ball (F9). SEE FIGURE 3.

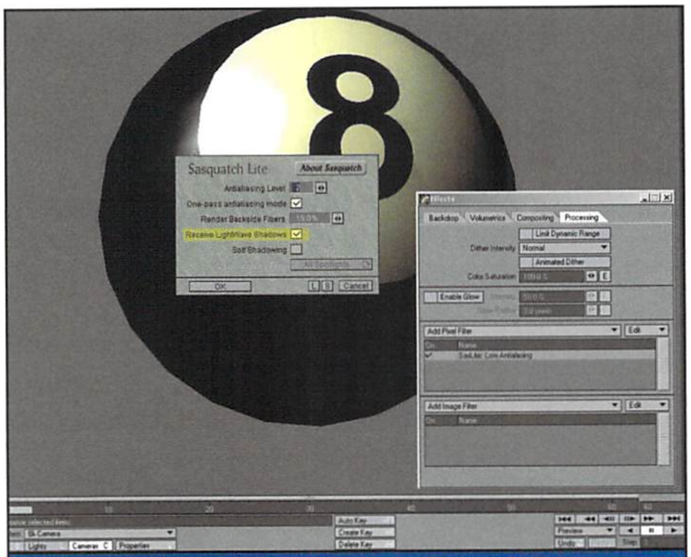

Sasquatch Lite | About Sasquatch
Antialiasing Level
One-pass antialiasing mode ✓
Render Backside Fibers
Receive LightWave Shadows ✓
Self Shadowing
OK | L | S | Cancel

FIGURE 2– IN SASLITE, SELECT "SELF SHADOWS"

7. Note: To keep the tutorial simple, there is only one light at 125% with default scene settings. You can improve the appearance of your rendered images with more lights and by placing the 8-Ball in an interesting environment. Remember to enable the Image Viewer (Items / Rendering / Render Options / Render Display / Image Viewer), to view your rendered images.

ADDING SASLITE TO AN OBJECT

8. Select Objects (O), Properties (P), and then under the Deform tab, click Add Displacement to open the popup menu, and select SasLite. SEE FIGURE 4.

9. Double Click on the text "SasLite" and change values to match FIGURE 5. The Fiber Color should be changed to White (255, 255, 255), then "OK."

10. Render the 8-Ball (F9). SEE FIGURE 6.

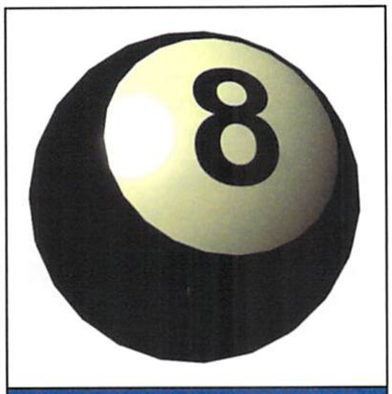
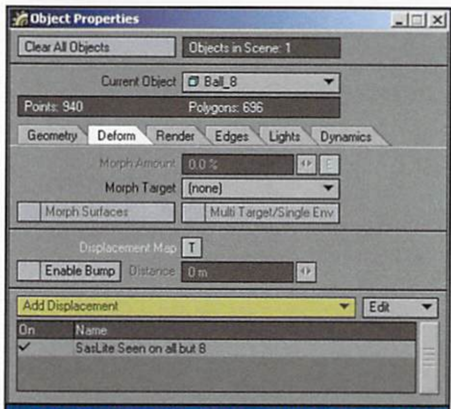


FIGURE 3– THE RENDERED 8-BALL



Object Properties

Clear All Objects | Objects in Scene: 1

Current Object: Ball_8

Points: 940 | Polygons: 696

Geometry | Deform | Render | Edges | Lights | Dynamics

Morph Amount: 0.0 %

Morph Target: (none)

Morph Surfaces | Multi Target/Single Env

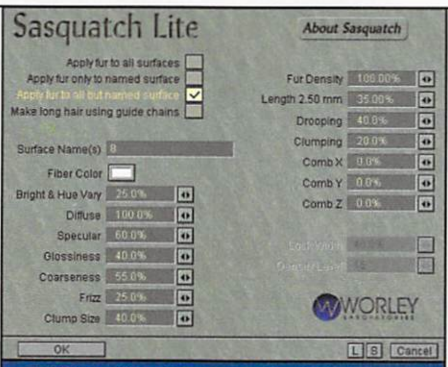
Displacement Map: T

Enable Bump | Distance: 0 m

Add Displacement | Edit

On	Name
✓	SasLite Seen on all but 8

FIGURE 4– SELECT SASLITE UNDER *DEFORM*



Sasquatch Lite

Apply fur to all surfaces ☐
Apply fur only to named surface ☐
Apply fur to all but named surface ✓
Make long hair using guide chains ☐

Surface Name(s): 8

Fiber Color: ⬜

Bright & Hue Vary	25.0%
Diffuse	100.0%
Specular	60.0%
Glossiness	40.0%
Coarseness	55.0%
Frizz	25.0%
Clump Size	40.0%
Fur Density	100.00%
Length 2.50 mm	35.00%
Drooping	40.0%
Clumping	20.0%
Comb X	0.0%
Comb Y	0.0%
Comb Z	0.0%

OK | L | S | Cancel

FIGURE 5– MATCH STEP 9 SETTINGS TO THESE

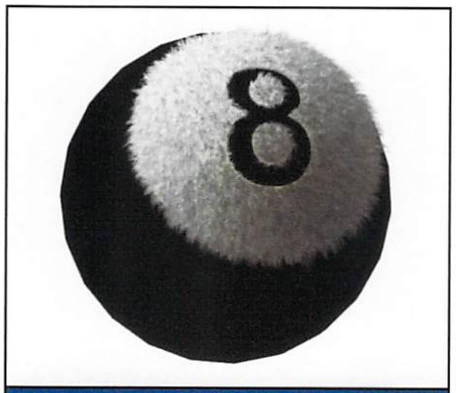
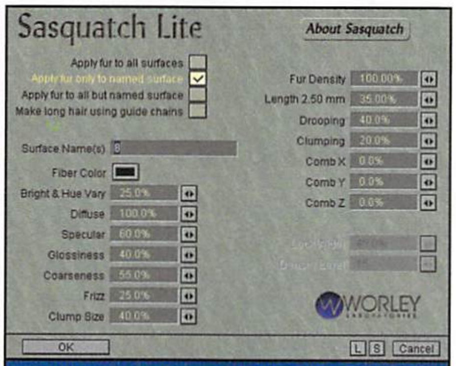


FIGURE 6– STEP 10'S RENDERED 8-BALL



Sasquatch Lite

Apply fur to all surfaces ☐
Apply fur only to named surface ✓
Apply fur to all but named surface ☐
Make long hair using guide chains ☐

Surface Name(s): 8

Fiber Color: ⬛

Bright & Hue Vary	25.0%
Diffuse	100.0%
Specular	60.0%
Glossiness	40.0%
Coarseness	55.0%
Frizz	25.0%
Clump Size	40.0%
Fur Density	100.00%
Length 2.50 mm	35.00%
Drooping	40.0%
Clumping	20.0%
Comb X	0.0%
Comb Y	0.0%
Comb Z	0.0%

OK | L | S | Cancel

FIGURE 7– MATCH STEP 11 SETTINGS TO THESE

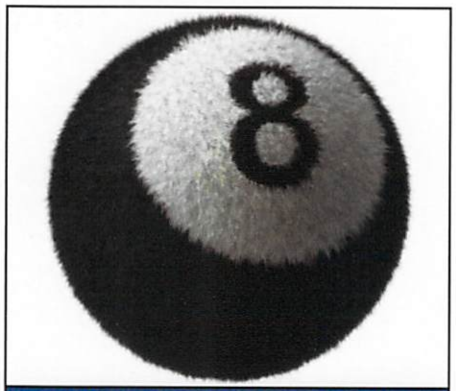
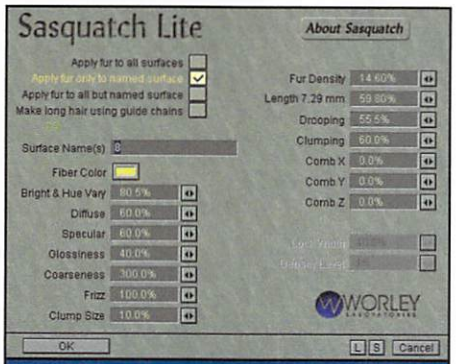


FIGURE 8– STEP 12'S RENDERED 8-BALL



Sasquatch Lite

Apply fur to all surfaces ☐
Apply fur only to named surface ✓
Apply fur to all but named surface ☐
Make long hair using guide chains ☐

Surface Name(s): 8

Fiber Color: 🟨

Bright & Hue Vary	80.5%
Diffuse	60.0%
Specular	60.0%
Glossiness	40.0%
Coarseness	300.0%
Frizz	100.0%
Clump Size	10.0%
Fur Density	14.60%
Length 7.29 mm	59.80%
Drooping	55.5%
Clumping	60.0%
Comb X	0.0%
Comb Y	0.0%
Comb Z	0.0%

OK | L | S | Cancel

FIGURE 9– MATCH STEP 14 SETTINGS TO THESE

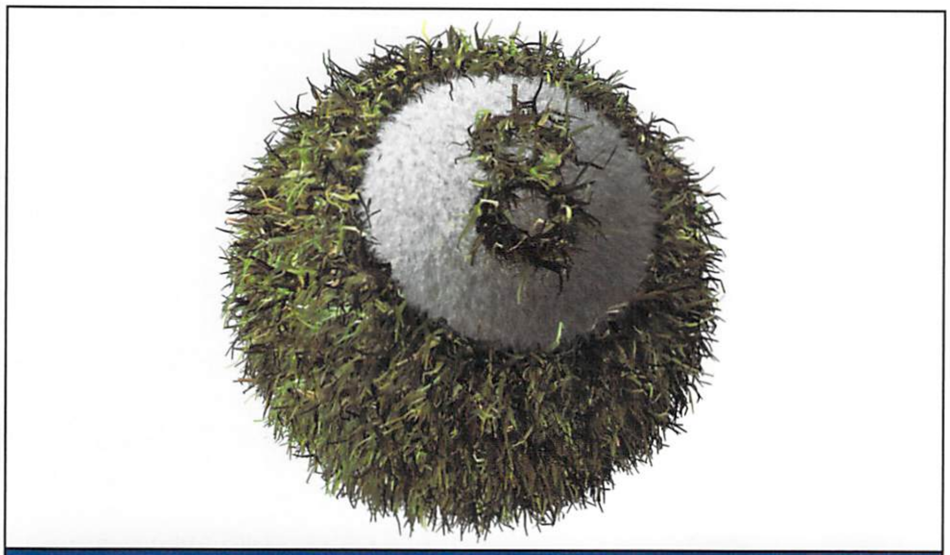


FIGURE 10– THE FINAL GRASSY 8-BALL

11. Add another SasLite Displacement (Step 11), and change values to match *FIGURE 7*. The Fiber Color should be changed to Black (0, 0 .0), then "OK."

12. Render the 8-Ball (F9). *SEE FIGURE 8*.

ADDING GRASS TO THE BLACK SURFACE

13. Try adjusting SasLite parameters so the values are greater than the maximum allowed by the adjustment arrows. For example, to create grass with SasLite, you will need to type in the Coarseness value (200% or more).

14. Add another SasLite Displacement (Step 11), and change values to match *FIGURE 9*. The Fiber Color should be a yellowish color (237, 251, 140), then "OK."

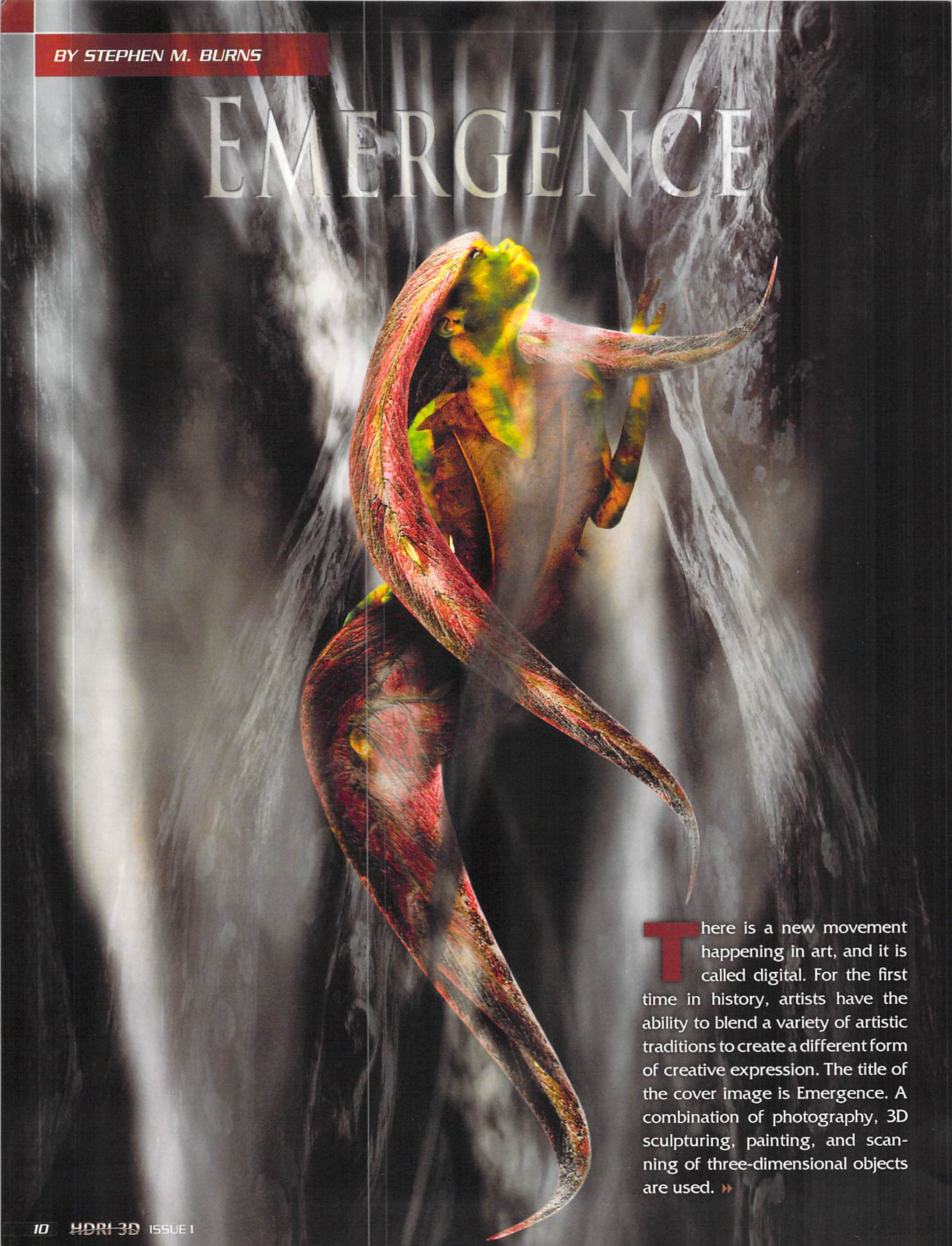
15. Render the 8-Ball (F9). *SEE FIGURE 10*.

Try using one SasLite Texture for short grass with Fur Density at 100%. This will be the short grass. Add another texture with low Fur Density 7%, Clumping at 100% and a small Clump Size. By layering textures, you can get some very interesting landscapes. If you assign different surfaces to landscape objects, you will be able to vary SasLite grass textures and add Clumps of Long Grass.

BRAD CARVEY HAS BEEN DOING COMPUTER ANIMATIONS FOR A LONG TIME. IN 1969, BRAD USED AN ANALOG COMPUTER, WHICH WAS THE SIZE OF A CAR, TO PRODUCE HIS FIRST COMPUTER ANIMATION. BRAD IS AN ELECTRICAL ENGINEER AND AN EMMY AWARD-WINNING MEMBER OF THE VIDEO TOASTER DEVELOPMENT TEAM. HE PREFERS TO DO FEATURE FILM WORK. HIS CREDITS INCLUDE FILMS LIKE *MEN IN BLACK*, *STUART LITTLE*, *BLACK HAWK DOWN*, *KATE & LEOPOLD* AND *MASTER OF DISGUISE*.

BY STEPHEN M. BURNS

EMERGENCE



There is a new movement happening in art, and it is called digital. For the first time in history, artists have the ability to blend a variety of artistic traditions to create a different form of creative expression. The title of the cover image is Emergence. A combination of photography, 3D sculpturing, painting, and scanning of three-dimensional objects are used. »

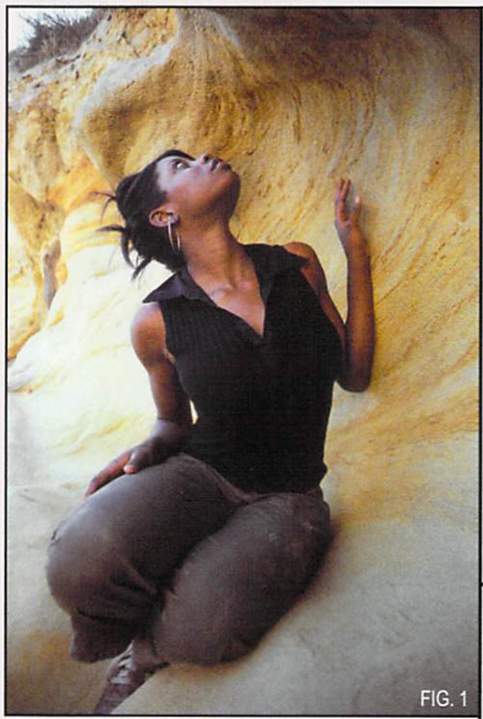


FIG. 1

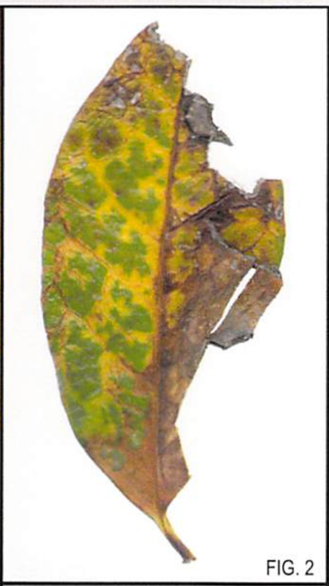


FIG. 2

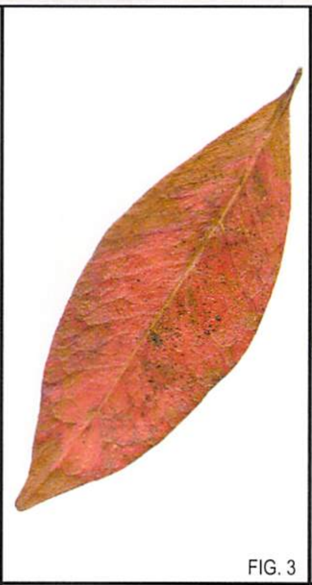


FIG. 3

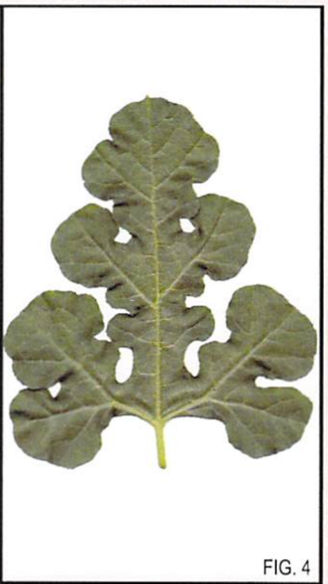


FIG. 4

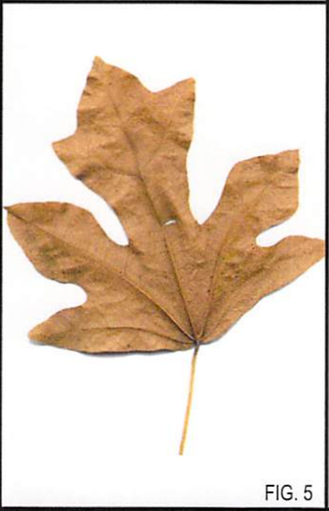


FIG. 5

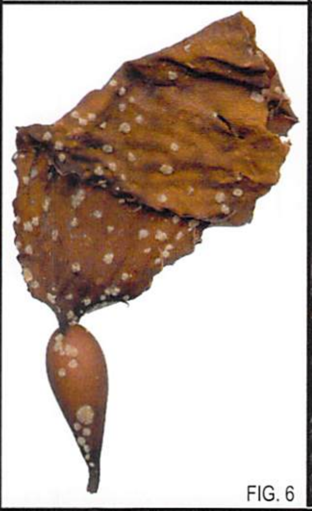


FIG. 6

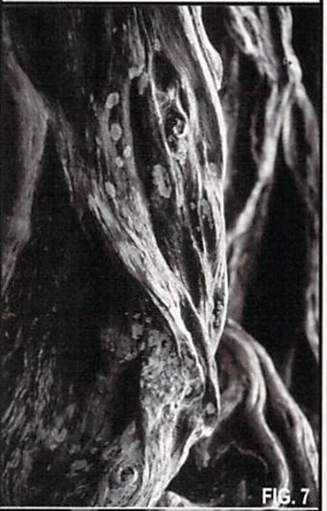


FIG. 7

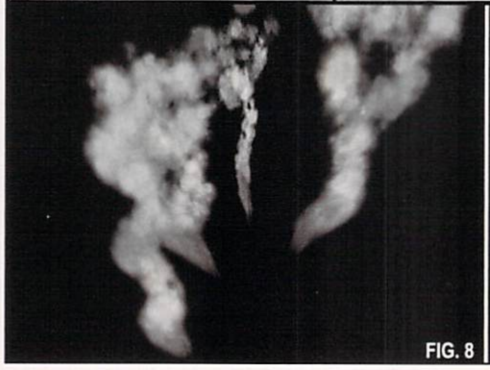


FIG. 8

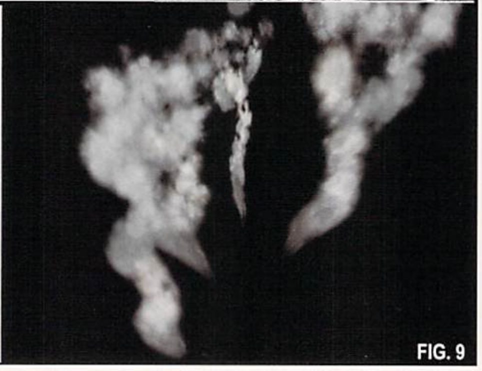


FIG. 9

The following images are used in the creation of *Emergence:*

Figure 01–*Portrait on the beach*

Figure 02–*Green and yellow leaf*

Figure 03–*Red leaf*

Figure 04–*Green leaf*

Figure 05–*Brown leaf*

Figure 06–*Dried kelp*

Figure 07–*Tree bark detail*

Figure 08–*3D generated smoke 1*

Figure 09–*3D generated smoke 2*

The photograph of the model and the photo of the tree bark were shot with the Canon D60 digital camera. This gave me a 6.1 megapixel image. All of the leaves were scanned into my computer at a resolution of 1200 pixels per inch with the HP Scanjet 6300 C, producing file sizes of 30 megabytes and up. The smoke streams are 3D-rendered HyperVoxel objects created in LightWave [8].

I chose these objects primarily for their texture, and with each one being so unique from the other, it was a challenge deciding how to utilize them to complete the final result. I could have gone in a variety of directions, but I chose to envision the textures on more of an abstract level and mold them to become an integral part of the model. Let's discover how it was done.

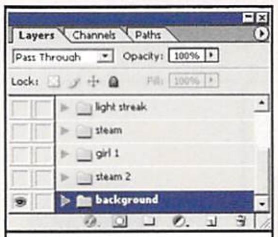


FIG.10– 10 LAYER SETS

STEP 1

To keep the many layers organized, a series of layer sets was produced. Within these sets, the appropriate layers will be placed into them. Each one will be introduced along the way, but I started with the Background layer set. *(Fig. 10)*

STEP 2

The tree bark has an interesting biomorphic feel to it, so it was ideal to serve as a background for my human model. After duplicating the layer *(Ctrl-J)* and inversing it horizontally *(Edit>Transform>Flip Horizontal)*, I used a mask to hide the seams where the two images came together. Painting with black on the mask makes the bark go away. Painting with white brings it back. As a result, the two produced an interesting symmetrical array of fluid shapes. *(Fig. 11)*

STEP 3

Next, the background is given a sense of depth by allowing the viewer to focus more on the foreground areas. Two techniques are used to achieve this. First, a new layer is created. By holding the Alt key on the keyboard and selecting the Merge Visible command from the Layers submenu, all layers will merge into the new one without merging the original ones. The next step involves applying Gaussian Blur to the new layer, because this will serve to provide the shallow depth of field. *(Fig. 12)*

Once completed, a mask is associated and edited so that much of the blur effect is restricted mostly to the center portion of the background, as shown in *figure 13*.

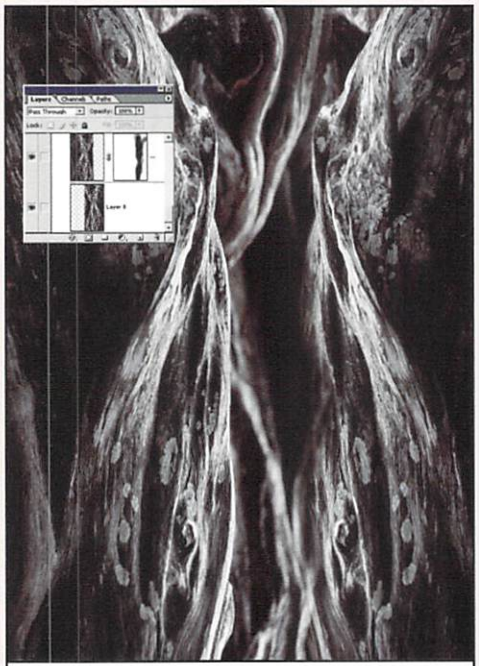


FIG. 11– BACKGROUND CREATED

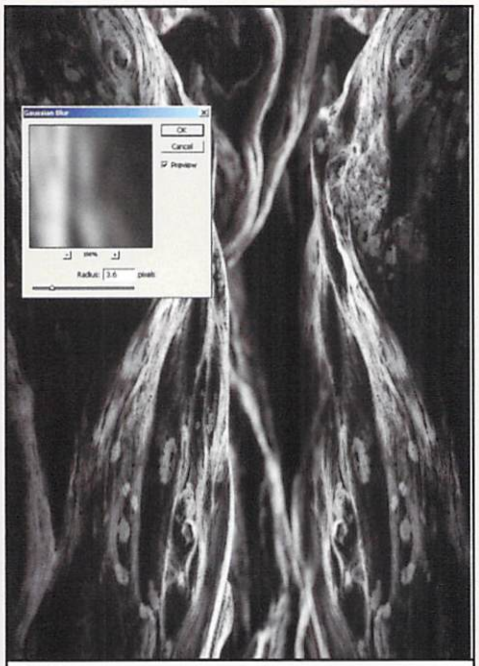


FIG. 12– GAUSSIAN BLUR ADDED TO BACKGROUND

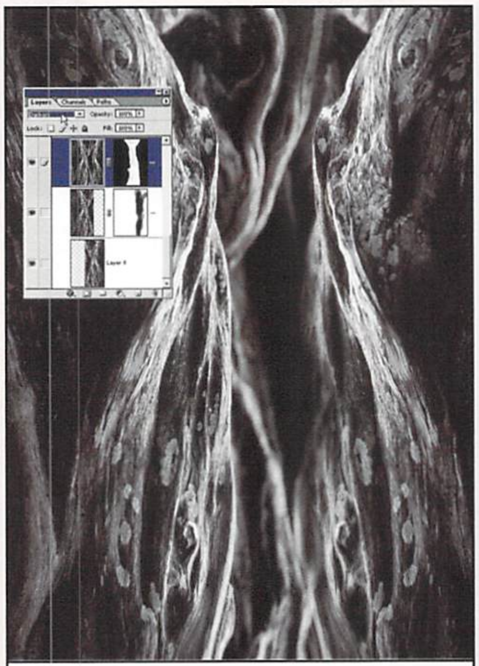


FIG. 13– MASK ADDED TO BACKGROUND

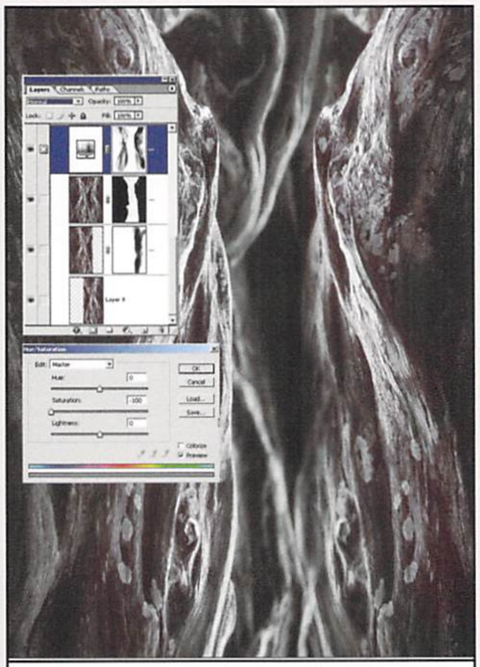


FIG. 14– HUE & SATURATION MASK EDITED

STEP 4

The background has a reddish color cast, so using a Hue and Saturation adjustment layer, the Saturation slider is taken all the way to the left, leaving the image B&W. Because I did not want the color completely taken away, the mask is edited to isolate the reddish hue to the foreground areas.

STEP 5

The background still had quite of bit of tonal information that could compete with the main subject matter, so a Curves adjustment layer is used to restrict the tonal range to deeper shades of gray. The white points on the curve are dragged down to lower the highlights to a richer tone, as shown in *figure*

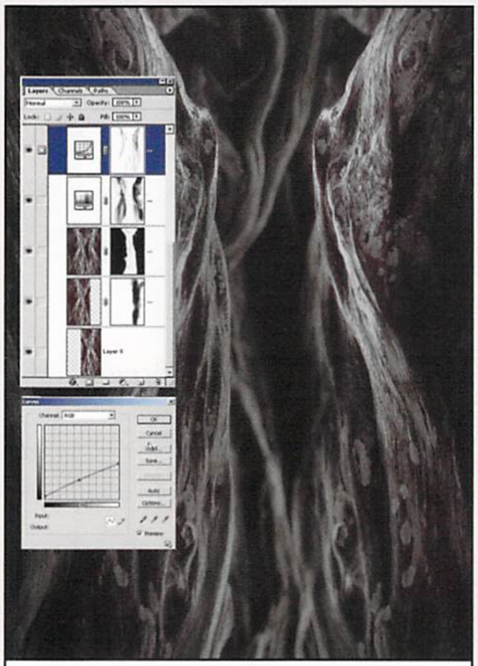


FIG. 15–CURVES ADJUSTMENT LAYER APPLIED

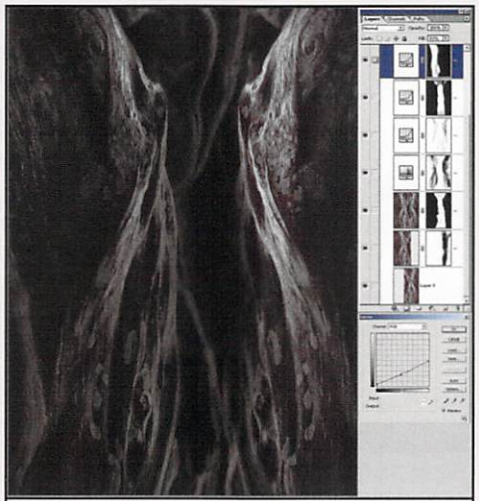


FIG. 16– CURVES ADJUSTMENT LAYER DUPLICATED

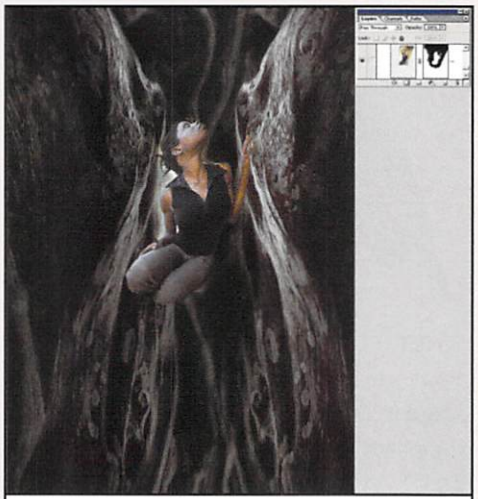


FIG. 17– MODEL PLACED IN "GIRL" LAYER SET

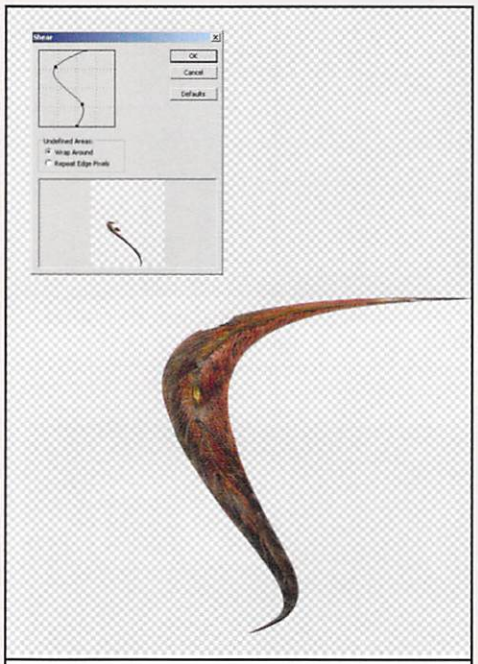


FIG. 18– SHEAR APPLIED TO RED LEAF

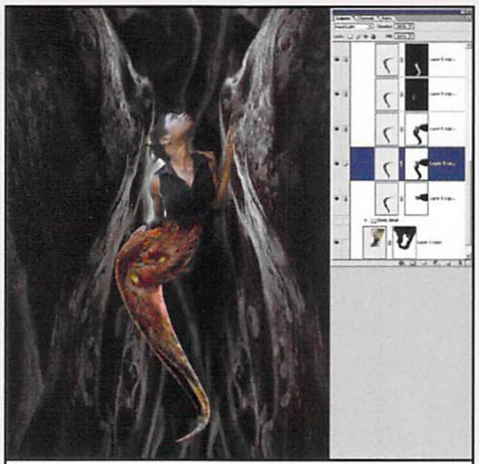


FIG. 19– SCULPTED RED LEAF

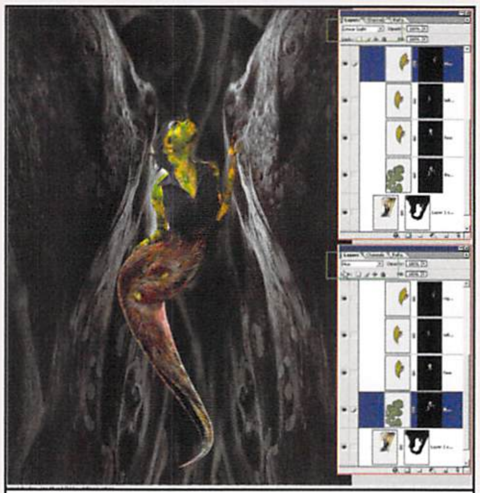


FIG. 20– TEXTURING OF THE BODY

15. The mask is edited to apply its effect in the center region of the background.

STEP 6

The Curves layer is duplicated to insure that the tones are very rich, and once again the mask is edited to restrict the tones to the center of the image. *(Fig. 16)*

STEP 7

Now we are ready to bring in the model. She was placed into the "Girl 1" layer set organized on top of the background, and the Distort tool *(Edit>Transform>Distort)* is used to stretch her body up toward the upper right. *(Fig. 17)*

STEP 8

To change the lower part of her body into a leaf-like figure, the Shear tool *(Filters>Distort>Shear)* is used on the Red leaf. *(Fig. 18)*

STEP 9

The tail is positioned with a mask added. The mask is edited to mold the leaf to the model's lower body. It is also duplicated several times, and each layer is edited to sculpt the leaf in the form shown in *figure 19*.

STEP 10

Next, the Green & yellow leaf and the Green leaf is applied to the body. Figure 20 shows how the Green & yellow leaf's blend mode is set to Linear Light blend mode in one layer example. In the lower layer example, the Green leaf's blend mode is set to Hue. Both examples blend well with the skin and maintain the model's contour. In the next step, let's discover some details as to how this is initially applied. *(Fig. 20)*

STEP 11

Figure 21 shows the leaf before a mask is applied, as well as the results after the mask is edited. The use of the WACOM tablet for editing the mask and applying painting techniques is, in my humble opinion, invaluable.

STEP 12

Here are the results of the Green leaf when applied. Notice how the mask is edited to allow the green hue to dominate only in selected regions. *(Fig. 22)*

STEP 13

In the same way the tail was created in Step 8, the Red leaf is added to the head of the model to represent hair. These layers are placed into the Head Set folder. *(Fig. 23)*

STEP 14

Using the Brown leaf, its textures are restricted to the model's upper body to form a type of clothing unique from the original. No blend modes were used here. I wanted the original detail of the leaf. Finally, the kelp is used to form the collar.

STEP 15

The Brown leaf was so successful for the upper body that I decided to use it as an additional composite for her face and neck region, but this time the blend mode was changed to Overlay. This is an example without the mask applied. *(Fig. 25)*

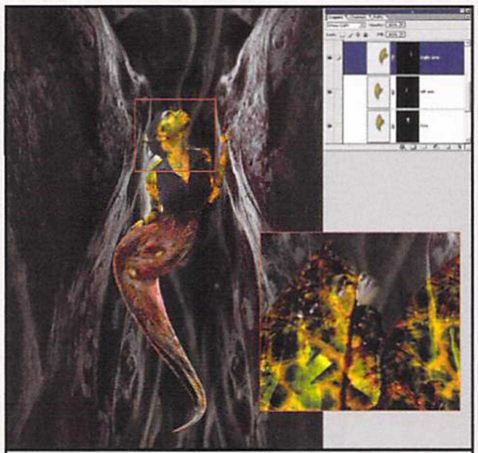


FIG. 21– TEXTURING DETAIL OF GREEN & YELLOW LEAF

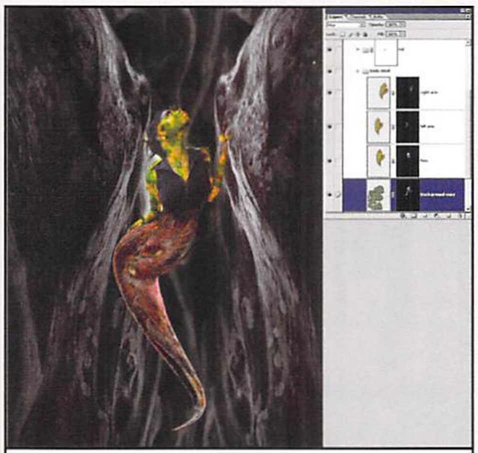


FIG. 22– TEXTURING DETAIL OF GREEN LEAF

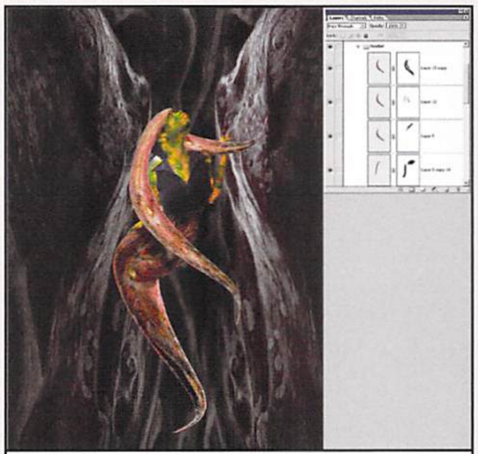


FIG. 23–HEAD DETAIL WITH RED LEAF

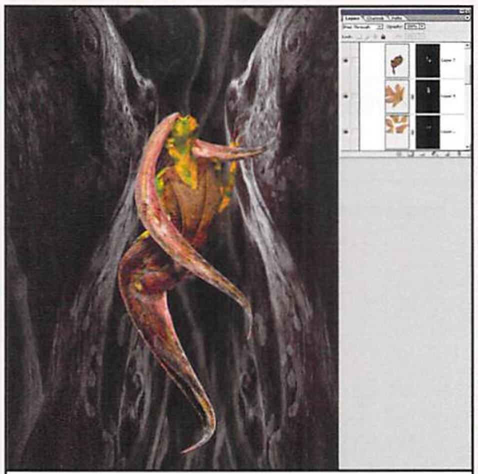


FIG. 24– BROWN LEAF AND DRIED KELP DETAIL UPPER BODY

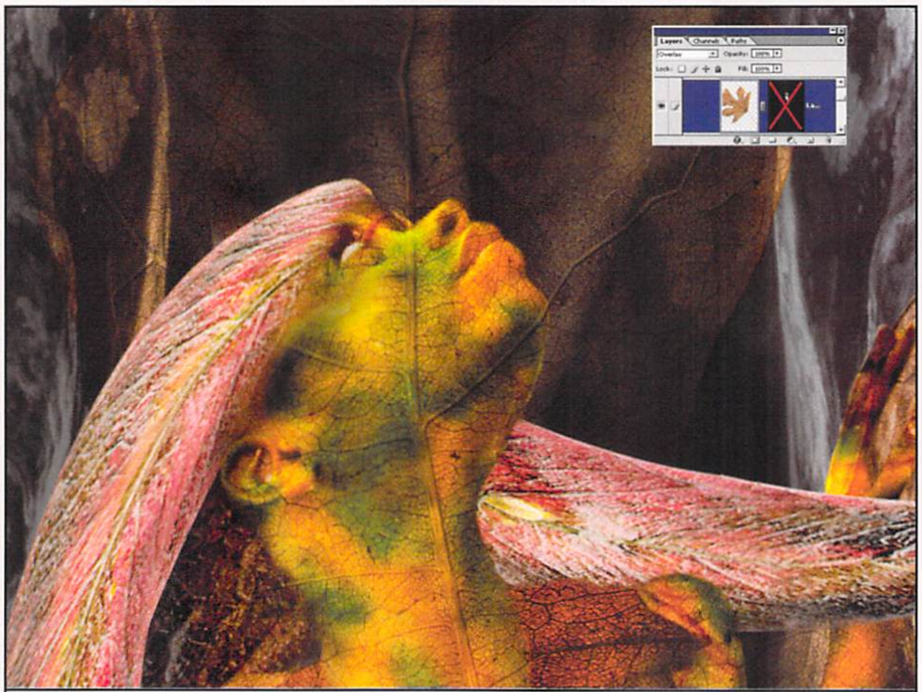


FIG. 25– BROWN LEAF APPLIED TO FACE AND NECK

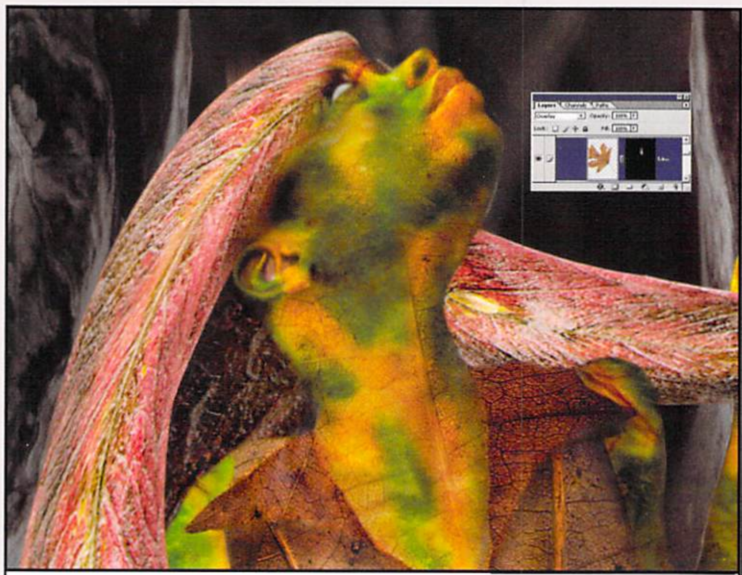


FIG. 26– BROWN LEAF APPLIED TO FACE AND NECK WITH MASK APPLIED

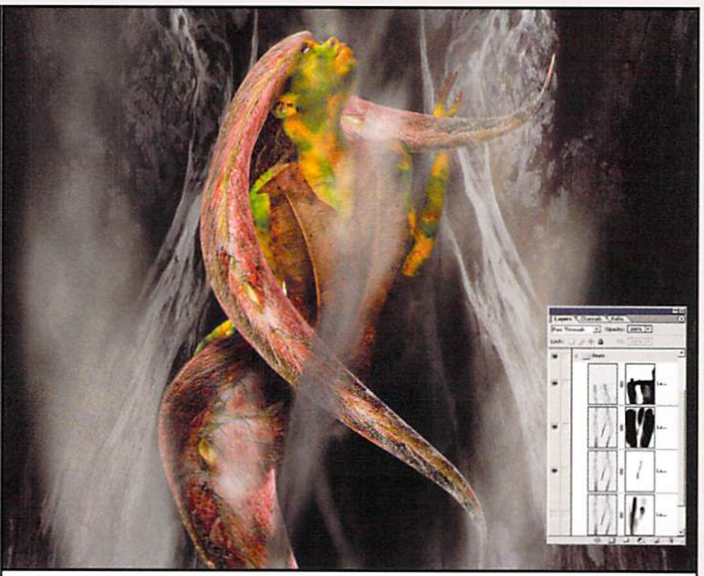


FIG. 27– SMOKE STREAMS WITH MASK APPLIED

Here is an example of the Brown leaf with the mask edited to restrict its detail to the face and neck region. *(Fig 26)*

STEP 16

Now it's time to bring in the smoke streams that were rendered in LightWave [8]. Two images were rendered and saved out as a TIFF so that the smoke would be rendered onto a transparent background. Once the layer was placed, it was resized and duplicated to compose the steamy smoke streams around the model. These images were placed into the Steam 2 layer set that is positioned just above the "background" layer set but below the "Girl1" folder. Later on, you will see the details for creating the smoke streams with Hypervoxels in LightWave [8]. *(Fig 27)*

STEP 17

In this step, more of the smoke streams were applied, but this time they were placed into the Steam layer set, which is positioned above the "Girl 1" folder. These layers' blend modes were set to Screen to enhance the whites. Now the effect is more steam than smoky. *(Fig 28)*

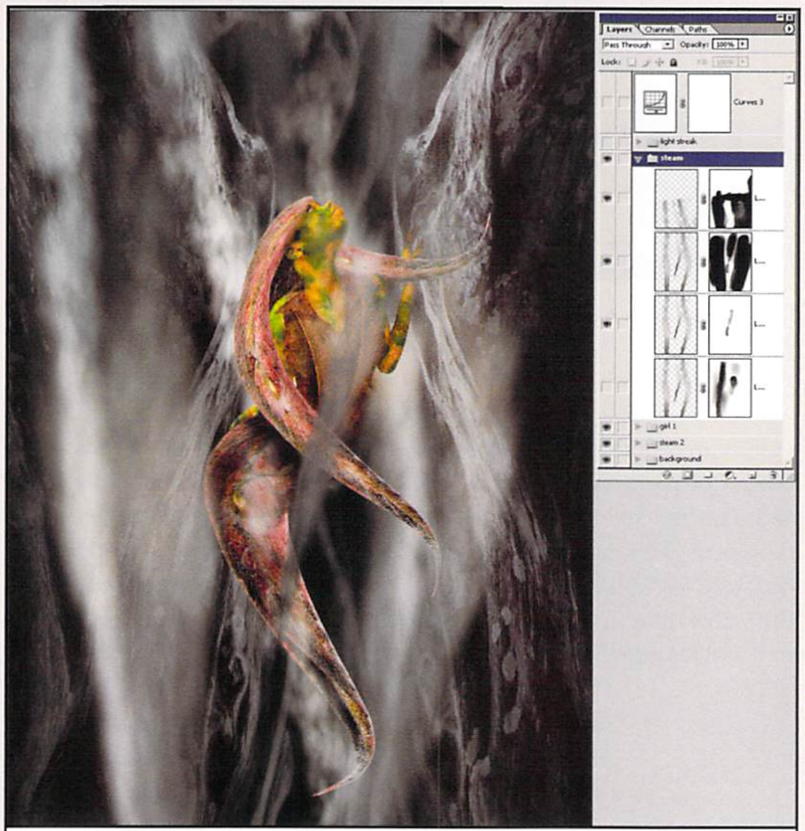


FIG. 28– SMOKE STREAMS WITH MASK APPLIED IN "STEAM" LAYER SET

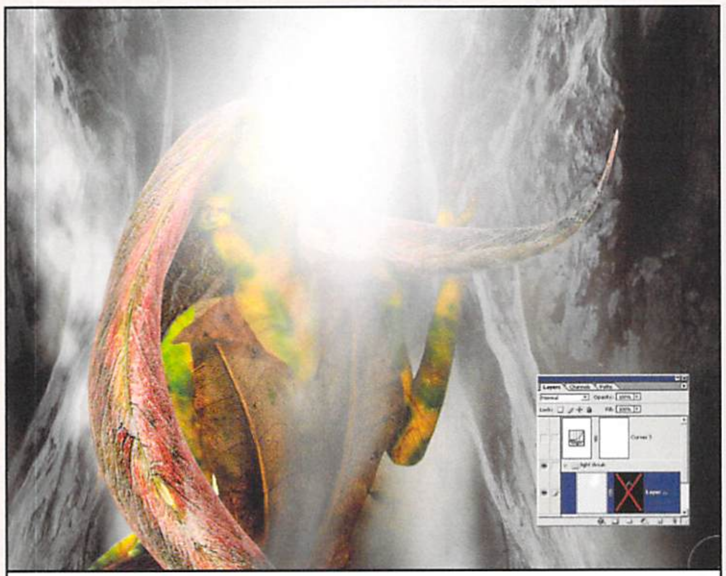


FIG. 29– WHITE PAINT APPLIED OVER MODEL'S FACE

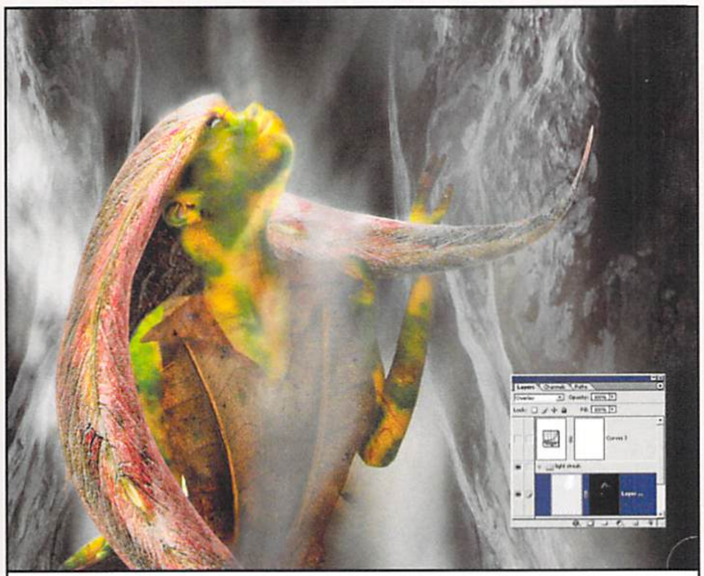


FIG. 30– OVERLAY BLEND MODE SET TO WHITE PAINT

STEP 18

The Light Streak layer set is placed on top of all other layers. Within it, a layer with a small swatch of White paint is applied and positioned over the model's face. *(Fig. 29)*

STEP 19

Once its blend mode is changed to Overlay, a mask is applied to restrict its effects to the edge of the figure's face and leaf hair. This will represent the highlights reflected from above the model. *(Fig. 30)*

STEP 20

The last step is to give the image a little more contrast using the Curve adjustment layer. This did the job.

That's it for the basic image. Now let's look at some of the settings used to create the smoke effects in LightWave [8]. *(Fig. 31)*

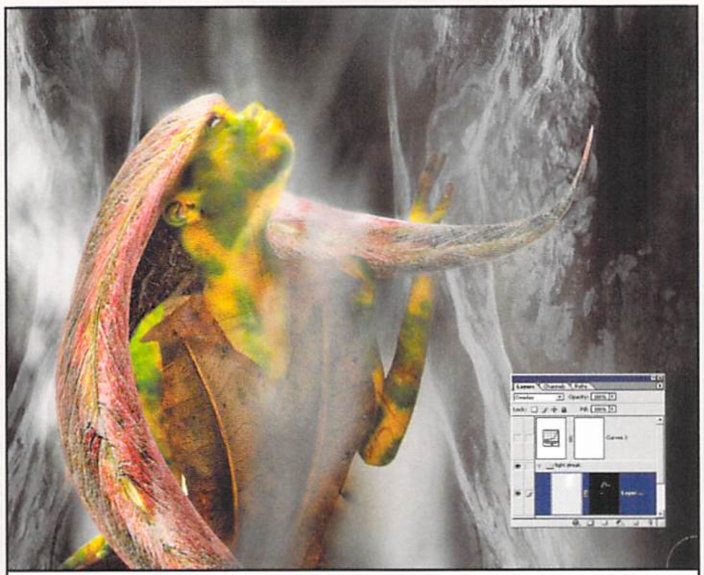


FIG. 31– CURVE APPLIED TO COMPLETE THE IMAGE

SMOKE SETTINGS IN LIGHTWAVE [8]

I used the basic smoke texturing properties that NewTek provided on their site with a slight twist of my own settings. I used a lot of experimentation to get the right amount of particles to interact with the Wind Effector to show movement near the top of the smoke stream, with the lower half left alone. The following steps will show how the particles were composed and textured.

STEP 1

This is the camera view of the placement of three nulls *(Items>Add>Null)* with particle emitters attached to them. To attach the emitter, select a null object and hit the "P" key to get the Properties panel. Under the Add Dynamics panel, select Emitter. Now the object has its own emitter associated with it.

As a note, one null object was created and associated with an emitter. Hypervoxels were used to texture the particles. When satisfied, the null was cloned *(Ctrl-C)* twice, and all three were composed in the camera view. *(Fig. 32)*

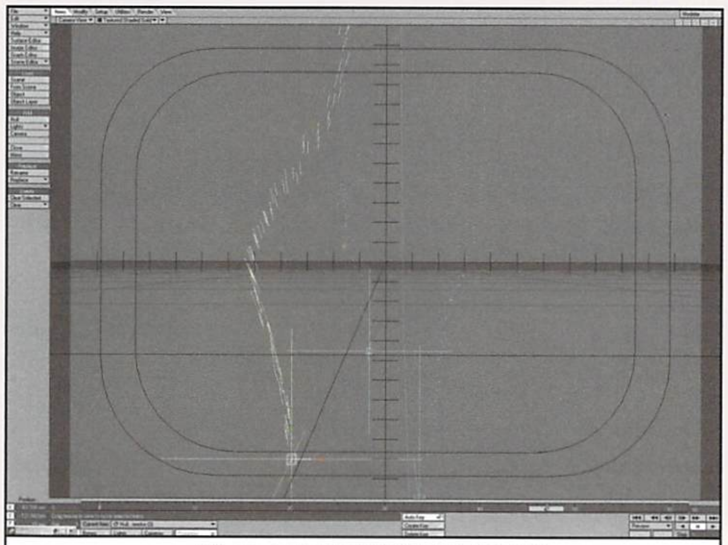


FIG. 32– CAMERA VIEW IN LIGHTWAVE [8]

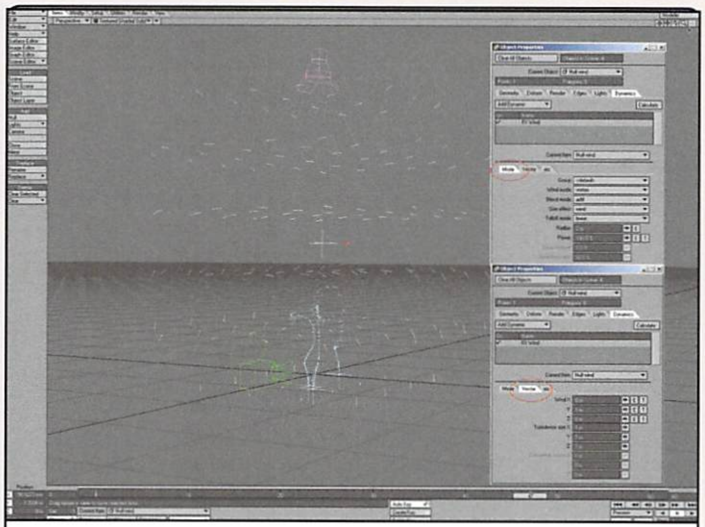
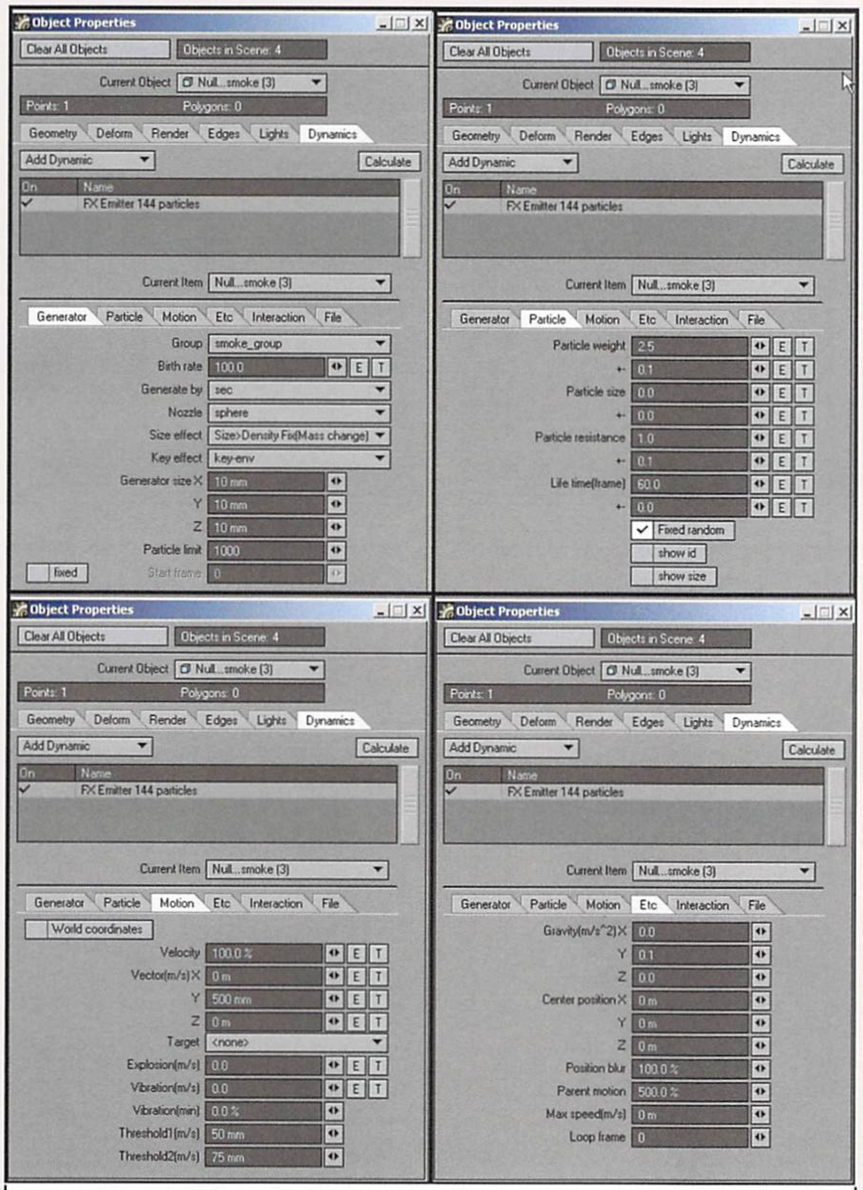


FIG. 37– WIND EFFECTOR PROPERTIES



Objects in Scene: 4
Current Object: Null...smoke (3)
Points: 1
Polygons: 0
✓ FX Emitter 144 particles
Current Item: Null...smoke (3)

Group: smoke_group
Birth rate: 100.0
Generate by: sec
Nozzle: sphere
Size effect: Size>Density Fix(Mass change)
Key effect: key-env
Generator size X: 10 mm
Y: 10 mm
Z: 10 mm
Particle limit: 1000
fixed
Start frame: 0

Particle weight: 2.5
+-: 0.1
Particle size: 0.0
+-: 0.0
Particle resistance: 1.0
+-: 0.1
Life time(frame): 60.0
+-: 0.0
✓ Fixed random
show id
show size

World coordinates
Velocity: 100.0 %
Vector(m/s) X: 0 m
Y: 500 mm
Z: 0 m
Target: <none>
Explosion(m/s): 0.0
Vibration(m/s): 0.0
Vibration(min): 0.0 %
Threshold1(m/s): 50 mm
Threshold2(m/s): 75 mm

Gravity(m/s^2) X: 0.0
Y: 0.1
Z: 0.0
Center position X: 0 m
Y: 0 m
Z: 0 m
Position blur: 100.0 %
Parent motion: 500.0 %
Max speed(m/s): 0 m
Loop frame: 0

FIG. 33 *(TOP LEFT)*, FIG 34 *(TOP RIGHT)*, FIG 35 *(BOTTOM LEFT)*, FIG 36 *(BOTTOM RIGHT)*

STEP 2

The next four examples are the settings used to create the action and behavior of the particles.

Figure 33–*Generator Paricle Properties*

Figure 34–*Particle Properties*

Figure 35–*Motion Paricle Properties*

Figure 36–*Additonal Paricle Properties*

STEP 3

Once the particles are created, a fourth null is created, but this time, instead of an emitter being added to it, a Wind Effector is added. The following shows the options chosen for the Wind Effector. *(Fig. 37)*

Step 4

To texture the particles to look like smoke Hypervoxels, first open the Effects panel *(Windows>Volumetrics and Fog Options).* Next, add Hypervoxels *(Add Volumetrics> HyperVoxels).* The initial settings are shown in figure 38.

Step 5

Under the Geometry tab, the Particle Size is set. After clicking the "T" button next to the Particle Size, a gradient is set. The options used are as shown in figures 39 - 41.

Step 6

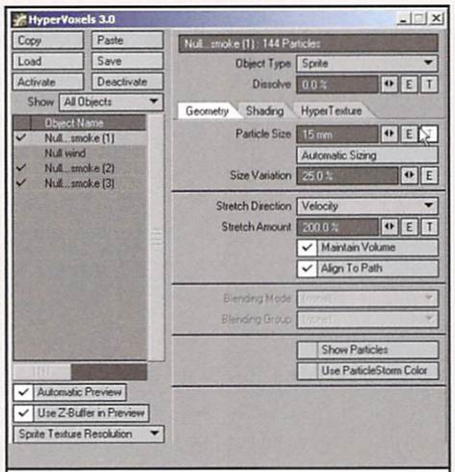
Next, the Shading tab's selected; this is where we get to texture the particles. *(Fig. 42)*

Step 7

Five bars are used to define the gradient, and the Input Parameter is set to Particle Age. The results of each bar is as shown in figures 43-47.

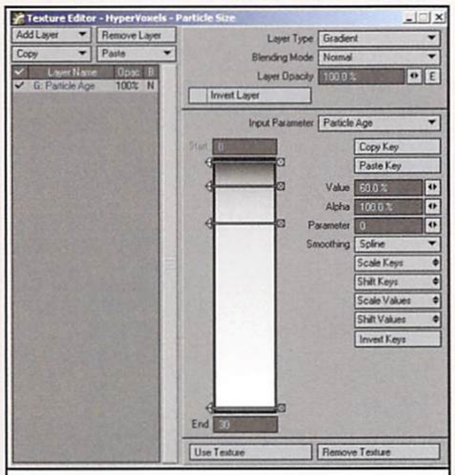
Step 8

Four bars are used to define the gradient, and the Input Parameter is set to Particle Age. The results of the Opacity gradient options are shown in figures 48-51.



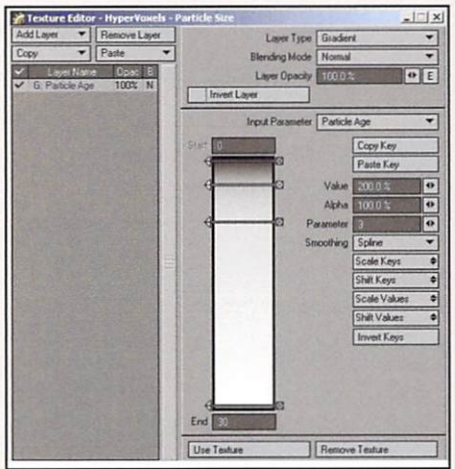
Null...smoke (1): 144 Particles
Show: All Objects
Object Name: ✓ Null...smoke (1); Null wind; ✓ Null...smoke (2); ✓ Null...smoke (3)
Object Type: Sprite
Dissolve: 0.0 %
Particle Size: 15 mm
Automatic Sizing
Size Variation: 25.0 %
Stretch Direction: Velocity
Stretch Amount: 200.0 %
✓ Maintain Volume
✓ Align To Path
Show Particles
Use ParticleStorm Color
✓ Automatic Preview
✓ Use Z-Buffer in Preview

FIG. 38– HYPERVOXEL PROPERTIES



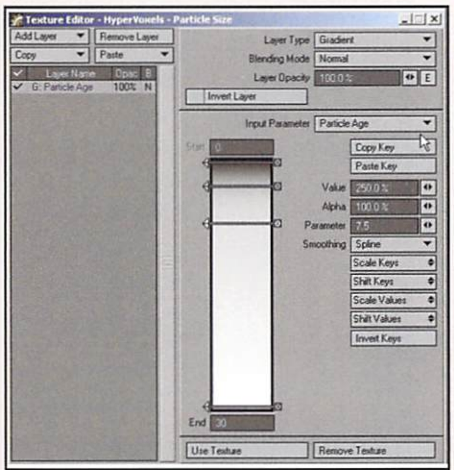
✓ G: Particle Age 100% N
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Input Parameter: Particle Age
Start: 0
Value: 60.0 %
Alpha: 100.0 %
Parameter: 0
Smoothing: Spline
End: 30

FIG. 39– TEXTURE SIZE OPTION



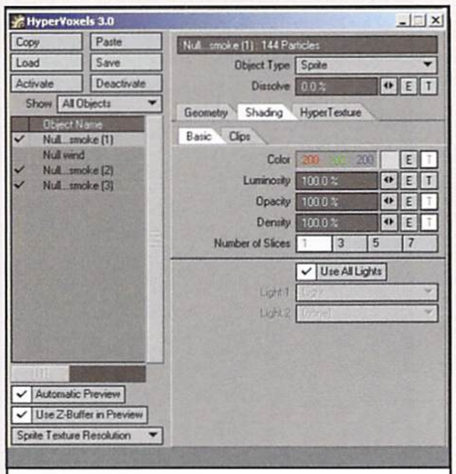
✓ G: Particle Age 100% N
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Input Parameter: Particle Age
Start: 0
Value: 200.0 %
Alpha: 100.0 %
Parameter: 3
Smoothing: Spline
End: 30

FIG. 40– TEXTURE SIZE OPTION



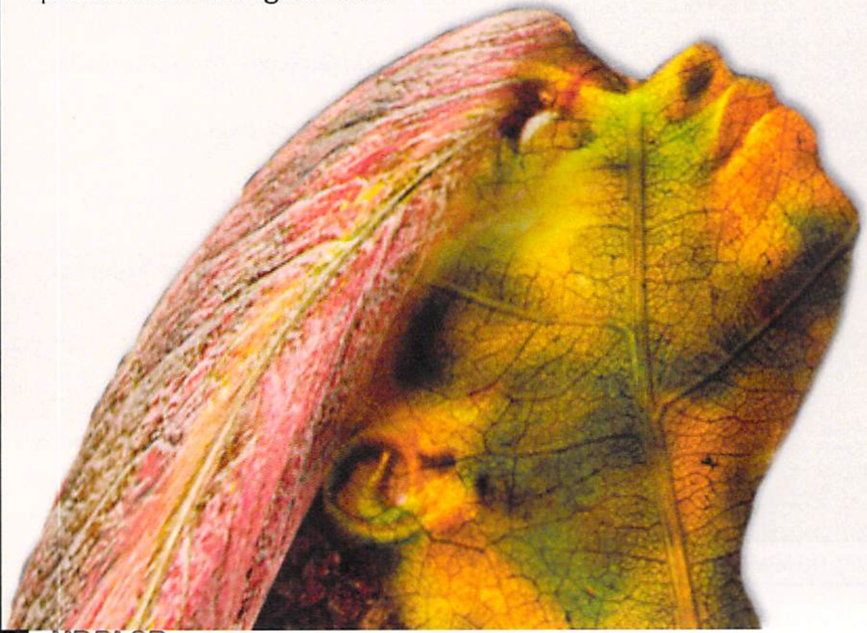
✓ G: Particle Age 100% N
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Input Parameter: Particle Age
Start: 0
Value: 250.0 %
Alpha: 100.0 %
Parameter: 7.5
Smoothing: Spline
End: 30

FIG. 41– TEXTURE SIZE OPTION



Null...smoke (1): 144 Particles
Show: All Objects
Object Name: ✓ Null...smoke (1); Null wind; ✓ Null...smoke (2); ✓ Null...smoke (3)
Object Type: Sprite
Dissolve: 0.0 %
Color: 200 200 200
Luminosity: 100.0 %
Opacity: 100.0 %
Density: 100.0 %
Number of Slices: 1 3 5 7
✓ Use All Lights
✓ Automatic Preview
✓ Use Z-Buffer in Preview

FIG. 42– SHADING OPTIONS



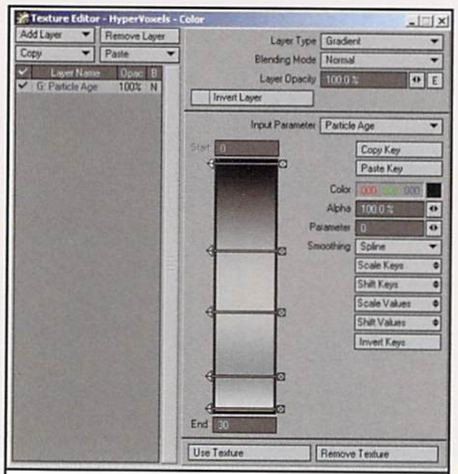

Texture Editor - HyperVoxels - Color
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Input Parameter: Particle Age
Start: 0
Color: 000 000 000
Alpha: 100.0 %
Parameter: 0
Smoothing: Spline
End: 30

FIG. 43– COLOR OPTIONS

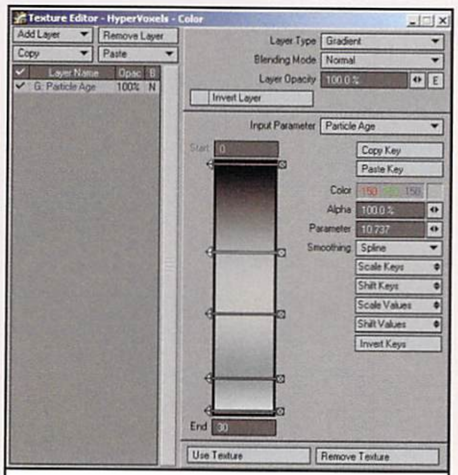

Texture Editor - HyperVoxels - Color
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Input Parameter: Particle Age
Start: 0
Color: 150 [illegible] 150
Alpha: 100.0 %
Parameter: 10.737
Smoothing: Spline
End: 30

FIG. 44– COLOR OPTIONS

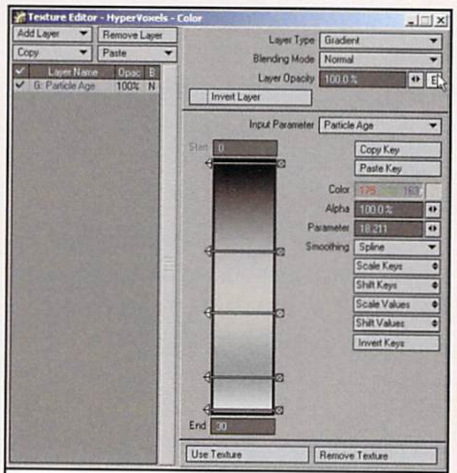

Texture Editor - HyperVoxels - Color
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Input Parameter: Particle Age
Start: 0
Color: 175 [illegible] 163
Alpha: 100.0 %
Parameter: 18.211
Smoothing: Spline
End: 30

FIG. 45– COLOR OPTIONS

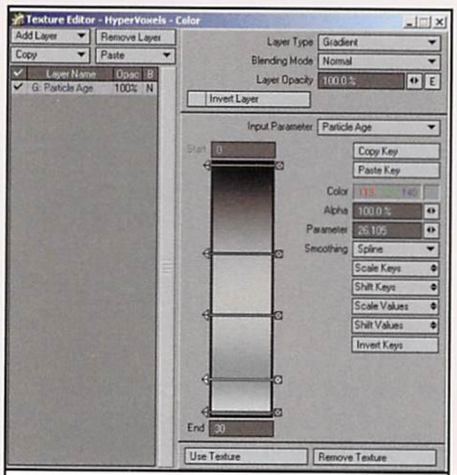

Texture Editor - HyperVoxels - Color
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Input Parameter: Particle Age
Start: 0
Color: 115 [illegible] 140
Alpha: 100.0 %
Parameter: 26.105
Smoothing: Spline
End: 30

FIG. 46– COLOR OPTIONS

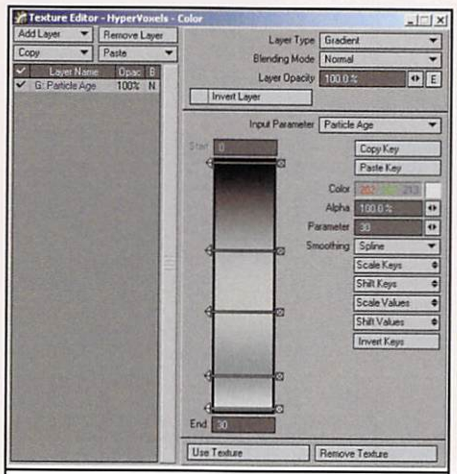

Texture Editor - HyperVoxels - Color
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Input Parameter: Particle Age
Start: 0
Color: 202 [illegible] 213
Alpha: 100.0 %
Parameter: 30
Smoothing: Spline
End: 30

FIG. 47– COLOR OPTIONS

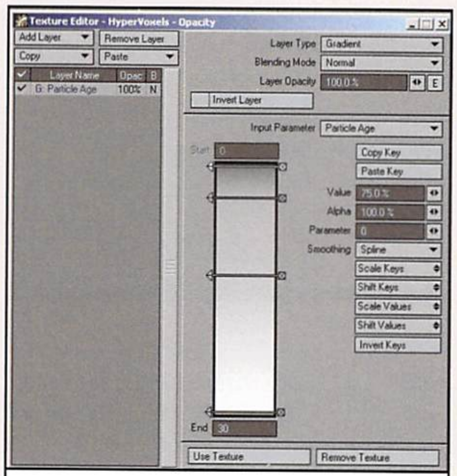

Texture Editor - HyperVoxels - Opacity
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Input Parameter: Particle Age
Start: 0
Value: 75.0 %
Alpha: 100.0 %
Parameter: 0
Smoothing: Spline
End: 30

FIG. 48– OPACITY OPTIONS

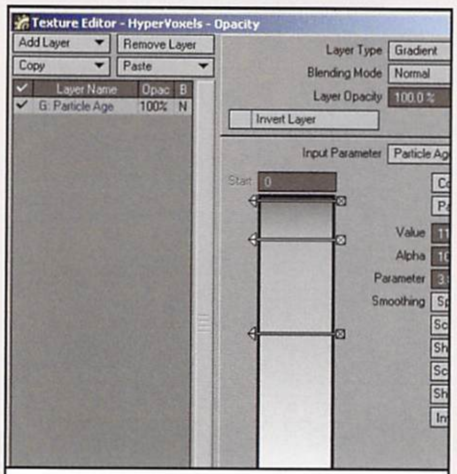

Texture Editor - HyperVoxels - Opacity
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Input Parameter: Particle Ag
Start: 0

FIG. 49–OPACITY OPTIONS

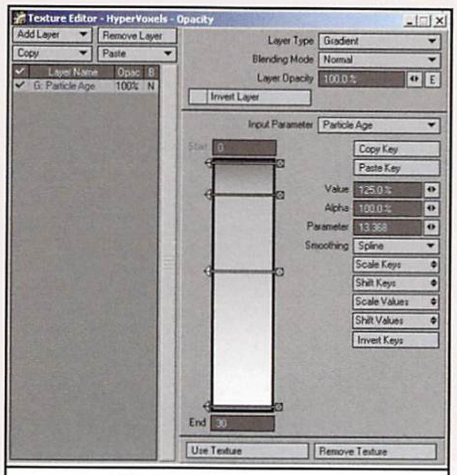

Texture Editor - HyperVoxels - Opacity
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Input Parameter: Particle Age
Start: 0
Value: 125.0 %
Alpha: 100.0 %
Parameter: 13.368
Smoothing: Spline
End: 30

FIG. 50– OPACITY OPTIONS

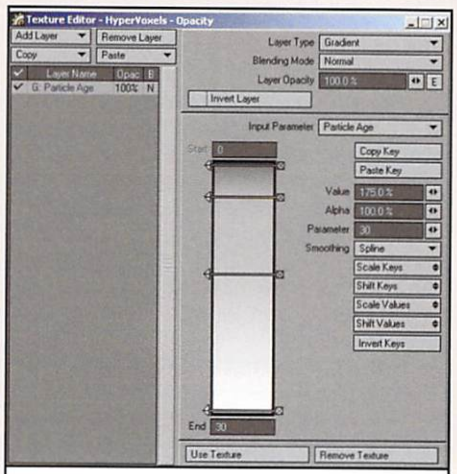

Texture Editor - HyperVoxels - Opacity
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Input Parameter: Particle Age
Start: 0
Value: 175.0 %
Alpha: 100.0 %
Parameter: 30
Smoothing: Spline
End: 30

FIG. 51– OPACITY OPTIONS

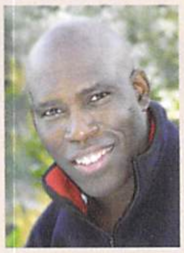
STEP 9

Four bars are used to define the gradient; the Input Parameter is set to Particle Age. The results of the Density gradient options are shown in figures 52-55.

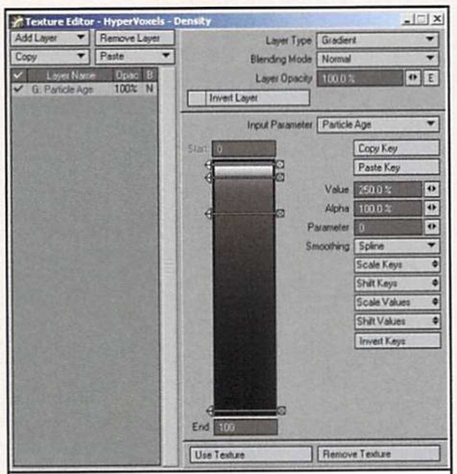
STEP 10

Finally, the Hyper Texture tab is selected to give the fog some physical character. The options are shown in figure 56.

There it is. Once rendered, the final image is saved as a TIFF and brought into Photoshop for editing. I hope you enjoyed this article. If you would like to see more of my work, please visit my site at *www.chromeallusion.com.*

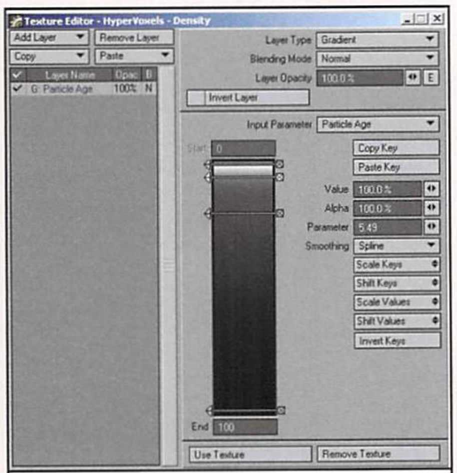


STEPHEN BURNS HAS DISCOVERED THE SAME PASSION FOR THE DIGITAL MEDIUM AS HE HAS FOR PHOTOGRAPHY AS AN ART FORM. HIS BACKGROUND BEGAN AS A PHOTOGRAPHER 22 YEARS AGO AND IN TIME, PROGRESSED TOWARD THE DIGITAL MEDIUM. STEPHEN HAS BEEN A CORPORATE INSTRUCTOR AND LECTURER IN THE APPLICATION OF DIGITAL ART AND DESIGN FOR THE PAST 8 1/2 YEARS. HE HAS BEEN EXHIBITING DIGITAL FINE ART INTERNATIONALLY AT GALLERIES SUCH AS DURBAN ART MUSEUM IN SOUTH AFRICA, CITIZENS GALLERY IN YOKAHAMA, JAPAN, AND CECUT MUSEUM OF MEXICO TO NAME A FEW. PART OF HIS EXHIBITING WON HIM 1ST PLACE IN THE PRESTIGIOUS SEYBOLD INTERNATIONAL DIGITAL ARTS CONTEST. OWNER OF *BURNS DESIGN*, HE TEACHES DIGITAL MANIPULATION WORKSHOPS IN THE SAN DIEGO AREA AND NATIONWIDE. HIS TEACHING STYLE COMES FROM HIS ABILITY TO SHARE AN UNDERSTANDING OF PHOTOSHOP SO THAT THE STUDENT HAS THE ABILITY TO INTUITIVELY APPLY IT TO HIS/HERS CREATIONS.



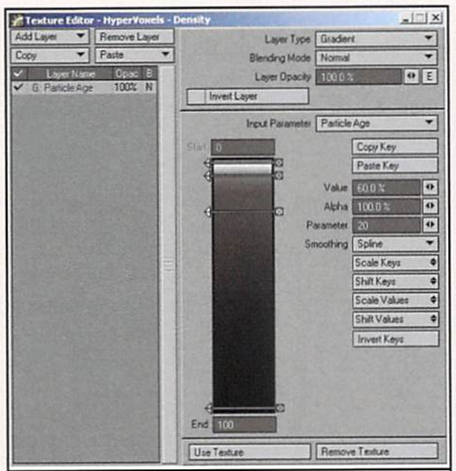
Texture Editor - HyperVoxels - Density
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Input Parameter: Particle Age
Start: 0
Value: 250.0 %
Alpha: 100.0 %
Parameter: 0
Smoothing: Spline
End: 100

FIG. 52– DENSITY OPTIONS



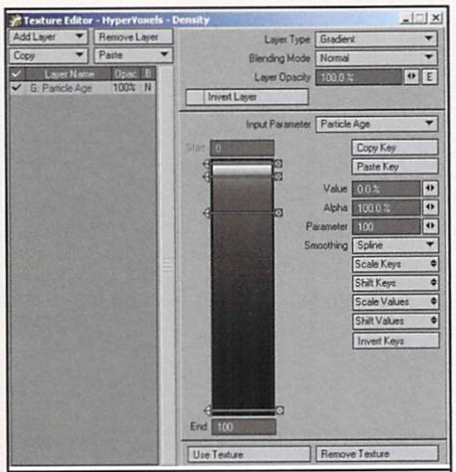
Texture Editor - HyperVoxels - Density
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Input Parameter: Particle Age
Start: 0
Value: 100.0 %
Alpha: 100.0 %
Parameter: 5.49
Smoothing: Spline
End: 100

FIG. 53– DENSITY OPTIONS



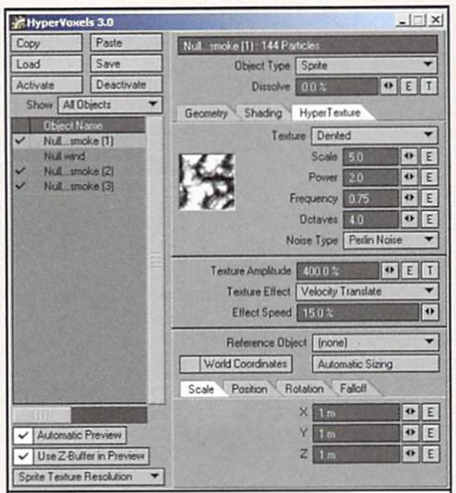
Texture Editor - HyperVoxels - Density
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Input Parameter: Particle Age
Start: 0
Value: 60.0 %
Alpha: 100.0 %
Parameter: 20
Smoothing: Spline
End: 100

FIG. 54– DENSITY OPTIONS



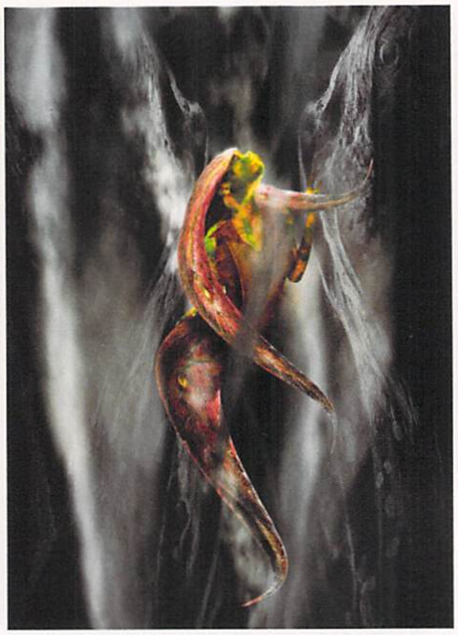
Texture Editor - HyperVoxels - Density
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Input Parameter: Particle Age
Start: 0
Value: 0.0 %
Alpha: 100.0 %
Parameter: 100
Smoothing: Spline
End: 100

FIG. 55–DENSITY OPTIONS



HyperVoxels 3.0
Null_smoke (1) : 144 Particles
Object Type: Sprite
Dissolve: 0.0 %
Texture: Dented
Scale: 5.0
Power: 2.0
Frequency: 0.75
Octaves: 4.0
Noise Type: Perlin Noise
Texture Amplitude: 400.0 %
Texture Effect: Velocity Translate
Effect Speed: 15.0 %
Reference Object: (none)
X: 1 m
Y: 1 m
Z: 1 m
Automatic Preview
Use Z-Buffer in Preview

FIG. 56– DENSITY OPTIONS



BY NICHOLAS BOUGHEN

A Newfound Philosophy on 3D and Life

The world of electronic communication has become populated with faceless personalities hiding behind invented names. No doubt you've seen this many times; people who feel they can get away with anything because they're using a handle instead of a real name. It's easier, I guess, to speak your mind without fear of repercussion when nobody knows who you are. Minds are less inhibited, and so are our expressions. There is a good side to this; good in that freedom to speak openly and honestly without fear of repercussion is beneficial to us all. But there's a nasty side to this freedom as well. Some people use aliases and email addresses as hiding places from where they can assault anybody. Common courtesies are trampled and forgotten. Libel runs pretty rampant. Email addresses come and go. It's a cheap, immature sport. Most of us who have been using email since 300 baud modems were invented, who remember when the Internet consisted of only Bulletin Boards and Gopher, just accept this as a fact of life and move on. It's a kind of wisdom, I suppose. But we become guarded over time, maintaining a certain distance with our Internet and email companions. Unless you are corresponding with old friends, you never really know, after all, who is at the other end. You don't share everything with your Internet acquaintances. At work, you maintain a professional demeanor and distance.

When I made plans to attend SIGGRAPH in Los Angeles this year, it occurred to me that I would be meeting some of those long-time Internet acquaintances. CG and compositing artists, studio owners, software engineers and developers, sales people, customer service reps, publishers, authors, and many others; these were people I had talked to on the phone, corresponded with through email or instant messaging or on CG forums for years, yet I had never met most of them. I was really looking forward to that.

I also knew I would be meeting many new people — some whose names are well known, who are the gurus and forefathers and groundbreakers of our profession; others who, like me, are the trench warriors of visual effects, largely unknown except within the small circle of artists, coordinators and software engineers that encompass our daily lives. And so, when I arrived in L.A. on Monday night, I dropped off my luggage and headed straight for a party to start meeting people.

NICK AND LARRY SCHULTZ SAYING *CHEESE*

The funny thing ... I mean the really funny thing ... is that I wasn't really prepared for what happened next. I had imagined that I would meet these people, finally attaching a face to the name. I'd shake their hands and say "great to finally meet you" and maybe we'd talk about our work or something. What I wasn't really expecting was that I'd sit down and start chatting and finding out that these people, formerly only email ghosts, had lives and (believe it or not) interests outside of 3D, and families and life experiences and funny stories about being a teenager and everything ... just like normal people. It is painfully obvious in hindsight, but it is funny, too, because it hadn't really occurred to me before, I was so disconnected and buried in my own tiny reality behind a big monitor in the bottom floor of a company somewhere far away from here.

NICK HANGING OUT WITH DEUCE BENNETT

That was the beginning of a transformation for me. I began to realize that I had kept these professional acquaintances at a respectful, professional distance for so long, it had simply become habit. I had missed out on the fact that they are all smart, interesting, living people — people who believe passionately in what they're doing — people who take the bug reports and feature requests and really try to make sense out of them to the best advantage of all the users. Other artists like me who pound the keyboard daily, begging the software to perform, creating miracles out of pixels and electrons and committing not just hours and days, but blood, sweat and tears to every pixel of every frame.

So the next day, when I first went to the exhibition floor, I committed myself to seeking out everyone I could and taking the time, if possible, to sit down and get to know them a bit. I didn't get to everyone on my list, but I managed to spend time with some really great folks and made friends out of some of those email addresses and contact numbers.

Over lunch, I met and talked to some great artists from CG houses all over the place. I just grabbed my sandwich, took an empty chair at a table, and before I knew it found myself involved in debates over techniques, software and hardware with other artists who, like me, seldom received the opportunity to share ideas with people from other companies. *"What software do you use for X task?" "Why do you use it?" "You should really try Y software because...." "Z software SUCKS!"* Yeah, there were many *"Z software SUCKS!"-- "No, it doesn't"* discussions.

I got to have a long talk with Steve Hill from 2D3, whom I had plagued with feature requests and bug reports for years. Steve, like so many others out there, is not just a technical support specialist. He's a really smart man who's strongly dedicated to his product and his user base. I spent an evening listening to the childhood mayhem stories of Larry Schultz, who has been a LightWave artist and author for ... well ... way longer than I have. I hung out with Timothy Albee, author and maker of *Kaze: Ghost Warrior*, and I happened across the inimitable Steve Worley, creator of such software marvels as Sasquatch and FPrime. I ran into them at a party. Tim Albee and I sat on either side of Steve, like bookends on the way back to the hotel and, a little drunk, grilled him about his software. I mean, how many times are you going to have one of your favorite software engineers as captive an audience as that? I hung out a lot with the Wordware folks, Wes Beckwith and Steve Warner and the gang. I also got to meet and talk to a bunch of the NewTek crew ... you know, the people LightWave artists complain to when software doesn't behave as expected? It was great fun cornering Dr. Andrew Cross, *THE* ear of LightWave software development, and suggesting a load of "really important" feature requests. I hung out with William "Proton" Vaughan and Deuce Bennett and Ben Vost and others during those days, getting to know WHO they are, not just what they are. What a bunch of great guys.

TALKING WITH THE *BOUJOU* BOYS

I actually got to meet and talk with Dick van Dyke, who has been a LightWave artist since the Amiga days. He was just sitting there in the audience at the NewTek stage, so I wandered over and introduced myself, eventually sitting down with him once he struck up a conversation about LightWave [8]. It was a little surreal at first, listening to him offer his thoughts on the new software, and responding with my own remarks and opinions. I mean, who ever knew Dick van Dyke was just another CG artist like the rest of us? Well, OK, not exactly like the rest of us. We can't all afford a 42" plasma monitor. But what a pleasant fellow he turned out to be. That one experience, more than any other, really served to crystallize a feeling that had been growing all week — that all of us are just people working hard to do our best, that we all feel the joy of our successes as poignantly as the pain of our failures, even if we're giant movie stars, or Oscar winners, or coding geniuses, or brilliant artists, or famous personalities, or just the quiet artists in the corner, pounding out our CG workload each day.

BONDING WITH THE *WORDWARE* GANG

Work will be a little different for me now. Not that the job has changed. I still have to create impossible imagery using hardware and software pushed right to the edge, and often over it. My software or hardware will still fail me from time to time. That's the price we pay for working at the bleeding edge. But it's easier to take, now that I know the people at the other end are working just as hard as I am, just as long every day, and that they're just as driven, detail-oriented and obsessed with perfection.

What a difference a face makes.

It is now the evening before I have to submit this piece to editorial. Disaster has struck, bringing us at Rainmaker an even more poignant reminder that we should each do everything we can to know and enjoy those around us, to benefit from their experience and personality.

Rainmaker lost its Skipper, President and CEO Bob Scarabelli, this morning. Bob died of natural causes while biking near his home on Vancouver's North Shore.

It's easy to think that a loss like this might not have much impact on the day-to-day operations of a company as large as Rainmaker — easy if you didn't know Bob. I once sat in a general meeting during the industry slump early in 2002 and watched Bob choking up as he announced that Rainmaker, for the first time in its history, was forced to implement job cuts. It ended up only being a couple of jobs, but to Bob, it was like sending members of his family out into the cold.

Bob was the kind of guy who would appear at my desk from time to time just to check out what I was working on. It seems the company was never too large for him to spend at least a little time talking with everyone. He was a busy, driven guy, but never failed to hear you out if you needed his ear. Bob was like our Dad.

He set the tone for the company, as a good leader should, always making known that people were his highest priority. He knew that the highest quality can only be achieved by the best people under the best circumstances. When Bob regarded Rainmaker, he knew it could be such a place and he strived every day to make it so.

NICK WITH ALICE & CHARLES EDGIN, PUBISHERS OF HDRI 3D

The industry has lost a visionary, a man of passion and strength, and our leader.

And now, when I look around the room where I work, where I share my days with a powerhouse of talent, I'm reminded that any of us can switch off without notice. We're all human, after all, and each of us is worth getting to know.

We all miss Bob; surely not more than his wife and two young children do, but he was a huge presence among us, and we find ourselves now facing a large empty place where he was.

NICHOLAS BOUGHEN IS A SENIOR VISUAL EFFECTS ANIMATOR AND LEAD LIGHTING ARTIST AT RAINMAKER IN VANCOUVER, BRITISH COLUMBIA, CANADA. HIS CREDITS INCLUDE *I, ROBOT, GARFIELD, GOODBOY!, STARGATE SG-1, STARGATE: ATLANTIS, DEAD LIKE ME, AFTERSHOCK:EARTHQUAKE IN NEW YORK, VOYAGE OF THE UNICORN, SNOW WHITE*, AND OTHERS.

NICHOLAS IS THE AUTHOR OF THE BEST SELLING LIGHTWAVE BOOK *LIGHTWAVE 3D 7.5 LIGHTING*, AND ITS NEW 2ND EDITION *LIGHTWAVE 8 LIGHTING*. HE HAS RECENTLY RECEIVED AN EMMY NOMINATION FOR OUTSTANDING SPECIAL VISUAL EFFECTS IN A SERIES FOR HIS WORK ON THE TELEVISION SERIES, *DEAD LIKE ME*.

BY EDDIE ALCAZAR

FORE!!!!

MODELING AND TEXTURING A PHOTOREALISTIC GAME MODEL QUICKLY AND EFFICIENTLY

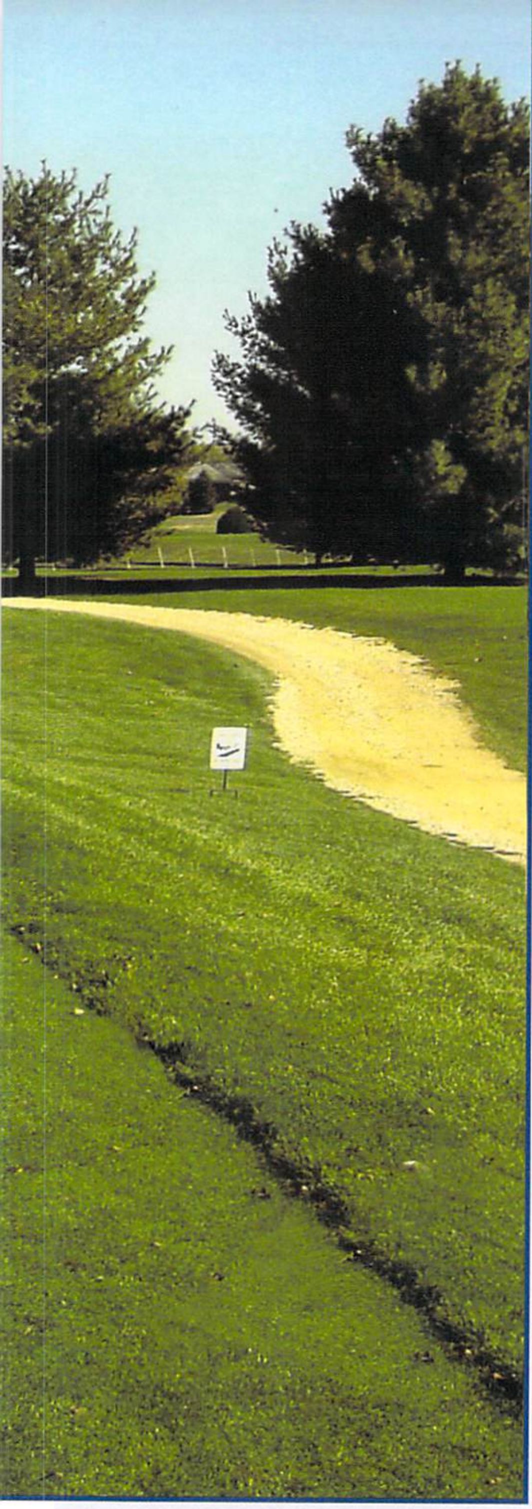
This tutorial will take you through the process of building and texturing a game model in Maya. It will show you that with the help of a few handy plug-ins and a digital cam or good reference, it can be done in a fairly quick amount of time. I will go into more detail with the head since it's what people tend to judge more closely. Then you should be familiar with the tools and be able to follow the creation of the remaining parts with ease. This tutorial uses references I've collected, but I keep the descriptions open enough so that you're not limited and you should be able to follow no matter what human ref you use.

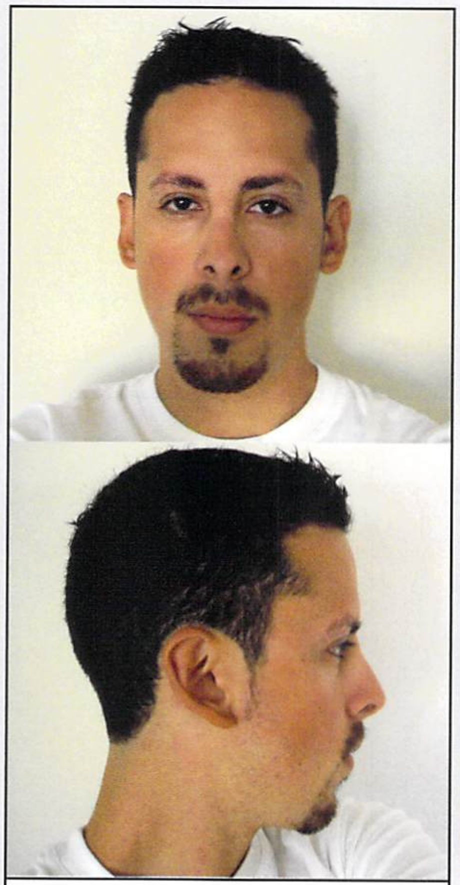
1 Go to Highend3d.com or use a search engine to download: Ultra Image planes, CPS, and MJ Poly tools. Install.

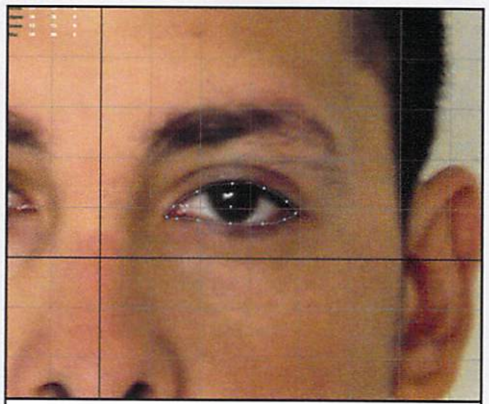
2 Gather as much reference material as possible. I will be using my mug shot for the head ref. *(See Ref Step 2 Next page)*

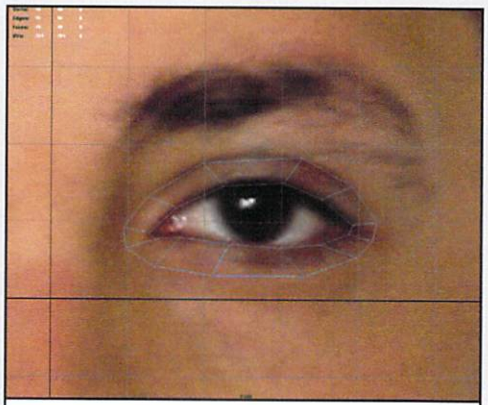
3 Make sure the front and side are scaled correctly; an easy way to do this is with rulers in Photoshop. Most of the time you will run into problems with camera distortion, so the more images of the ref you have from different perspectives the better. Save each separately.

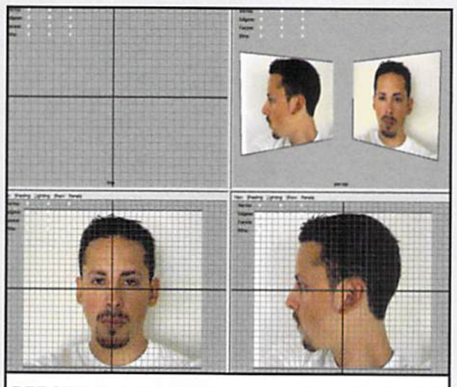
4 Next, open up Maya and execute the Ultra IP plug-in. Create and load in the images accordingly and press the Create Front and Side Camera tab. Make sure to move the planes out of the way by setting the Z coordinates in the Ultra Ip menu to Front -20.000, and the Side Camera to -20.000. I also prefer to uncheck the grid in the Show menu

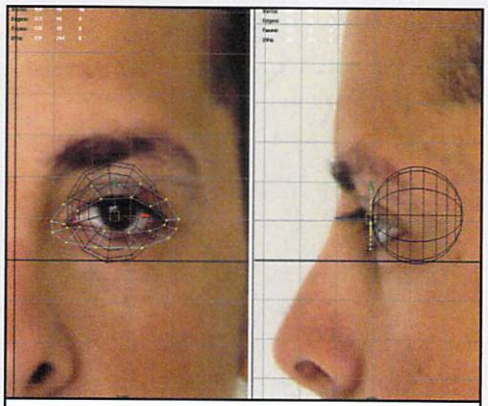



REF STEP 2– HUMAN HEAD REFERENCE IMAGES


MODELING STEP 3– START BY OUTLINING THE EYE


MODELING STEP 4– EXTRUDE EDGES


REF STEP 4– ULTRA IP PLUG-IN WITH HEAD IMAGES


MODELING STEP 6– POSITION EYE AS SHOWN

above the perspective window. Now we are all set up to model the head. *(See Ref Step 4)*

HEAD: MODELING

1 Hit spacebar to maximize the front window.

2 Go to *Polygons > Create Polygon* tool

3 I will start by outlining the eye. Twelve sides should be enough.

4 Next, right-click on the geometry and enter Edge mode. Select all the edges and go to *Edit Polygons > Extrude Edge.* You will see your Manipulator Tool pop up. Left-click on the small, light blue circle (this will move the tool to the center of your geometry). Now you can left-click the center and scale the extrusion out. Right-click to enter Vertex mode and move verts around so it looks similar to the image.

5 We need to make an eyeball. *Create > Polygon Primitives > Sphere Options >* enter 11 for subdivisions around axis and height.

6 Rotate 90 on the X-axis and position it accordingly, also moving the outer eye geometry to accompany it. (I usually put the eyeball in a separate layer to avoid selecting it).

7 Next, we will use MJ Poly Tools ELS (edge loop split). Select a vertical edge in the outer eye geometry and *MJ Poly Tools > Edge Loop Split.* Another ring around your geometry should have

appeared; you can control the position by changing the Slide channel in the right channel box. Slide it closer to the center.

8 Select the center poly and push it in towards the eyeball.

9 Split the outer eye again and start pushing and pulling the outer verts according to the ref image. Make sure you keep the edges as evenly spaced as possible. This will benefit you with deformations later on. Go back and forth between the views, making sure each vert is placed properly. Keep in mind that it's very rare to find perfectly matching front and side images, so you will have to eyeball a lot along the way. As you can see on my model, the geometry fits snug in front but not as close in the side view; as long as you're close it will be OK. I think it's more important to use the front as a guide, since you will be using the same front image for the texture.

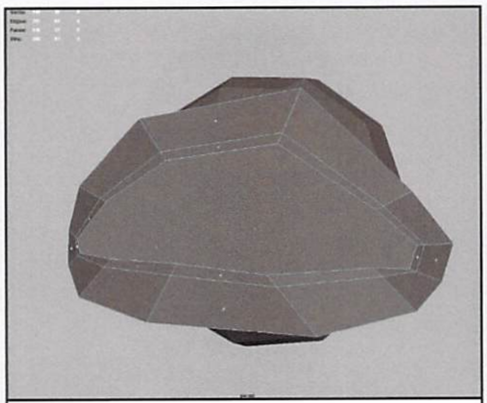
10 Select the bottom five edges and extrude down. Sculpt. Do the same with the upper four edges and for the remaining three, extrude and snap to center.

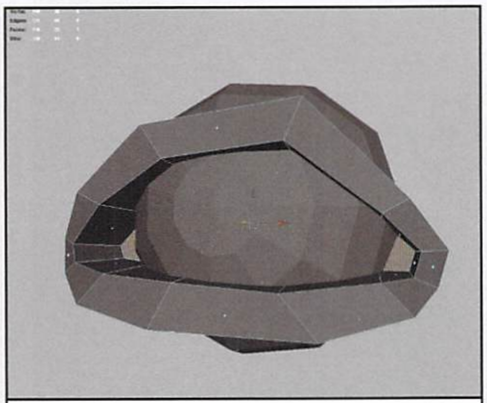
11 Merge the two verts together on the right.

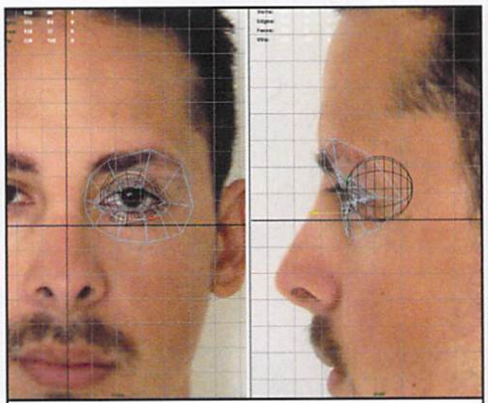
12 Extrude the upper and lower edge of the bridge of the nose and merge the loose verts.

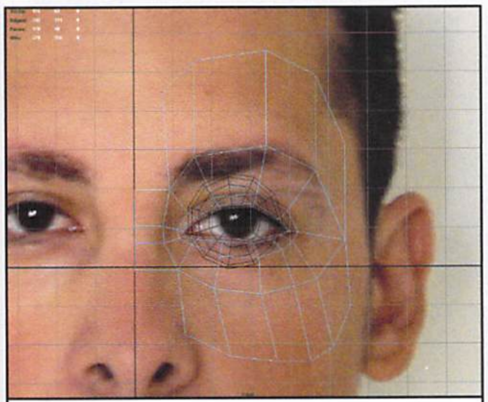
13 Sculpt on each of the views.

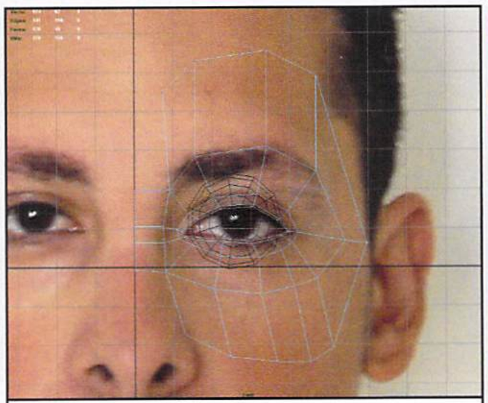
14 Next, we will do the mouth. Same as the eye, we will use the Create Poly tool. (note: CPS has a handy tool window that makes it easier to select tools) Make a 10-edge outline of the mouth.

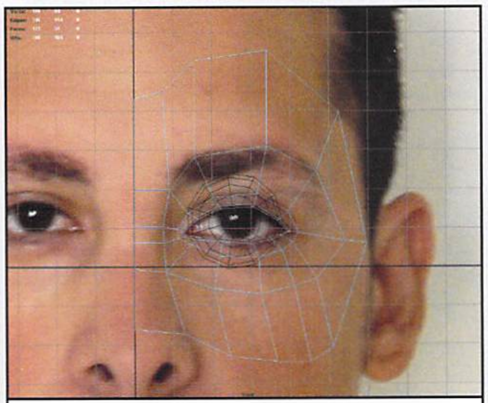

MODELING STEP 7– EDGE LOOP SPLIT

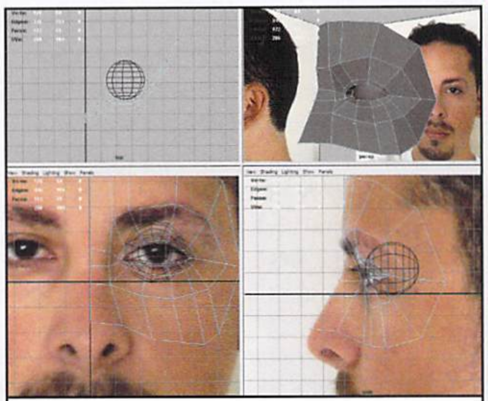

MODELING STEP 8– PUSH CENTER POLY IN

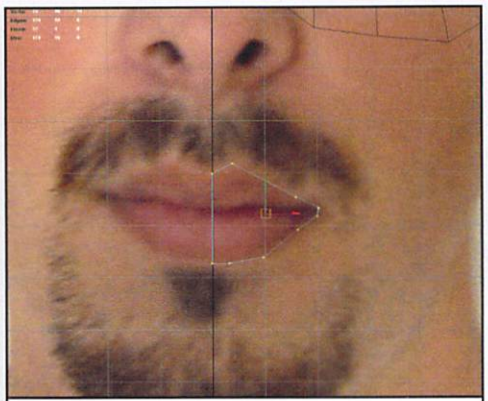

MODELING STEP 9– MATCH GEOMETRY TO REF IMAGE

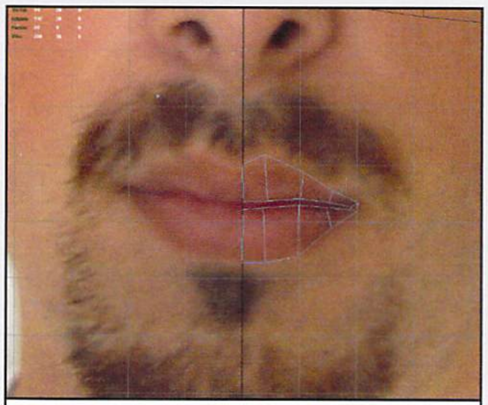

MODELING STEP 10– SCULPT THE EDGES

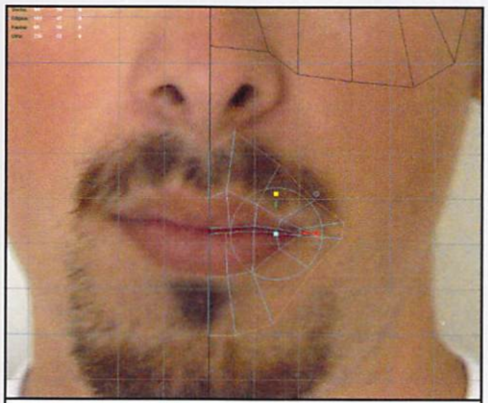

MODELING STEP 11– MERGE THE TWO VERTS ON RIGHT

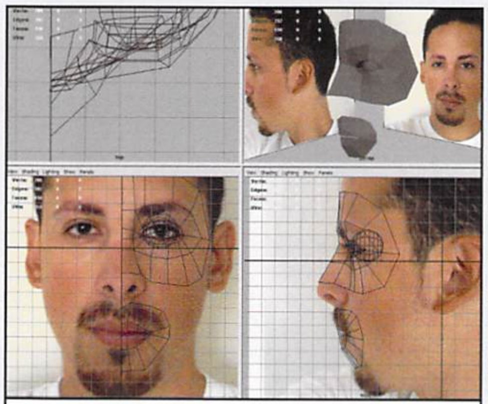

MODELING STEP 12– EXTRUDE AND MERGE

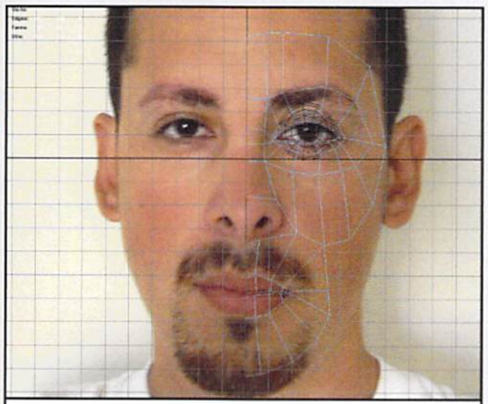

MODELING STEP 13– SCULPT ON EACH VIEW

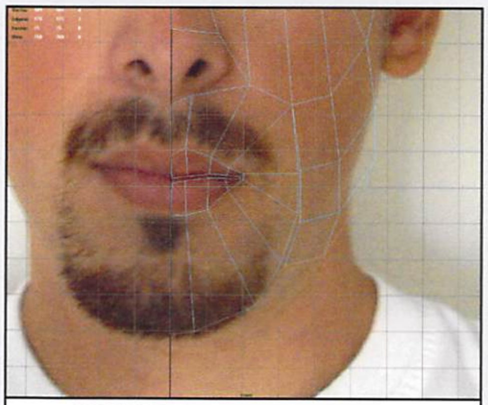

MODELING STEP 14– CREATE OUTLINE OF MOUTH

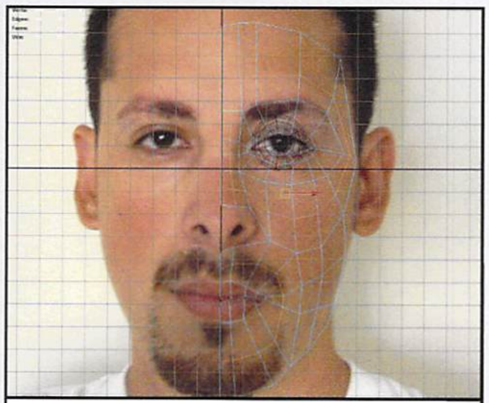

MODELING STEP 15– EXTRUDE & SCULPT MOUTH EDGES

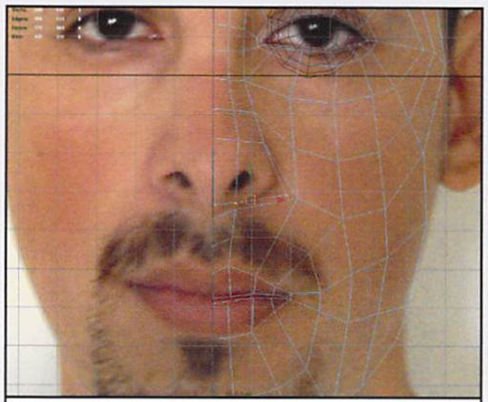

MODELING STEP 16– EXTRUDE OUTER RIM OF MOUTH

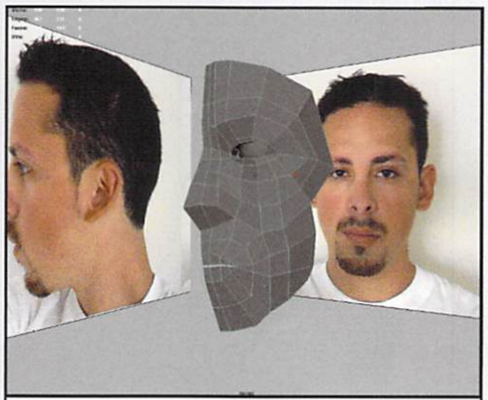

MODELING STEP 17– SCULPT ALL VIEWS


MODELING STEP 19– MERGE THE UPPER LIP VERTS


MODELING STEP 20– EXTRUDE CHEEKS


MODELING STEP 22– ADD ADDITIONAL RINGS WITH *ELS*


MODELING STEP 23– ETRUDE AND MERGE NOSE


MODELING STEP 24– YOURS SHOULD LOOK LIKE THIS

15 Select all edges and extrude in. Sculpt and delete the face closest to grid since we are going to mirror it later on. Leave the mouth slightly open so you can easily tweak if needed.

16 Select the outer rim of the mouth and extrude.

17 Sculpt in all views.

18 Select the eye socket and mouth piece and *Polygons > Combine*.

19 Merge the upper lip verts as shown in the image.

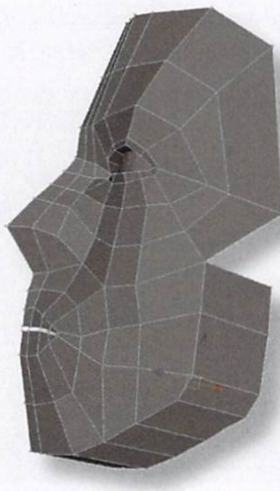
20 Extrude the two cheek edges down, and then extrude again two more times for each loop.

21 Merge the closest verts. Sculpt.

22 Select a vertical edge above the lips and ELS to add another ring. Do the same to the upper cheek and nose section. Sculpt.

23 Extrude the upper part of the nose to the bottom and merge.

24 So far you should have something similar to the image in terms of structure.



25 Next, extrude the jaw edges.

26 Merge the loose verts.

27 There are two edges on the forehead that are too close together; to conserve polygons and even out the structure, we are going to delete them. Then, merge the loose verts to opposite ends, creating two triangles. (Note: you want to limit triangles on your mesh as much as possible, since there are no deformations on the forehead, besides, maybe the eyebrows moving up it should be OK.)

28 Now we can extrude the rest of the head.

29 Sculpt.

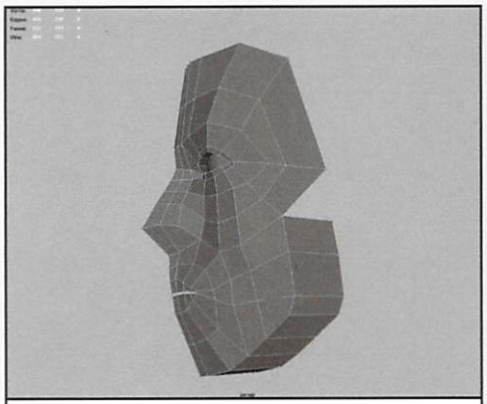
30 Extrude the neck and merge verts.

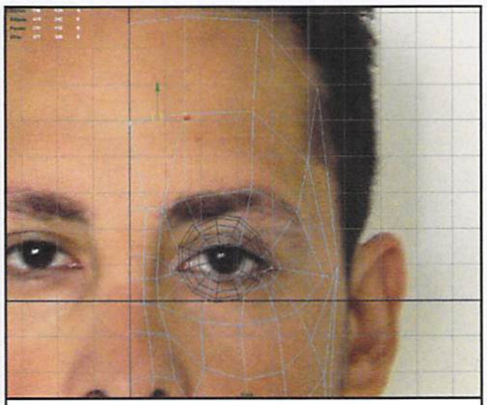
31 Select an edge in the empty spot where the ear should be and *Edit Polygons > Split* fill hole.

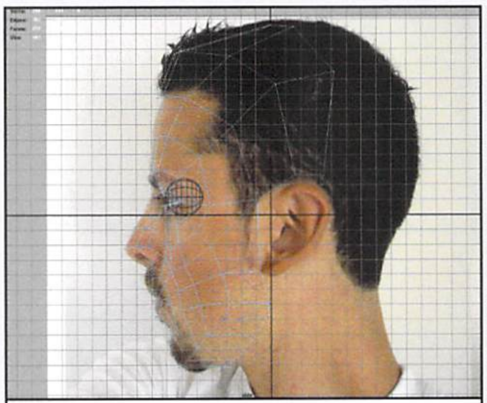
32 Split polys like the image using the *Edit Polygons > Split Polygons* tool.

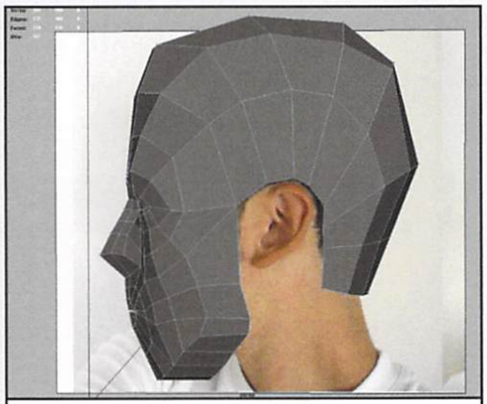
33 Extrude faces in the center of the ear.

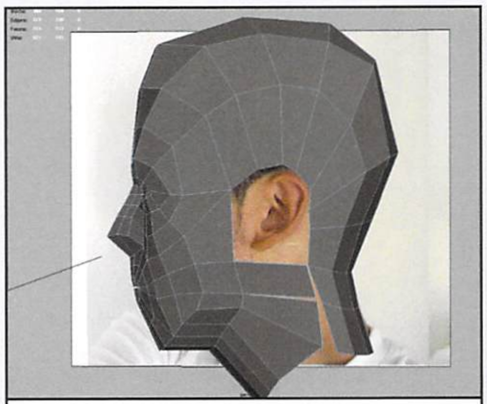
34 Split and extrude different parts according to the specific details of your character's ear; remember that it doesn't have to be a very high poly, basically just the outer rim and cavity should work. Sculpt.

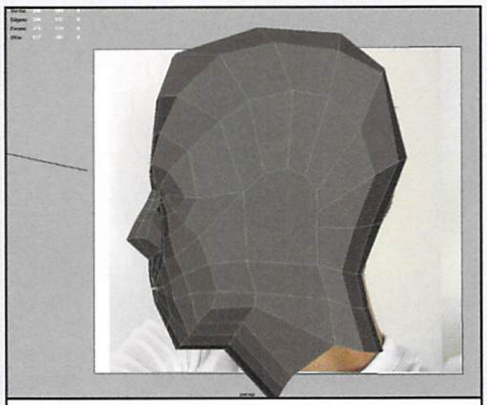

MODELING STEP 25– EXTRUDE THE JAWS

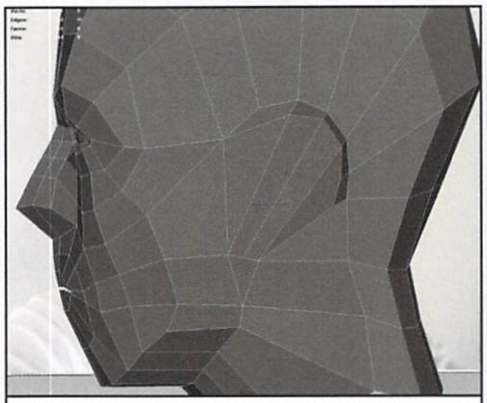

MODELING STEP 27– LIMIT TRIANGLES ON YOUR MESH

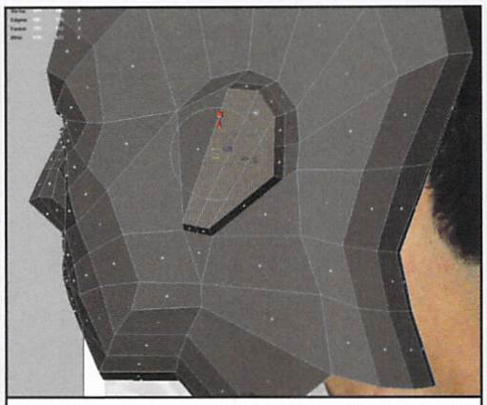

MODELING STEP 28– EXTRUDE THE REST OF THE HEAD

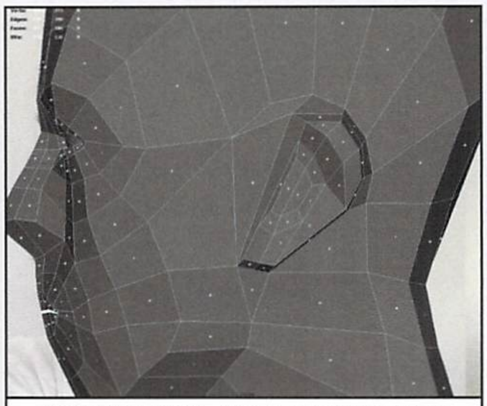

MODELING STEP 29– SCULPT THE HEAD

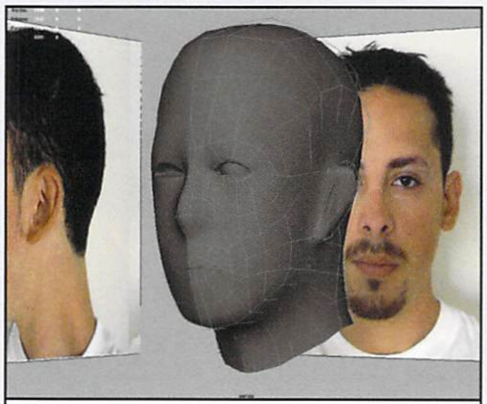

MODELING STEP 30– EXTRUDE NECK & MERGE VERTS


MODELING STEP 31– FILL THE HOLE WHERE THE EAR IS

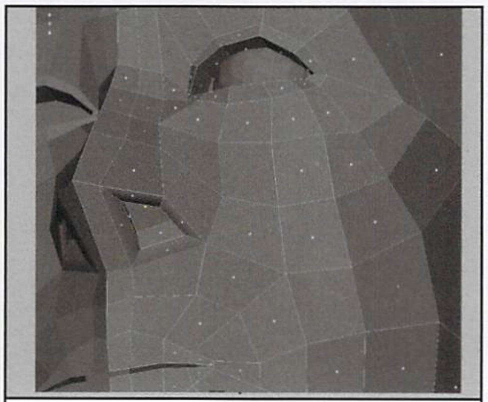

MODELING STEP 32– SPLIT POLYS AS SHOWN HERE


MODELING STEP 33– EXTRUDE FACES IN CENTER OF EAR


MODELING STEP 34– SCULPT BASIC EAR SHAPE



MODELING STEP 35–SOFTEN THE POLYGONS



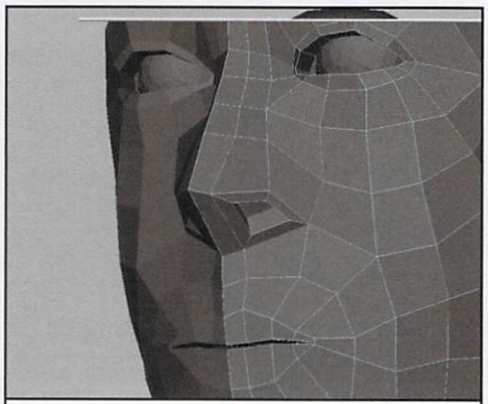
MODELING STEP 36–EXTRUDE IN THE NOSE

35 OK, now that we have a well-structured base, we can mirror and add details. Make sure all your verts at the half are centered on the grid. Open the CPS tab and check the mirror and stitch option and choose – X for the mirror direction. This will automatically smooth your model. Open the CPS Ctrl tab and hit the minus sign, and you're back to the unsmoothed version. You may also want to *Edit Polygons > Normals > Soften/Harden > All Soft.*

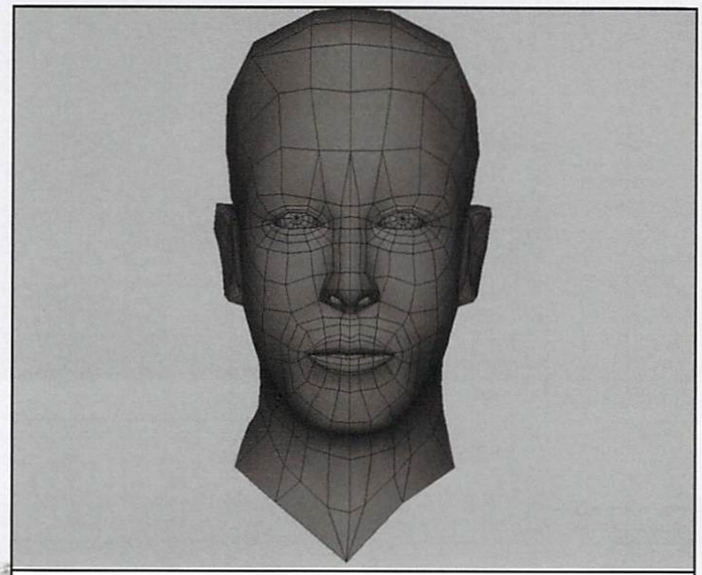
36 Extrude in the nose.

37 Do the same for the lips.

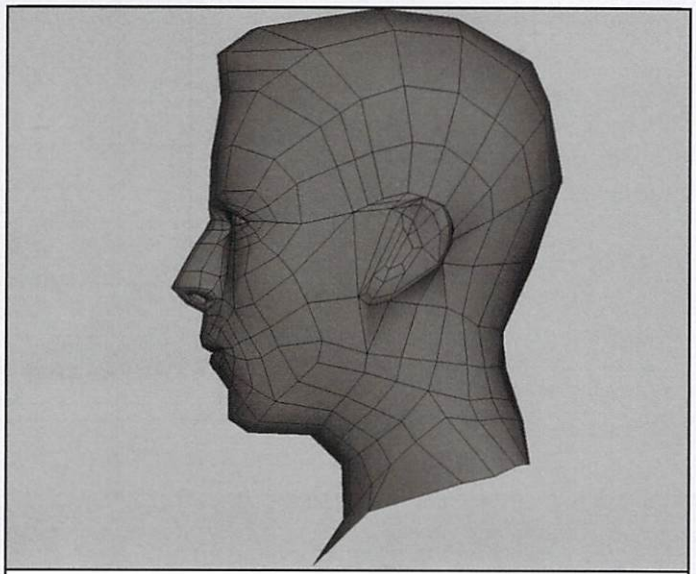
38 From this point on, it's a matter of sculpting, pushing, and pulling the verts as needed, adding a few loops and looking closely at your gathered reference. This is my final head model before I move on to texturing.



MODELING STEP 37– EXTRUDE IN THE LIPS



FRONT VIEW OF THE FINAL HEAD BEFORE TEXTURING



SIDE VIEW OF THE FINAL HEAD BEFORE TEXTURING

HEAD: TEXTURING

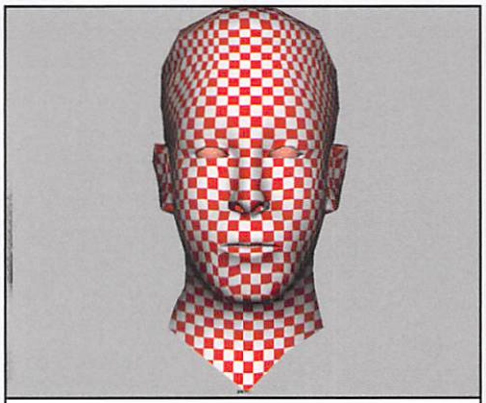
1 First delete the CPS from your mesh, go to the CPS CTRL tab and press the icon that's the 3rd down on the right, the icon with the scissors. When it asks you to delete base geometry, say yes.

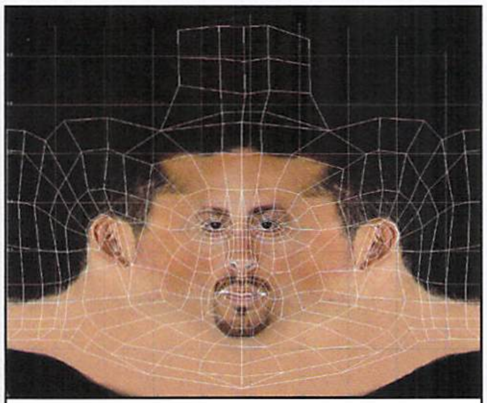
2 For the Head, I will be using *Edit Polygons > Cylindrical Mapping*. Make sure you assign a checkerboard texture to your model to fix and prevent any stretching.

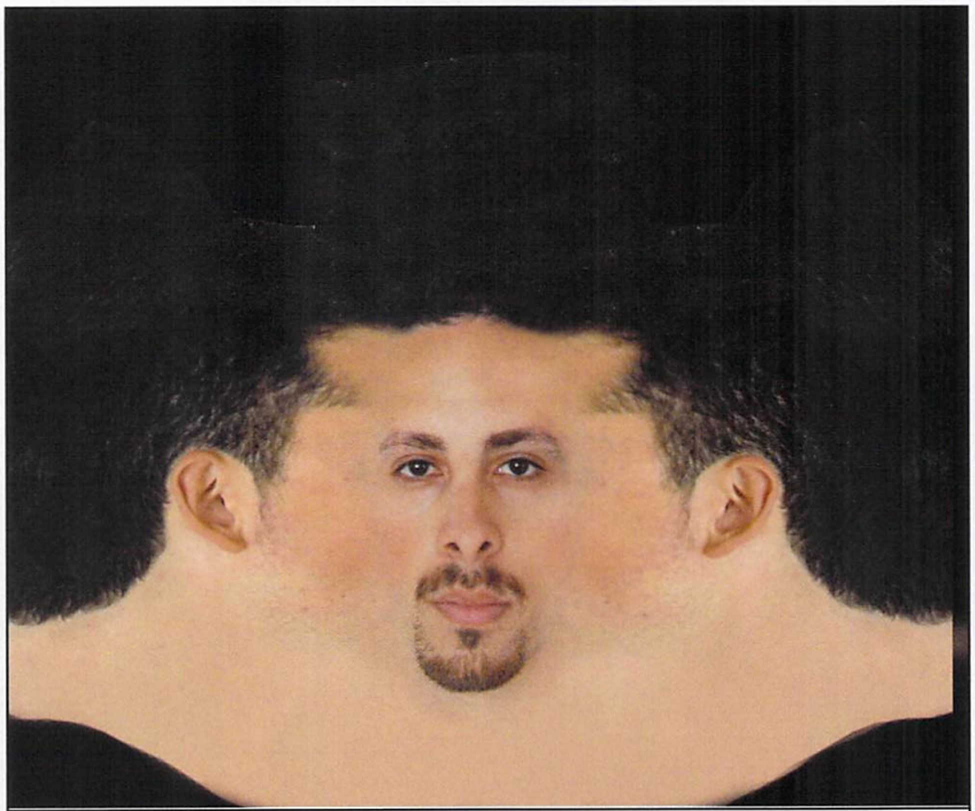
3 Once the checkerboard is evenly layed out, select your object, and in the UV Editor, go to *Polygons > Uvsnapshot*, enter in 1024 x 1024 and export. Bring the snapshot into Photoshop and use it as a template for placing and blending your reference images together. The front image should easily fit on the UV map, but the sides will need a bit more tweaks and color adjustments. Once you have one side done, copy the layer in Photoshop, duplicate, and flip to use on the other side. Adjust so each side is not exactly the same. Here is what I came up with. *(See UV Map at Right)*

4 Once the texture is applied on the model, you will need to fix certain UVs and verts for it to look how you want. *(See three images at bottom of this page)*

5 For the eyeball, I Planar Mapped and used my pupil on a 256 x 256 off-white background.

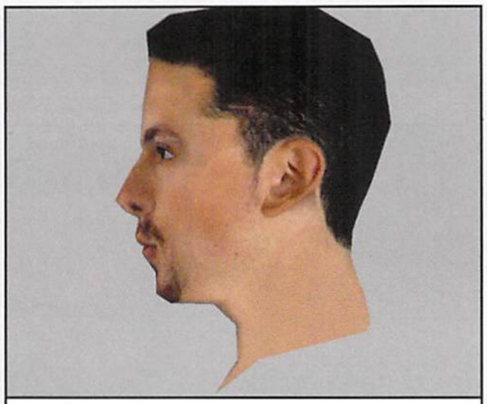

TEXTURING STEP 2– ASSIGN A CHECKERBOARD TEXTURE

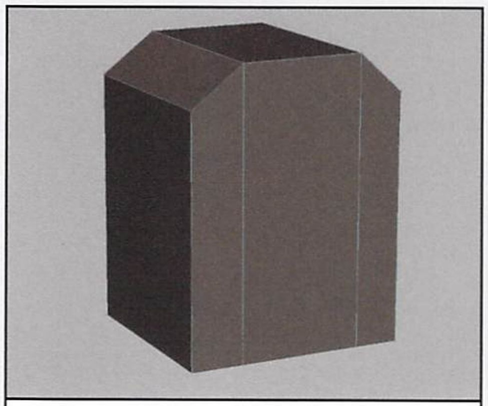

TEXTURING STEP 3– REFERENCE IMAGES ADDED

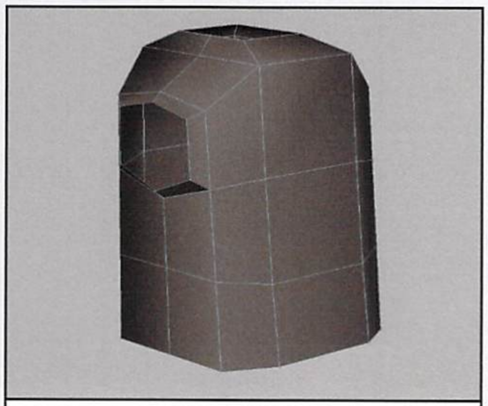

UV MAP WITH TEXTURING ADJUSTED TO MAKE EACH SIDE UNIQUE

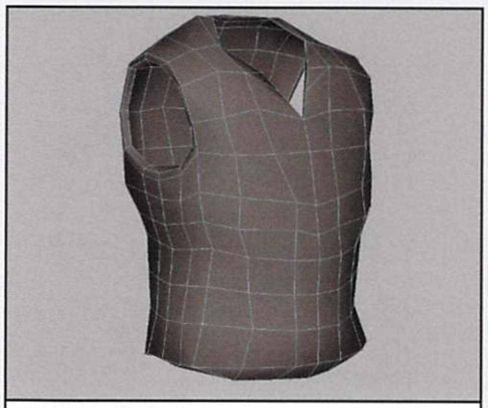
FRONT VIEW OF TEXTURE APPLIED TO THE MODEL

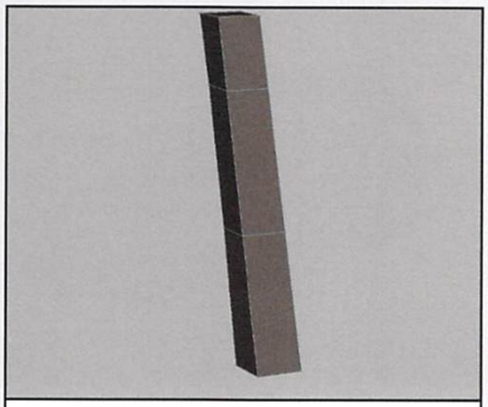
VIEW OF TEXTURE APPLIED TO THE MODEL

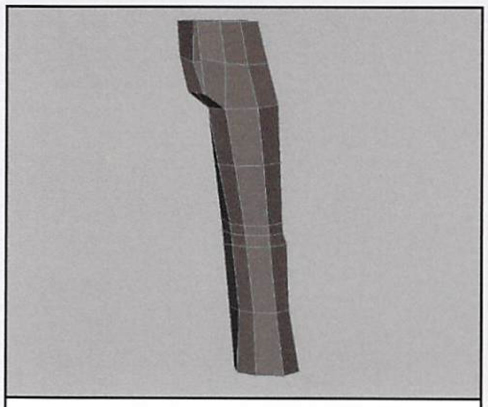

SIDE VIEW OF TEXTURE APPLIED TO THE MODEL

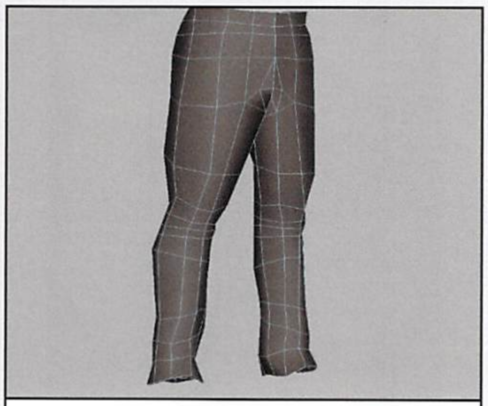

TORSO 1– START WITH 6-SIDED CUBE & SPLIT TWICE

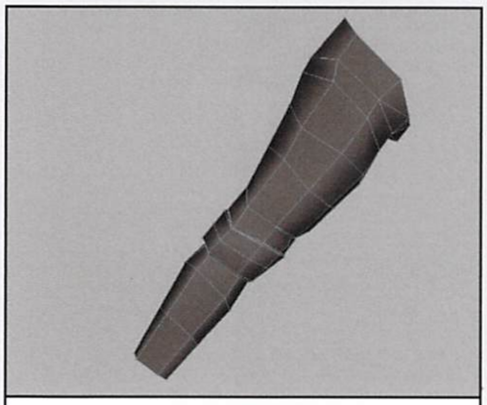

TORSO 2– DELETE FOR NECK AND ARM HOLES

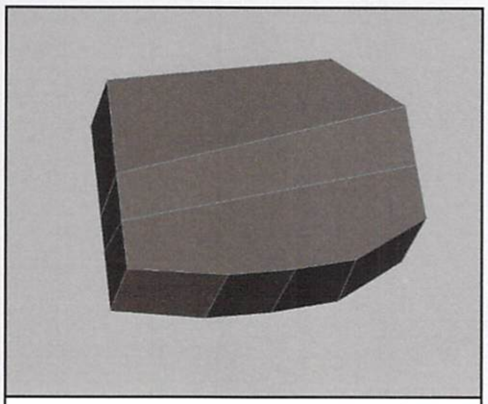

TORSO 3– SPLIT & SCULPT ACCORDING TO YOUR REF


LEGS 1– SPLIT TWICE AT WAIST AND KNEE


LEGS 2– DELETE WHERE WAIST & OTHER LEG WILL BE


LEGS 3– SCULPT AND MIRROR GEOMETRY


ARMS 1–ARMS ARE MUCH LIKE THE LEGS


HAND 1– SCULPT A CUBE SOMETHING LIKE THIS

BODY: MODELING (speed session)

UPPER TORSO

1 Start with a six-sided cube and split twice vertically.

2 Extrude the top face in and delete for the neck. Do the same for the arms.

3 Next, split and sculpt according to your reference.

LEGS

1 Start with a six-sided cube, stretch according to your ref, and split twice at waist and knee.

2 Delete poly where the waist will attach to the upper torso and also delete where the other leg will attach.

3 Add more loops and sculpt. Mirror geometry.

ARMS

1 Arms are basically built the same way as you did the legs, making sure there is enough geometry around the elbows and shoulders to deform properly.

HAND

1 Begin with a cube and split and sculpt so you have something similar to the image.

2 Extrude the faces where the fingers are and split down the hand, like the image.

3 Next sculpt, split, and make sure you have enough loops around all the joints of the fingers.

SHOE

1 This will be simple once you've completed the previous sections. Start from a box, sculpt, and split according to your reference.

Here's the completed body. *(See image, far right)*

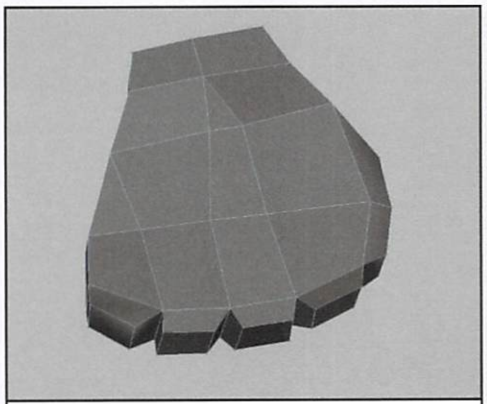
BODY: TEXTURING

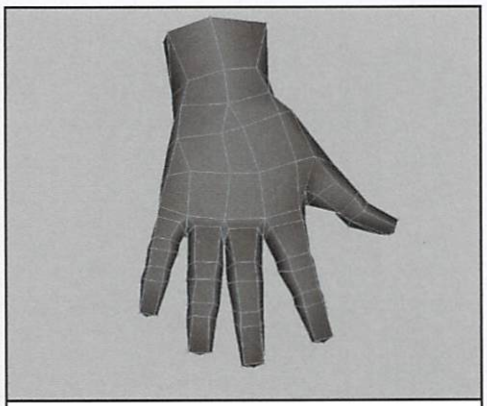
1 I used Cylindrical Mapping for just about everything except the hands, for which I used a Planar Map on the top and bottom. Make sure you always hide your seams in the most unnoticeable areas (under arm, between legs), and try to match them up as best you can in Photoshop.

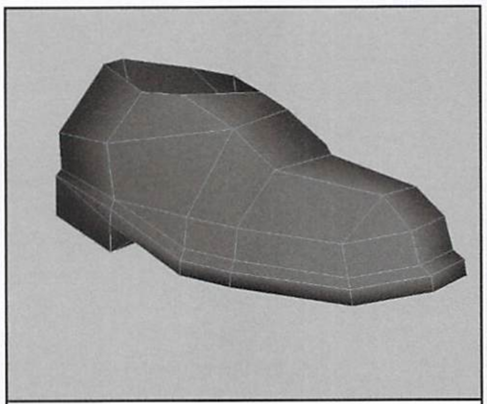
Note our final product, it contains 4300 Polygons.

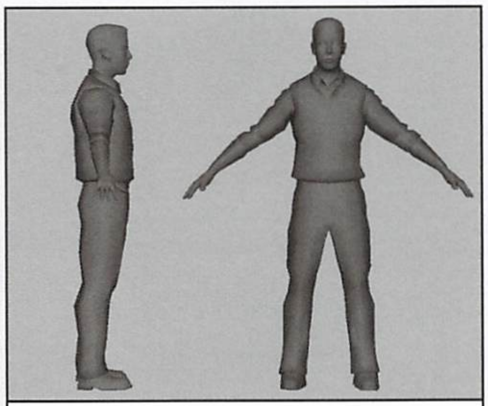
EDDIE ALCAZAR IS CURRENTLY A PART OF WWW.ALCAZAR-ENTERTAINMENT.COM WHICH HE, GREG LEMON AND TWO OTHER INDUSTRY FRIENDS STARTED IN EARLY 2004. SINCE THEN THEY HAVE BEEN BUSY WORKING ON AAA GAME TITLES, COMMERCIALS AND OTHER 3D WORKS. "OUR MAIN TARGET IS DELIVERING QUALITY ABOVE THE CLIENT'S EXPECTATIONS. WE TRY TO HANDLE ANY 3D RELATED JOB NEEDED. OUR EXPERIENCE HAS RANGED FROM PLATFORM GAME CONTENT CREATION AND VFX TO MEDICAL AND ONLINE GAMES."

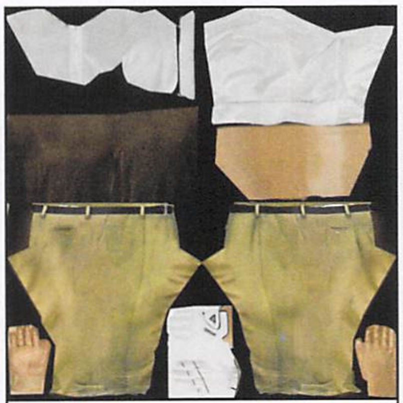
EDDIE HAS DEMONSTRATED AND PROVED HIS SKILLS WITH OTHER COMPANIES IN THE PAST, WORKING ON HUGE TITLES SUCH AS EA'S MEDAL OF HONOR'S PACIFIC ASSAULT AND BREAKTHROUGH AND HAS ALSO BEEN WORKING ON DIRECTING HIS FIRST INDEPENDENT FEATURE. HE CURRENTLY RESIDES IN SAN FRANCISCO. HIS ONLINE PORTFOLIO CAN BE SEEN AT **WWW.EDDIEALCAZAR.COM**

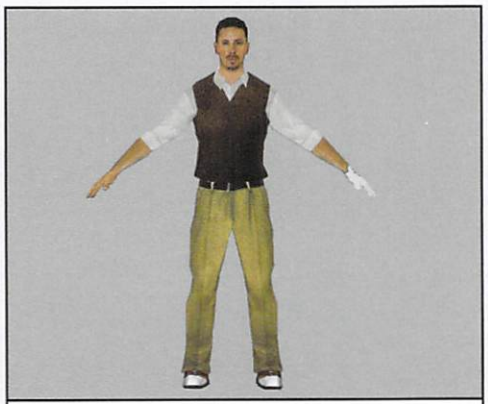

HAND 2– EXTRUDE WHERE THE FINGERS ARE

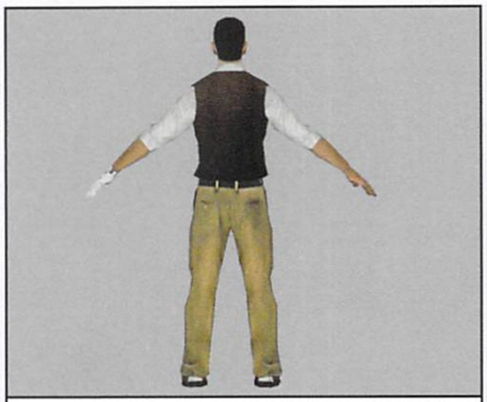

HAND 3– MAKES SURE JOINTS HAVE ENOUGH LOOPS

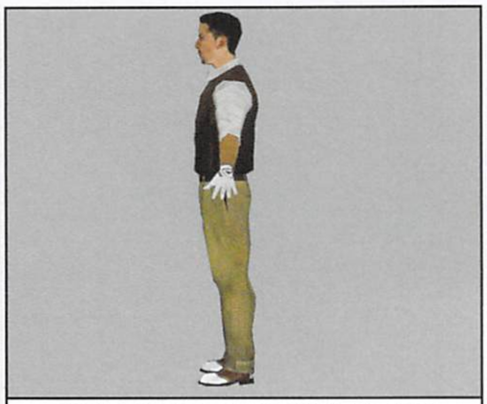

SHOE 1– START WITH A BOX AND SCULPT TO FIT REF


THE COMPLETED BODY PRIOR TO TEXTURING


BODY TEXTURE 1–TRY TO HIDE YOUR SEAMS


OUR FINAL CHARACTER


4300 POLYGONS MAKE UP THE FINAL PRODUCT


VIDEO GAME CHARACTER QUICK AND EASY

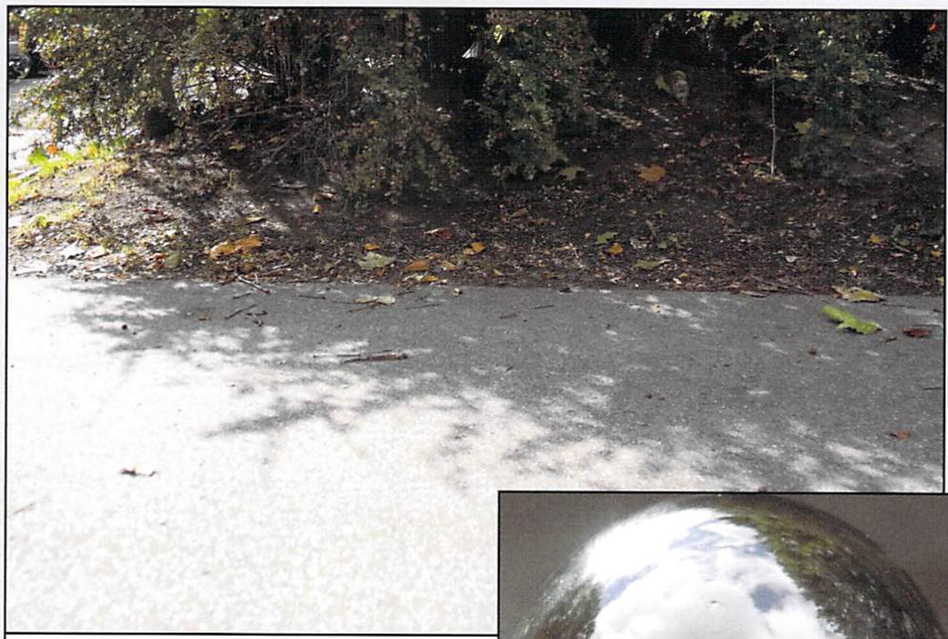
BY PETER HARTWIG

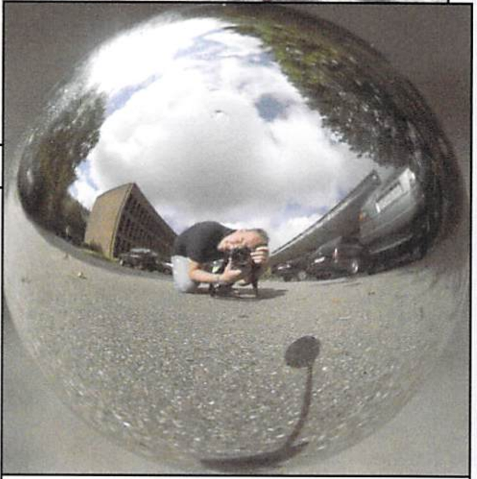
Integrating CG Into Real World Environments 101 – An XSI Tutorial

Marrying CG and live action is getting to a point where it's something almost everyone does or needs done. Even small commercial jobs with low budgets are integrating CG elements into whatever was originally shot, which is quite extraordinary when you take into account that this was considered a huge task just a few years back.

In this article, I'll sum up a few simple methods of CG integration, and I'll show you some tricks that really put the CG in the scene simply and easily.

Often there are issues with the lighting not being completely right, or maybe just not nuanced enough. Also, things like highlights burning out or shadows in the plate can cause trouble. Fear not! We'll look at these problems right now.


PIC 1– BACKGROUND PLATE IMAGE


PIC 2– CHROME CHRISTMAS BALL

For this example, I went out to the parking lot at my office and took two pictures with a Canon D10 camera, in 16-bit Canon raw format. The first image *(PIC 1)* is going to be the plate and the second *(PIC 2)* is the HDRI probe. A nice thing about the CRW format; this camera shoots in 16-bits, which makes one image enough for HDRI. I didn't have a proper mirror ball, so I used a chrome Christmas ball for this one.

In Photoshop CS, I imported the images and saved them out as 16-bit TIFF files, which are supported directly inside XSI.

In this example, I will be working on a still image, just to keep things simple, but the plate could just as well be live action film that was tracked. I usually track all my things in SynthEyes, which makes this process simple to do on clips, as well. If possible, you should still try to get the plates in 16 or 12-bits from the scanner.

The tutorial assumes you know your way around XSI, so I won't go into any detail about where to find the buttons, etc.

The first things I do in XSI are load the plate in as a rotoscope image and align the camera to it. Use the grid for this, and find some lines to align to. Next, create a plane, a cylinder *(PIC 3)* and a spot light. Also, go ahead and delete the default light now. Turn on the shadows for the spot and move it around to make the shadow from the cylinder fall like the shadow in the plate does. Move it somewhat far away, and in a viewport, look through the spot and make sure the cone doesn't cover too much more than what you see in the scene.

Now that we've got that set up, let's set up the HDRI lighting really quickly, as this will give us our basic lighting that we will be mixing with in the fxtree later.

Very often, I can't be bothered to unwrap probe images, so I just go ahead and create a large sphere that surrounds the scene and Planar Map the probe image on it. Then for the shader I use a Lambert and plug the image into Incandescence, Ambient and Diffuse. Also, turn off Primary Visibility for the sphere. *(SEE PIC 4)*

Go ahead and add a few more simple objects to see how the Final Gather will work.

Then it's time to turn on Final Gather in the render region. You'll probably have to fiddle a bit with the intensity parameter for the Incandescence on the environment sphere. *(SEE PIC 5)*

That's all the time we'll spend on that ... go ahead and fine-tune it yourself.

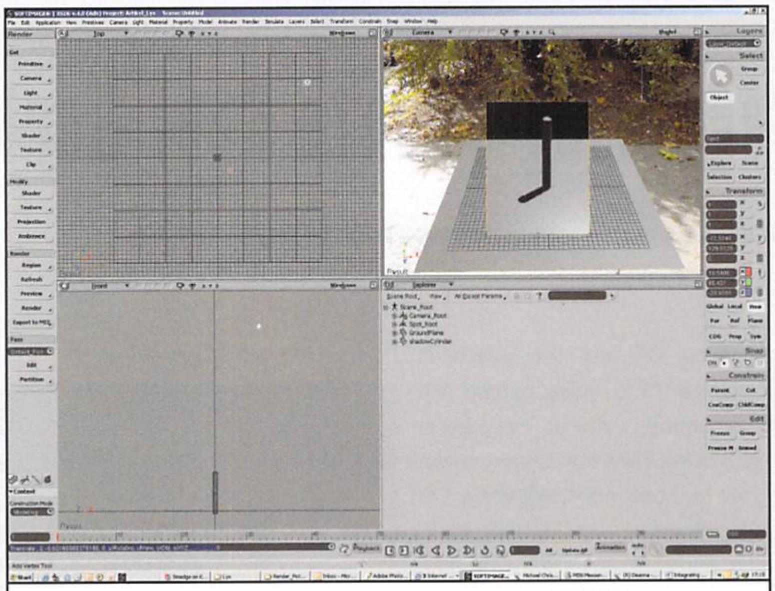
The next thing we'll look at is grabbing the gobo of the plate and putting it in your light. First of all, add a Camera Texture Projection to the plane, and put a Constant Shader with the plate as color on it. Then change to the spot light viewport, and maximize it. Right now, I can't seem to find an easy way to render out a spot light, but you could easily create a script to align a camera to the spot, etc. For now, I render a full screen render region, and right-click on it to save the region. Remember to turn off Final Gather for this render, and hide all the geometry except for the plane. *(SEE PIC 6 next page)*

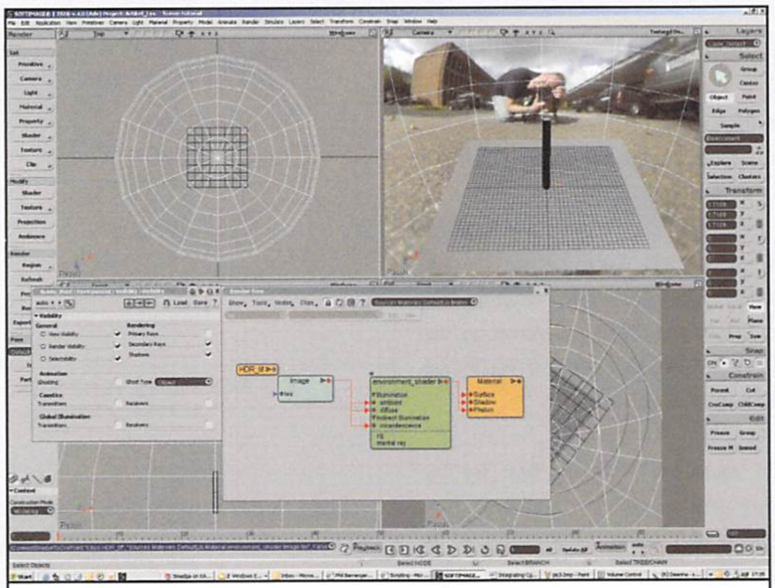
Next, open this image in a 2D software package, like Photoshop, and use a tool to adjust the levels to show the Shadow/Light line, and Paint/Blur whatever you need. *(SEE PIC 7 next page)*

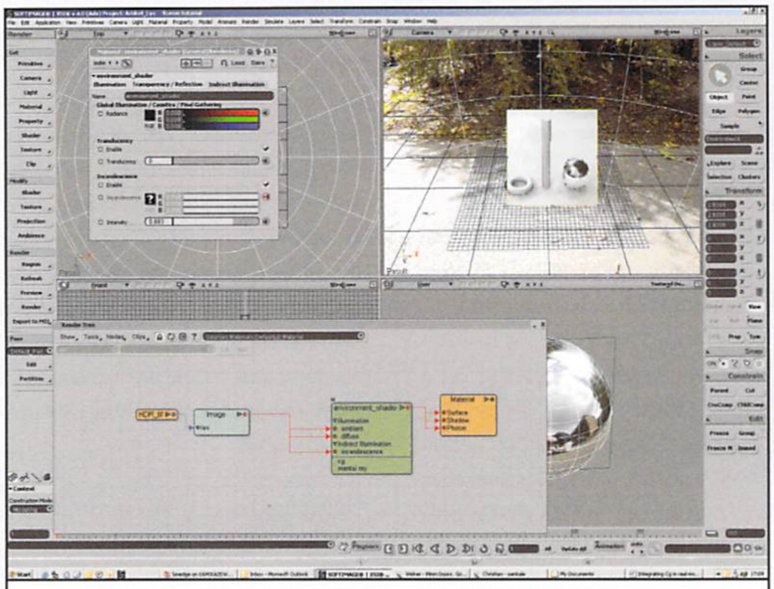
Now, set up a shader network for the light, like in PIC 8, and try rendering it with a few different objects in there to see how it works. Use the mixer to control the intensity of the gobo light. Once it works, leave it for a while; we're going to render out some passes and put it back together in the fxtree and tune it there.

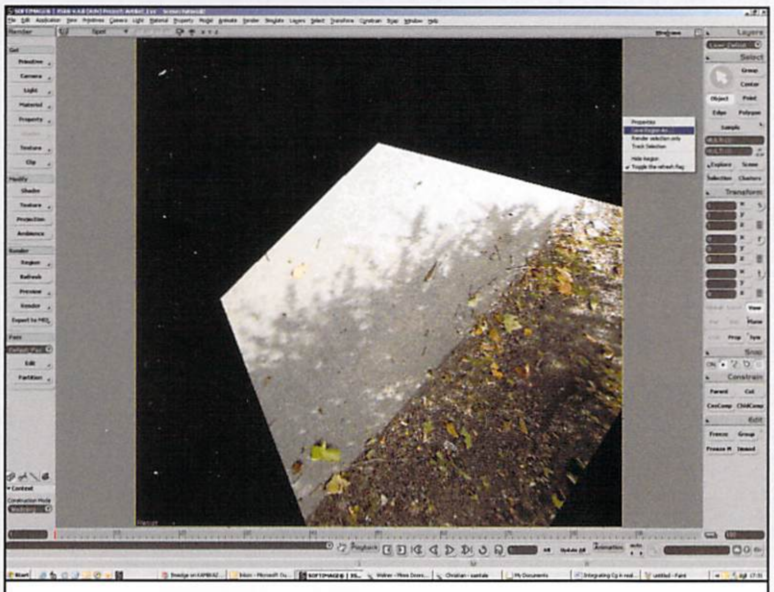
We're going to use a few passes for this next part. First of all, we'll make a pass for the gobo alone to control the light and shadow. Then we're going to set up a Final Gather pass for the main light in the scene, and then I like using the Fantastic Dirtmap plug-in to create an Ambient Occlusion pass.

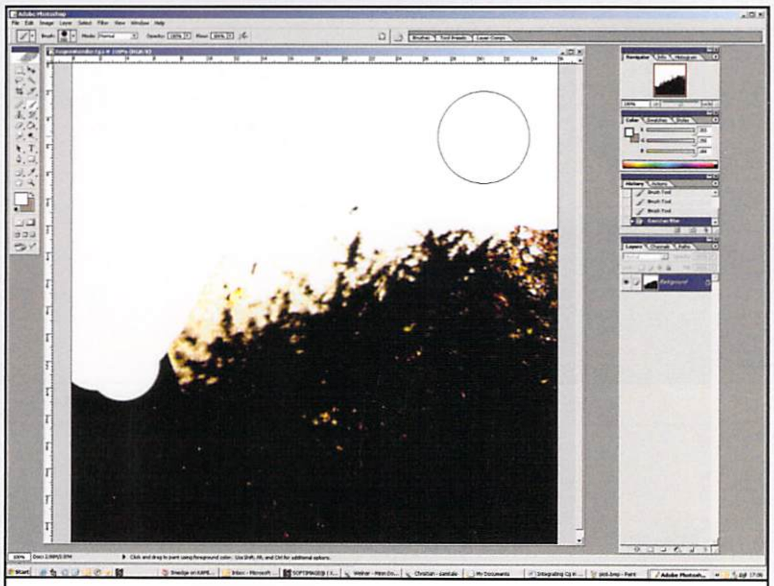
Let's quickly set them up. For the gobo, override all materials and give them a white Lambert. This pass should not have Final Gather on. The FG pass overrides the intensity of the spot and sets it to 0. Set up your Final Gather setting to whatever fits your project. I have used the Camera Projection again, to

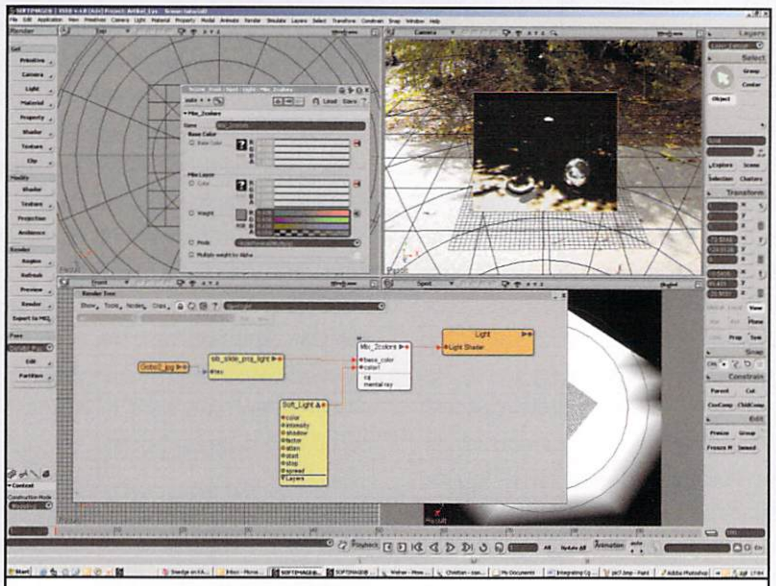

PIC 3– CREATE A PLANE WITH A CYLINDER AS SHOWN


PIC 4– CREATE A LARGE SHPERE THAT SURROUNDS THE SCENE


PIC 5– MAKE ADJUSTMENTS TO THE INTENSITY PARAMETER


PIC 6– HIDE ALL THE GEOMETRY EXCEPT FOR THE PLANE


PIC 7– IN PHOTOSHOP, PAINT BLUR WHATEVER NECESSARY


PIC 8– SET UP A SHADER NETWORK FOR THE LIGHT AS SHOWN

project the plate onto the plane, and then I piped a solid black constant shader into the shadow port of the material on the plane to allow for the shadows from the FG. Last but not least, set up an Ambient Occlusion pass by overriding all materials with a constant shader with a Dirtmap shader on it.

Go ahead and render all the passes before we go into the fxtree part...

Switch to the Compositing layout, and load in all the passes. Now, use the Final Gather pass as base, multiply with the two other passes, and adjust the intensities here. This gives you much better control to change shadow colors and values, etc., than if you did it in the 3D scene.

Another compositing trick when matching lighting is to load the plate and the CG, then start turning up the exposure while making sure that the whites burn out at the same time in your CG as in the plate; do the same for lower exposures to make sure the highlights stay visible at the same exposures.

This was just a quick and easy way to add CG elements to live action footage. The gobo effect really helps sell the idea that the CG is in there for real, so it's a good trick to know. Other than that, it's a question of breaking it down and constantly focusing on what seems wrong and why.

I really hope some of you picked up a few tricks that can help you out ... till next time...

PETER HARTWIG IS A FREELANCE TECHNICAL DIRECTOR WORKING IN DENMARK AT THE MOMENT. CURRENT PROJECTS INCLUDE FEATURE FILMS BY DANISH DIRECTORS LARS VON TRIER AND CHRISTOFFER BOE, AND SEVERAL TV COMMERCIALS. PRIOR TO THIS PETER WORKED AT **FRAMESTORE CFC** AS LIGHTING TD ON *HARRY POTTER AND THE PRISONER OF AZKABAN* AND *THUNDERBIRDS*.
AT 23 YEARS OF AGE PETER STILL HASN'T RECEIVED ANY FORMAL TRAINING, AND HAS BEEN LEARNING BY DOING, AND NOT BEING AFRAID OF ASKING.
AT THE MOMENT PETER LIVES WITH HIS GIRLFRIEND IN COPENHAGEN DENMARK. MORE WORK CAN BE SEEN ON ***WWW.IDIOTBARN.COM***

BY BRUCE G. WOLOSHYN

STARGATE SG-1 "LOST CITY PART 2" EMMY NOMINATION

RAINMAKER'S DIGITAL EFFECTS SUPERVISOR BRUCE G. WOLOSHYN SAYS IT'S ALL ABOUT THE PEOPLE.

MGM's *Stargate SG-1* has been fortunate enough to be nominated once again for a Primetime Emmy for Outstanding Special Visual Effects for a Television Series. The visual effects crew at Rainmaker was instrumental in acquiring that nomination.

THE EVIL ANUBIS HAS BEGUN A SYSTEMATIC ASSAULT ON THE EARTH, AND THE PLANET'S ONLY HOPE FOR SALVATION ONCE AGAIN RESTS WITH STARGATE COMMAND'S ELITE TEAM, SG-1.

THE RED CARPET AT THE SHRINE AUDITORIUM.

When I first heard about this episode (the season finale of our 7th year), it was described as the biggest effects episode ever for *Stargate SG-1*, not only in the volume of work, but in the complexity of the visual effects shots. An alien planet covered in lava; an aerial battle over the Antarctic; a firefight beneath the ice; the "freezing" of Colonel Jack O'Neill. The episode's description sounds grand, but for our team, the visual effects execution was all the more remarkable. These shots not only sounded impressive to me, they sounded just plain cool. And creating images that are "cool" has always been something that drives our team to put in the hours and come up with creative, near-impossible solutions to the many challenges of creating a stylistically strong, believable SG-1 universe.

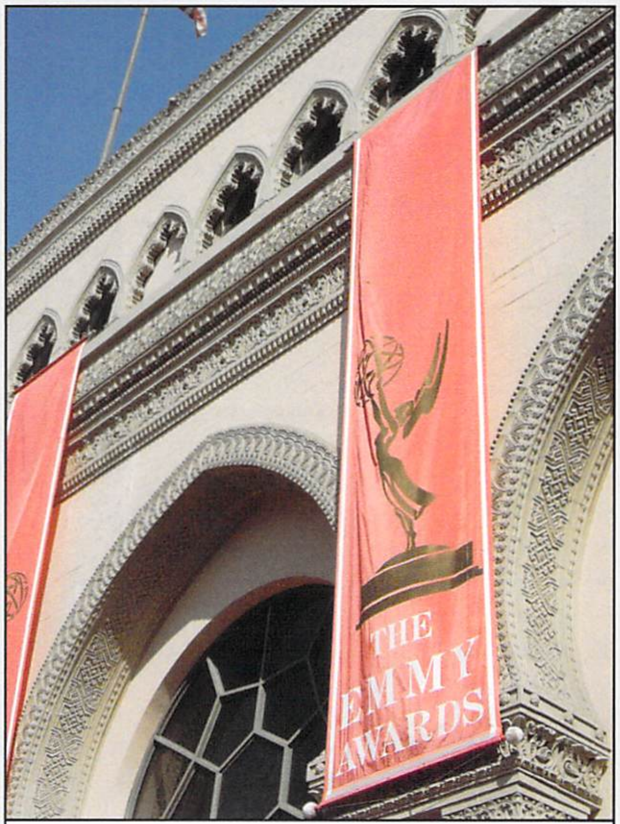
Of course we are armed with the latest technology has to offer. From LightWave and Maya, to Inferno and Fusion; from SGI to Boxx; we have it all and it all helps. But from the first sketch to the delivery of the last pixel, it is the people of this visual effects team, not the technology, who created all these "cool" images.

And it is the people for whom the Emmys are presented. People like James Tichenor, our Visual Effects Producer, and Michelle Comens, our Visual Effects Supervisor, who led our team with enough direction to maintain shot consistency and to tell a good story, but also provided enough freedom to let the artist teams enhance concepts and contribute their own designs and ideas. It's for the lead artists who shoulder the responsibility of working with their teams to ensure that every shot is completed to the satisfaction of our supervisors and executive producers.

There are many, many other people behind this nomination as well. Without our IT staff, we'd never make it from one day to the next. Without our visual effects support staff, we'd never know what we were doing from one day to the next. And as with any ballot, the number of names is limited. There are too many artists, technicians and coordinators on our team whose names didn't make it on to the nomination ballot, not because their contributions were unworthy of note, but simply because there were not enough slots for them all. They are all dedicated team members, without whose talent and hard work these shows could never be competed.

I'd be lying if I said that these nominations aren't special to me. As a farm boy growing up in the sixties and seventies watching shows like *Star Trek* and *Space:*

BRUCE AND SENIOR ANIMATOR, NICK BOUGHEN.



THE EMMY BANNERS HANG AT THE SHRINE AUDITORIUM.

1999, I dreamed of working on such amazing shows. And now, here I am, living the dream. I try to remind myself of that every once and a while when intense schedules and production problems take their toll on my stress level, my family, and my team. I feel privileged to be where I am. It is a unique and wonderful job to be involved in creating this superb work. Receiving recognition from our peers for something in which we take such pride is, indeed, very special.

It is made all the more special by the company we keep. This year's other nominees include the television series' *Dead Like Me, Stephen King's Kingdom Hospital,* and *Star Trek Enterprise.* All of these are shows I admire. All of them are peopled with extraordinary talent.

STARGATE SG-1 • "Lost City Part 2"
SCI FI • MGM Television in association with Double Secret Productions and Gekko Film Corporation

Michelle Comens	Visual Effects Supervisor
James Tichenor	Visual Effects Producer
Shannon Gurney	Visual Effects Coordinator
Bruce G. Woloshyn	Lead Visual Effects Compositor
Chris Doll	Lead Visual Effects Compositor
James Halverson	Lead Matte Artist
Craig Van Den Biggelaar	CGI Supervisor
Krista McLean	Lead CGI Artist
Patrick Kalyn	Lead CGI Animator

BRUCE G. WOLOSHYN IS ONE OF RAINMAKER'S MOST SEASONED DIGITAL EFFECTS SUPERVISORS. WORKING OUT OF VANCOUVER, BRITISH COLUMBIA, HE HAS BEEN WITH THE *STARGATE* FRANCHISES FROM THE VERY BEGINNING, STARTING WITH THE PILOT OF *STARGATE SG-1* BACK IN 1997. BRUCE HAS BEEN HONORED WITH FOUR EMMY NOMINATIONS FOR HIS OUTSTANDING WORK AS A LEAD DIGITAL COMPOSITING ARTIST ON THE SHOW. BRUCE AND HIS VERY SUPPORTIVE WIFE RAMONA MAKE THEIR HOME BY THE SEA IN SUNNY SOUTH SURREY, BRITISH COLUMBIA, WITH THEIR TWO SONS. HE IS CURRENTLY BURIED IN PRODUCTION WORK AS DIGITAL EFFECTS SUPERVISOR ON BOTH *STARGATE SG-1* AND *STARGATE ATLANTIS*.

BY CRAIG MONINS

MULTI-PASS RENDERING WITH LIGHTWAVE

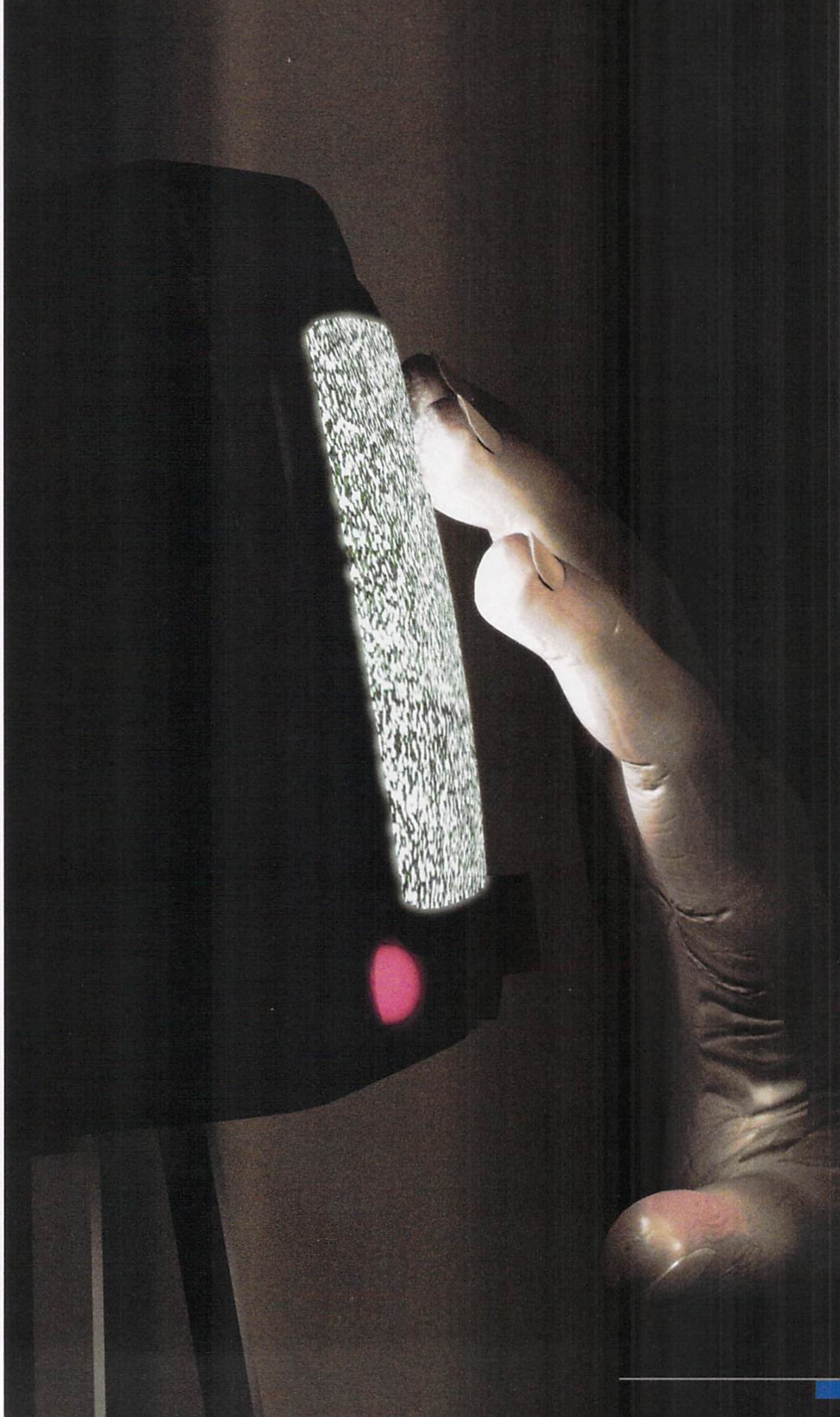
MPR WITH LIGHTWAVE

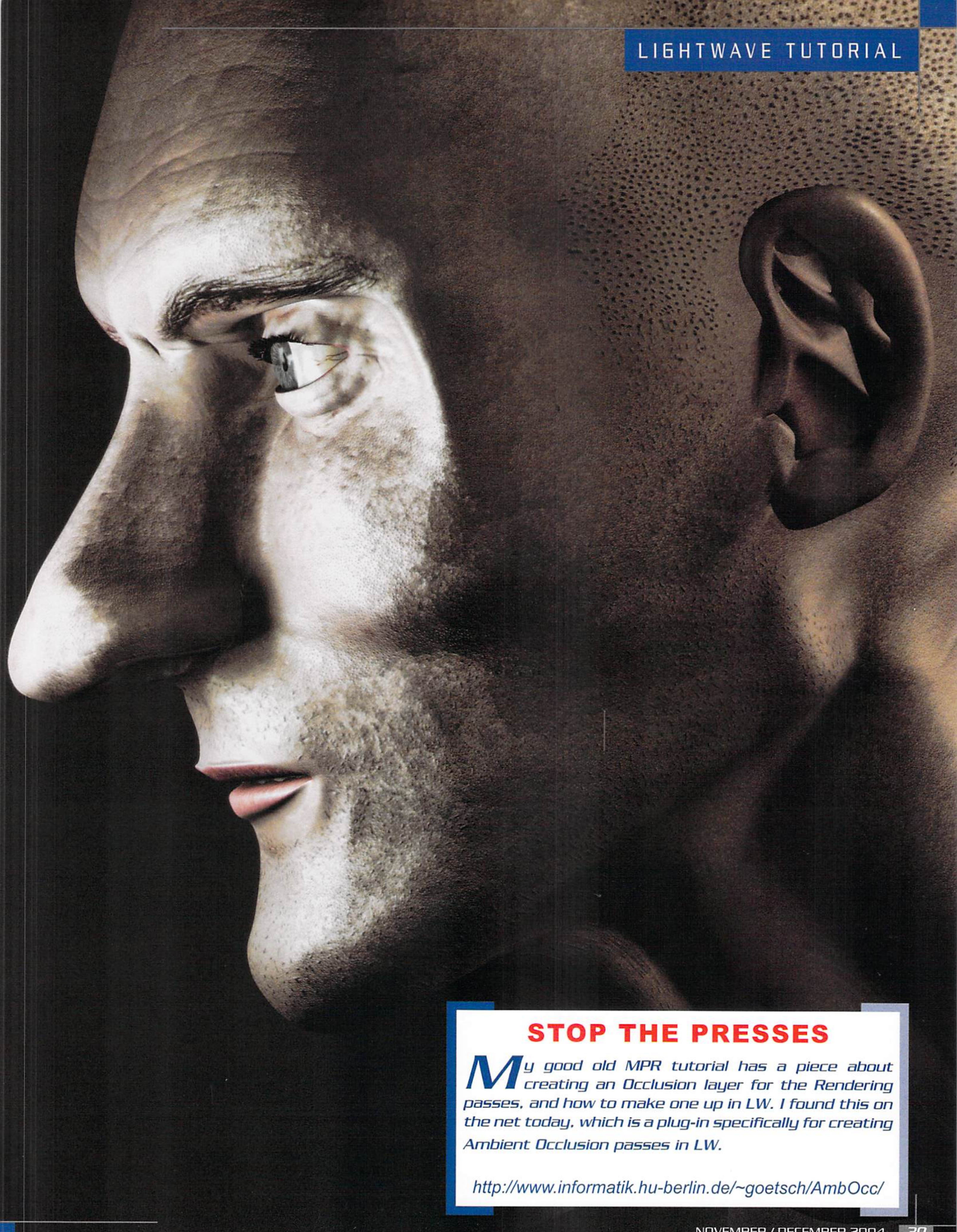
Hello again. In the last two issues of Keyframe, we were looking at various rendering, post, and vfx issues. It's time now for the third part of this small series, where we will be looking at multi-pass rendering (MPR).

MPR is a dead simple concept to get your head around, and there are a few different ways of doing it depending on the situation and its requirements. Basically what MPR involves is rendering off different elements of the scene, and recombining them in a compositor where the actual "look" is developed. In many ways, it's similar to the techniques we went through back in Keyframe Issue 39. The difference here is that we are separating out the elements from the Renderer, the Specular, the Diffuse, the Reflection, etc. This way we can use our compositing program to adjust properties much more quickly and easily than we could in 3D by playing with texture and light settings. It also gives us the option to explore a great many different looks until we get the one that's just right.

This technique is common as muck in all the big animation and effects studios, as it's simply the quickest way to work. Certain 3D packages have MPR options built into them for this very reason, Maya and C4D, most notably, and of course Pixar's Photorealistic Renderman is famed for the way it delivers different render passes for just this sort of work. In fact, its ability to handle different elements in this way is one of the things that makes PRman useful as a renderer in a production environment.

LightWave does offer an MPR option of sorts in its Photoshop export filter. Unfortunately, this is cumbersome, and pretty useless, and offers nowhere near as much control as the techniques I'm going to show you here. It's also worth noting that this method of doing multi-pass rendering works with just about any 3D software, even Poser. So let's get to it.





STOP THE PRESSES

My good old MPR tutorial has a piece about creating an Occlusion layer for the Rendering passes, and how to make one up in LW. I found this on the net today, which is a plug-in specifically for creating Ambient Occlusion passes in LW.

http://www.informatik.hu-berlin.de/~goetsch/AmbOcc/

First of all, go download some files to work with. You can get all my example files from the HDRI Web site at: www.hdri3d.com/resources.

The first thing I'm going to take you through is how to set things up to give a set of different rendered passes.

OK, for simplicity's sake, we'll only worry about the character in our image. We want to create a set of alternate scenes containing different versions of the model, one for each render pass. First things first. In the Object Properties render tab, make all of the other objects with the exception of the killer unseen to the camera, and set to Cast, but not Receive or Self Shadow. *(Image_01)* If you've been following the past couple of tutorials, you should remember how to do this nicely and speedily with the Spreadsheet Editor (or the new Scene Editor in 8). Now Save Scene As, and choose an appropriate name. This will become the diffuse (or beauty) pass, in which the character is rendered off normally as if MPR were not being used.

Once again Save As, and choose a new name. We'll start by setting up the Occlusion pass, so shot_ ocl.lws. The Occlusion pass (or Shadow pass) gives us a render that basically contains data about how strongly illuminated each point on the mesh is. Once in a compositor, this can then be used to selectively shade different elements of the render in subtly different ways as we chase down that perfect look.

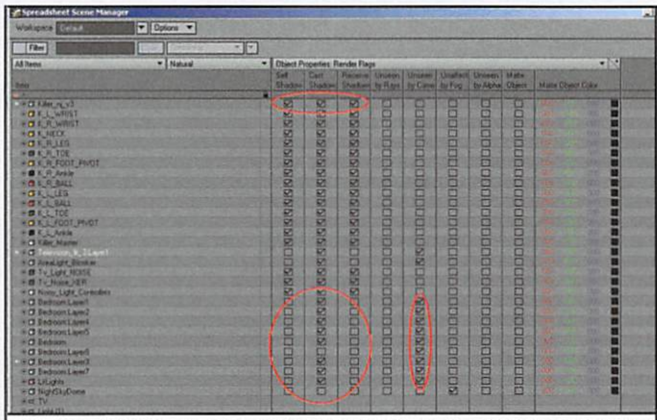


IMAGE 1– MAKING CERTAIN OBJECTS INVISIBLE USING THE SPREADSHEET

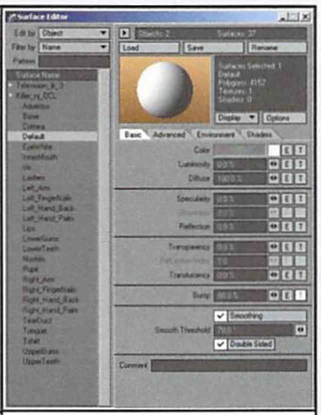


IMAGE 2– SETTINGS FOR THE OCCLUSION SURFACE

So now crack open Modeler and open your mesh. Save As, and call it, perhaps, mesh_ocl.lwo. What we want to do with this alternate model is to remove all its surface properties with the exception of the Bump map. If you look at the image *(Image_02)* you can see that all the surfaces have been changed to pure white, with 100% Diffuse, and everything except the Bump gotten rid of. Switch on over to Layout, and if the model hasn't automatically replaced itself, choose Replace With Object, and select the new mesh_ocl mesh. Finally, make sure only Ray Trace Shadows is turned on, as we don't need any Reflection, Refraction, or so forth here.

If you now render a frame, you'll get the following image *(Image_03)*, or something like it depending on your mesh and lighting.

This Occlusion pass does us another favor also. Without any of the colored texture detail getting in the way of our perceptions, it is much easier to get a real idea of just how good (or bad) a job we have done with our lighting set up, so right away we can tell that this render is shaping up well based on the shading alone, which is the most important part of good CG renders.

Next then we'll go for the Specular. Back in Modeler, open the original object, and basically do exactly as before, except this time, you'll want to make the surfaces pure black, with everything except the Specular and Bump channels turned off. Again, we make another copy of the scene using this model, called shot_spc, or something equally as appropriate. Again, we only want the Traced Shadow option on. Render this off. *(Image_04)*

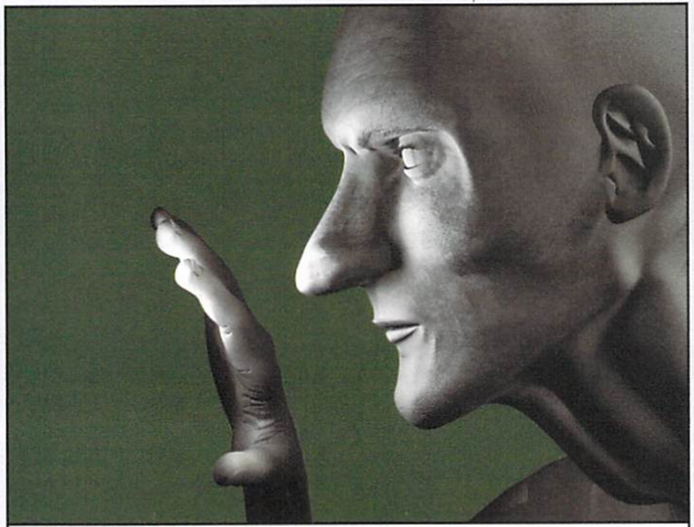


IMAGE 3– THE OCCLUSION PASS RENDERED

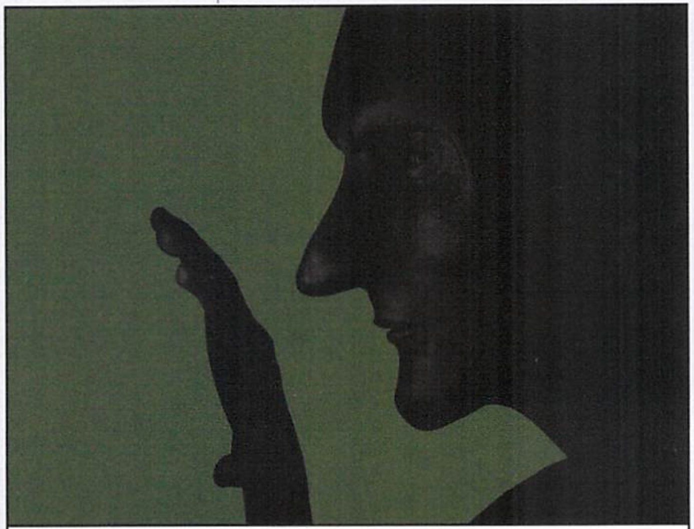


IMAGE 4– THE SPECULAR PASS RENDERED

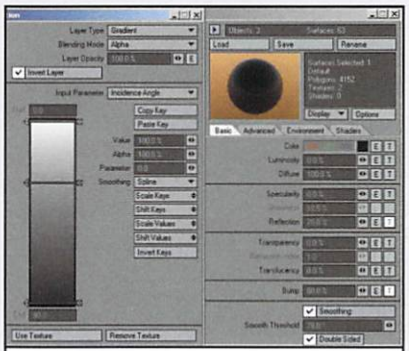

Layer Type: Gradient
Blending Mode: Alpha
Layer Opacity: 100.0 %
✓ Invert Layer
Input Parameter: Incidence Angle
Start: 0.0
End: 90.0
Value: 100.0 %
Alpha: 100.0 %
Parameter: 0.0
Smoothing: Spline
Objects: 2 Surfaces: 63
Surfaces Selected: 1
Default
Polygons: 4152
Textures: 2
Shaders: 0
Luminosity: 0.0 %
Diffuse: 100.0 %
Specularity: 0.0 %
Reflection: 20.0 %
Transparency: 0.0 %
Translucency: 0.0 %
Bump: 80.0 %
✓ Smoothing
✓ Double Sided

IMAGE 5– SETTINGS FOR THE REFLECTION SURFACE

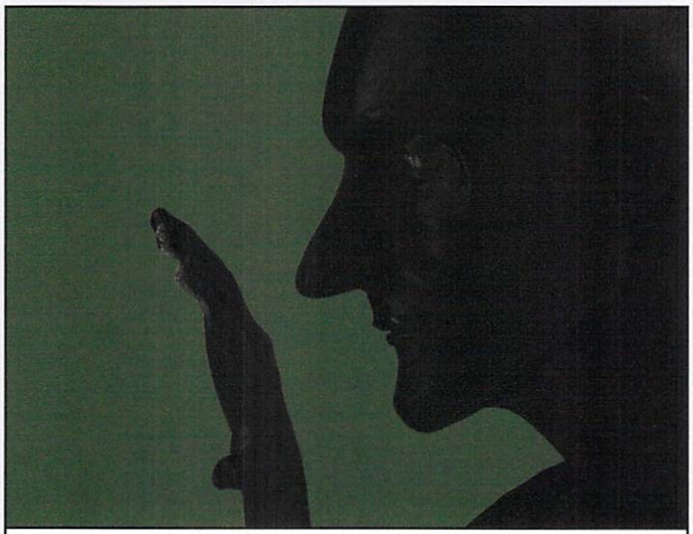


IMAGE 6– THE REFLECTION PASS RENDERED

Next comes the Reflection pass. Same thing again, new model and scene, with the model's surfaces being the same as for the spec, but obviously with Reflection and Bump on instead. You'll want to add a gradient to give the fresnel effect on the Reflection channel; you can also choose to copy the UV map used for the Specular into the Reflection channel if you like. *(Image_05)*. One thing to make sure of, though, is that the Reflection for the skin is set to a lower value than that of the eyes and fingernails (as these are shinier). If you want, you can always copy the eyes out to a second object layer, and do those separately for full control, but it's not totally necessary.

This time when we render, we'll want to activate the Ray Traced Reflection option in the Render panel, and turn the Recursion Limit down to 2 to help keep the render relatively speedy. You should get this. *(Image_06)*

Time for the last one, the Ambience pass. Using the original model as our subject, get rid of all the lights in the scene (and turn off the last one that refuses to leave), and turn up the Ambience in Global Illumination to 100%. Render. Obviously, this is our result. *(Image_07)*

Of course, lastly we go back and render off one pass for the background objects, by hiding the foreground ones. Like so. *(Image_08)*

Time now to crack open the compositor. I'm still using After Effects, and you'll find the AEP in the resource file, but you can do this in absolutely any compositor you like. You can do it in Photoshop too, you just don't get as much final control due to the lack of a precompose option.

First, the neatest trick. It's so obvious, I bet you'd never have thought of it. Load up the images you rendered, and drag the Ambience pass into a new composition. Above it drag the Occlusion pass, and set its Transfer Mode to Multiply. Bingo. A perfectly shaded render. *(Image_09)* Hang on though; this is pretty much what we'd get normally (ie. No MPR), isn't it?

Well, yes, but look at our options now. Firstly, you can alter its Opacity, allowing through more Ambience, and lightening the shadows. But wait, you can also copy the Occlusion layer and add it again (or play with its levels) to darken, or harshen the shadows. Or you can set it to be overlaid, or hard light. Now do you start to see some of the possibilities, and that's only using two passes? *(Image_10)*

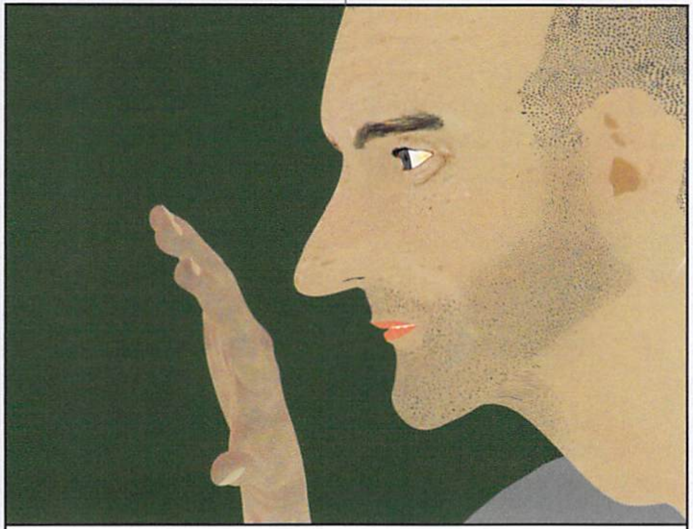


IMAGE 7– THE AMBIENCE PASS RENDERED

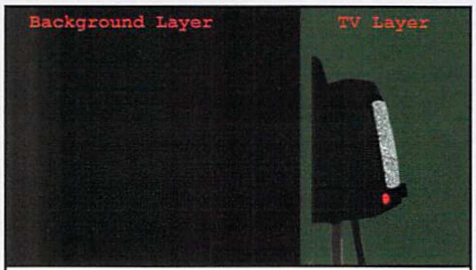


IMAGE 8– THE BACKGROUND AND TV LAYERS

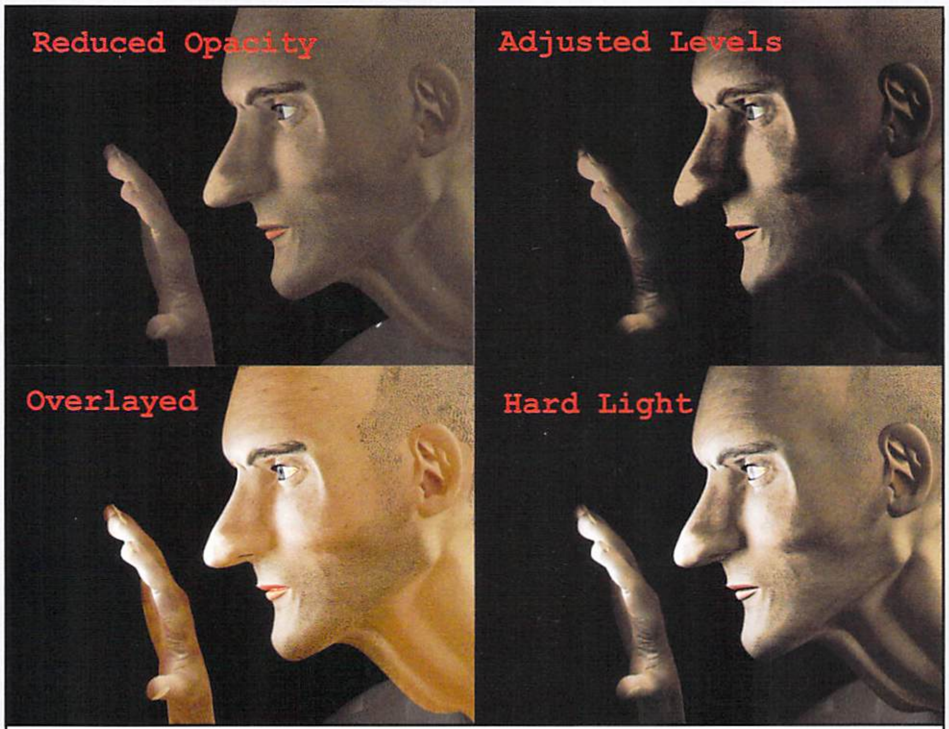


IMAGE10– THE VARIOUS EFFECTS ACHIEVED BY CHANGING THE TRANSFER MODE OF THE OCCLUSION PASS

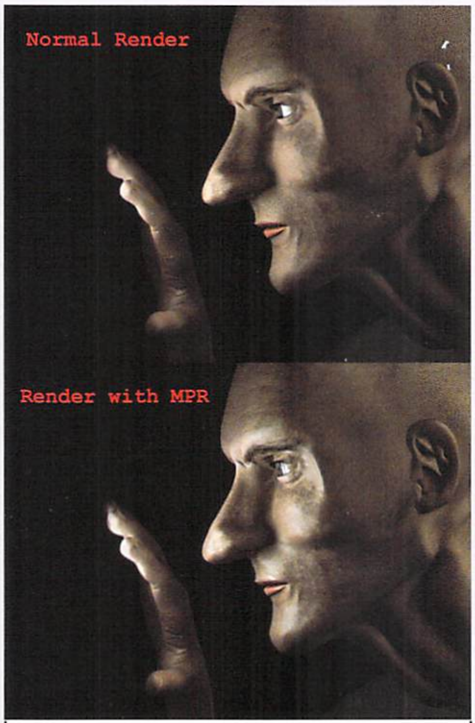


IMAGE14– THE SUBTLE DIFFERENCES BETWEEN A NORMAL RENDER AND MPR

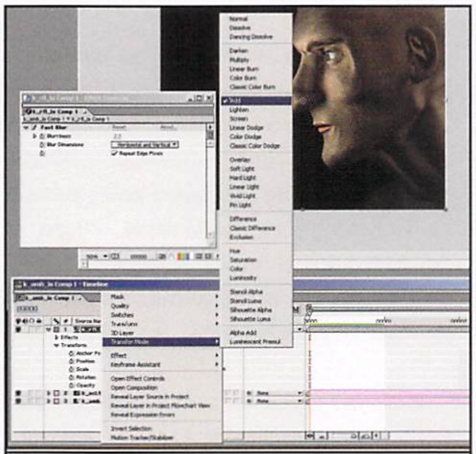


IMAGE 11– THE FIRST REFLECTION PASS IS ADDED

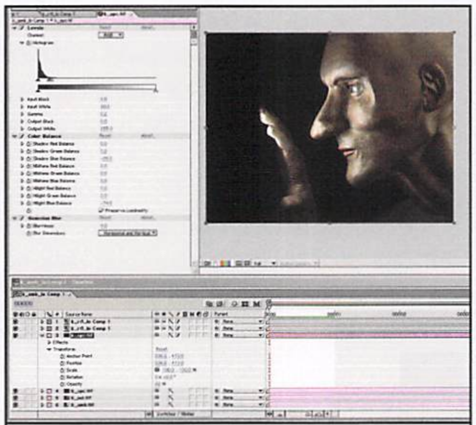


IMAGE 13– THE SPECULAR PASS IS ADDED

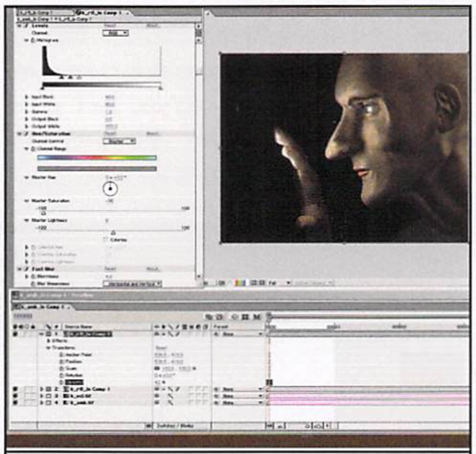


IMAGE 12– A SECOND COPY OF THE REFLECTION IS ADDED

Now create a new composition using the Reflection pass. Again, dump the Occlusion pass on top of this layer. I'm wanting to use the Occlusion pass to give more contrast to the reflection. I could just play with the levels, but this is far more useful. To do this, set the Occlusion layer's Transfer Mode to Overlay. See the subtle difference?

Drag this new reflection composition back on top of the two layers in our first composition. Right-click on it and choose Effect>Fast blur, and give it a minor blurring of 1 or 2 pixels. Finally, set its Transfer Mode to Add. I find this a bit bright, so I turn its Opacity down to 50%. *(Image_11)*. Looks nice, but I want to make those highlights ping just a little more. I copy and paste the reflection comp again, but this time add to it an extreme levels adjustment, and a Hue/Saturation effect, to desaturate it and remove the color. The Opacity can be toned down to around 40%, and I've also changed the Transfer Mode to Screen. *(Image_12)*

Once you've got this set up, you can go back to the reflection composition, and play around with it if desired, change the Transfer Mode on its Occlusion pass and so forth to further fine tune the reflection, and it'll update the main composition as you go. There's that control thing again.

OK, so now we can add the Specular pass. Drag it into the main composition, position it below the reflection layers, and set its Mode to Add. I find it looks good just like this. Maybe I want to brighten it a little, and maybe introduce some color into the highlights, so I add another copy of the Specular pass, give it an extreme levels adjustment again, adjust the color balance to turn it a little yellow, and finally blur it off about 4 pixels or so. Again, I set this to Screen, and down to 22% Opacity to soften it down some. *(Image_13)*

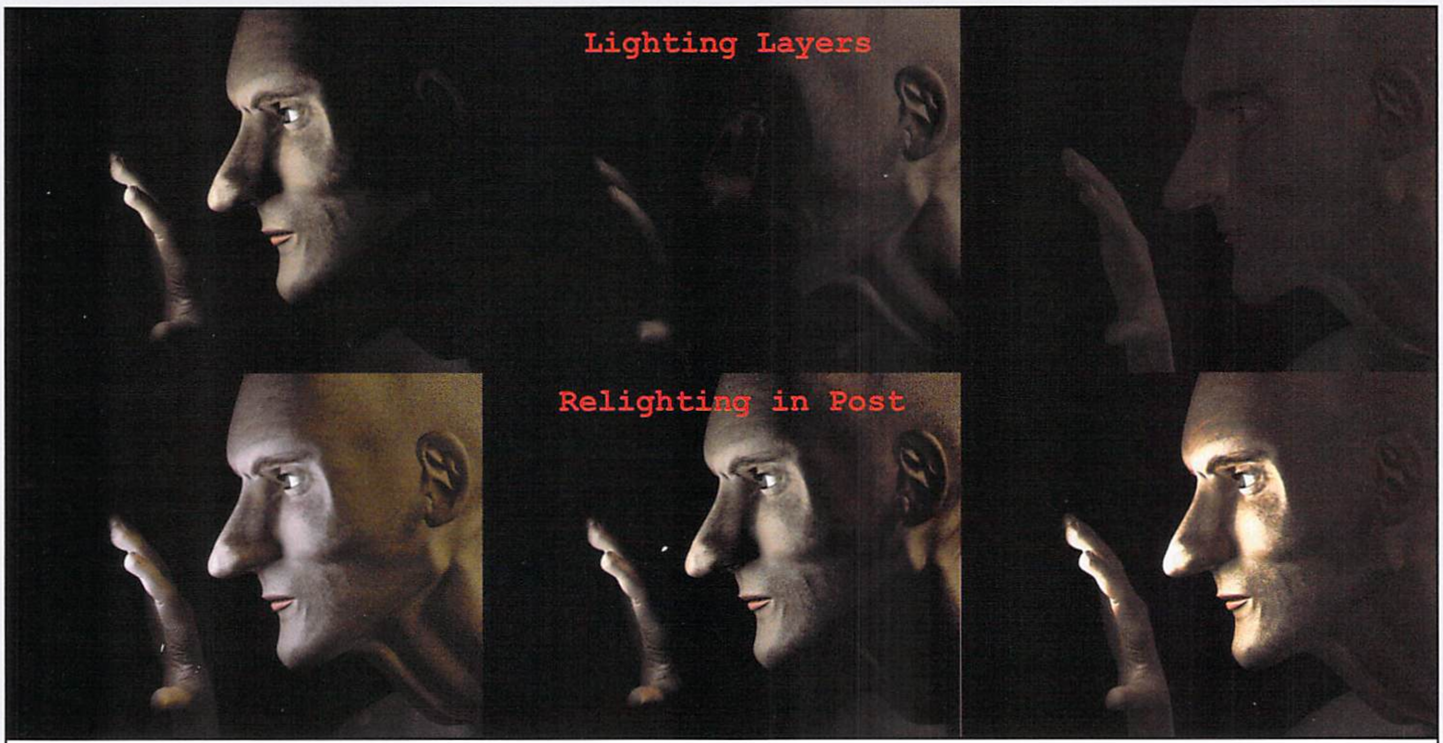


IMAGE15– THE VARIOUS EFFECTS ACHIEVED BY CHANGING THE TRANSFER MODE OF THE OCCLUSION PASS

Lastly, add to the very top the normal diffuse render. By toggling the Visibility on and off, you can see the difference in look achieved through MPR. True enough this look could have been achieved in LightWave, but it would have taken far longer to mess around with surface and light settings, not to mention all that added render time. *(Image_14)*

The interesting thing here is that all this is much more irrelevant for pure CG work like our image here. Where techniques like this really come in useful is with visual effects work, like our dinosaur from the last issue of KeyFrame. Working with multiple passes allows us to get far better matches when compositing CG elements onto live action backdrops. It lets us easily match things like shadow darkness and color, specular burn out and blurring, and so on. Now that you've got a handle on creating alternate scenes and models to get multiple passes, I'm going to show you one or two more small tricks that come in useful in other situations that involve MPR techniques.

This first one is also pretty obvious – relighting. How much time can you spend playing around with lights at different intensities and colors trying to get something looking just so? Try rendering off a few different passes of your regular model, but this time, do a different render for each light instead of each surface element. Do also an Occlusion and Ambience render, and you'll get full control of your scenes lighting in post. Place the Ambience and Occlusion in a pre-composition, then mix your lights in the main composition by varying their opacities and color balances. Finally, blend the Ambience/Occlusion layer over the top and quickly and easily experiment with hundreds of different looks. *(Image_15)*

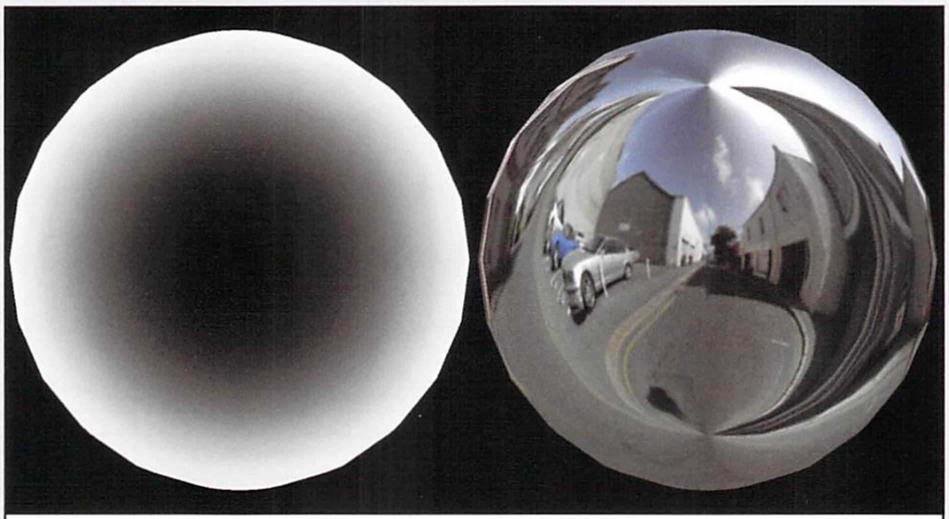


IMAGE16– THE TWO VERSIONS OF THE RENDER

But wait, that's not all. Here's a small trick for adjusting reflections in post. No doubt most of you will be familiar with the fresnel effect, which causes a reflective surface to have varying degrees of reflection intensity depending on the

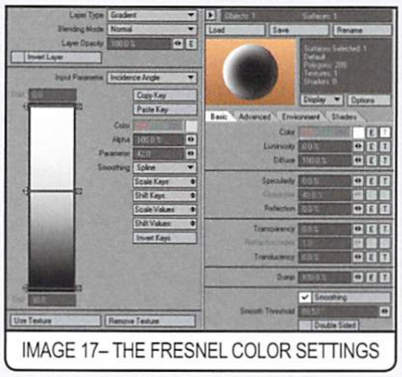


IMAGE 17– THE FRESNEL COLOR SETTINGS

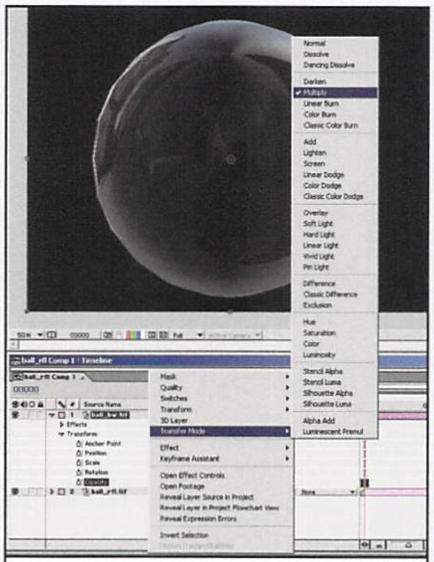


IMAGE 18– THE RESULT FROM COMPOSING THEM IN POST

viewing angle. Well, you can set it up in keeping with perfect physics, but in 3D we usually need to circumnavigate reality a bit to get things looking nice. Take any object that you want to add a fresnel effect to, and render off two versions. *(Image_16)* The first should be fully reflective, the second should be colored pure black, with the fresnel gradient in its Color channel so as it becomes white at glancing angles. *(Image_17)* Back in your compositor, multiply the black and white image over the top of the reflection pass, and by altering its Opacity and Contrast, you can control the fresnel in post. *(Image_18)*

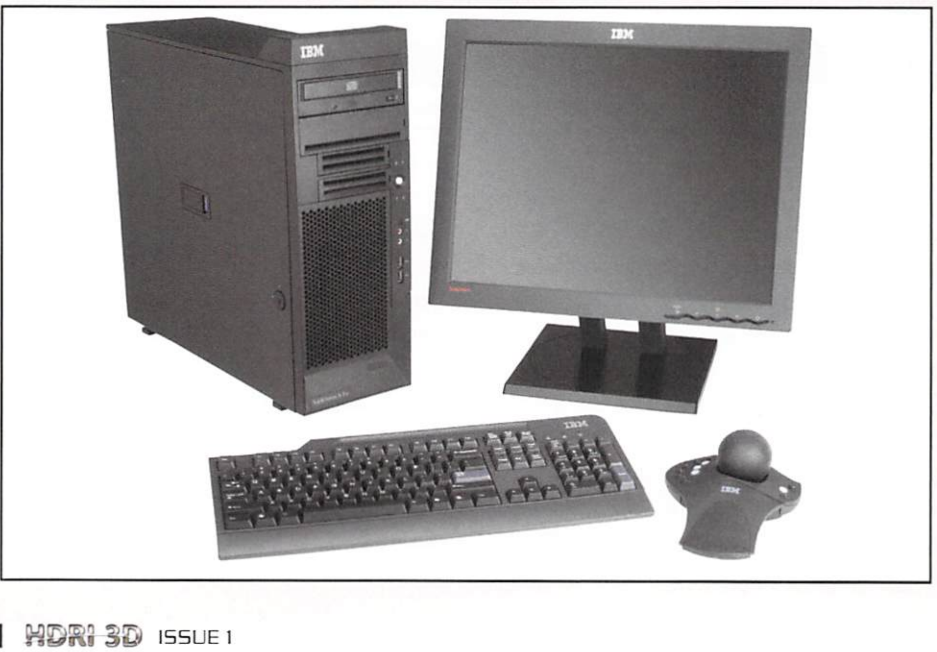
Well, I could go on, showing you countless ways to combine different passes in different ways to help out in a whole heap of different situations, but space is running short, and I'm sure that the techniques discussed here will give you more than a decent sized platform to start incorporating Multi-Pass Rendering into your own work.

C.S. MONINS IS A 3D ARTIST, AND INDEPENDENT FILM-MAKER FROM THE UK. HIS PERSONAL ANTICS AS WELL AS MUCH ABOUT HIS PERSONAL LIFE) CAN BE FOUND ON HIS WEBSITE AT *WWW.REBELHILL.NET*

GOD SAVE THE QUEEN

BY AKIKO ASHLEY

WHAT YOU GET IS EVEN MORE THAN WHAT YOU SEE... *THIS MACHINE IS FAST!*



The IBM IntelliStation APRO blows away everything in the IntelliStation line that came before it. This is a monster fast machine with the fastest processor on the planet. The IBM IntelliStation APRO in this review features a super fast, dual 64-bit Opteron 2.2 processor, 2 gigs of RAM, and a smoking NVIDIA Quadro FX 3000 card. This 64-bit chip runs both 64-bit and 32-bit apps, giving you the ability to migrate to 64-bit without losing support for your applications.

One of the technologies that makes the AMD Opteron so fast is the HyperTransport Bus technology. HyperTransport is a point-to-point link architecture designed for high bandwidth and low latency between integrated circuits. HyperTransport links the processors and chipsets on the motherboard and provides speed of up to 19.2 GBs of bandwidth per processor. It gives the chip the ability to be scalable. The Opteron chip has larger onboard cache than other models, which makes rendering on this machine so much faster.

Good examples of this rendering capability are the rendering results we have had from different software packages. For example, we modeled a character in LightWave3D and animated her in Messiah Studio using two machines. One of the machines was a Dual Xeon 3.06 processor top of the line workstation with the same setup as the dual Opteron APRO. We were using Windows XP Professional on both machines, which customizes to deliver a much more stable environment in which to run your software. The exact same software is on both machines.

Here are some of the amazing results.

We rendered the character on both boxes in Messiah Render. The animation was a sexy walk cycle on both machines. Oddly enough, loading the file on the Dual Xeon made the machine crash. We decided to reload it just in case it might be beginner's luck. Nope, it crashed loading the scene again.

On the other hand, the APRO dual Opteron was a breeze without any problems. We started our rendering tests. Immediate results from rendering individual frames told us that the dual Opteron APRO was killing the Dual Xeon with rendering speeds up to 50 percent faster. Frames on the Dual Xeon were taking 70 seconds to render and only 45 seconds on the dual Opteron APRO. At times, we were even seeing twice the speed on the APRO. We swapped the files on both machines to make sure software corruption could not be blamed on tainting the results. We got the same results. The dual Opteron APRO was the king of rendering in this test.

Animation is another test we decided to run just to get a feel of both boxes while working on projects. This is an interesting fact to share; the Dual Xeon fan came on four times as much as the fan on the dual Opteron APRO, and the Dual Xeon crashed six times while the APRO stayed completely stable. We even tried another Dual Xeon to make sure this was not a fluke. Same results.

Project Messiah does not write any software to take advantage of either processor, so this was a pretty fair test. We did other animation and rendering tests and got the same results. In fact, we even did a motion capture test and had so many crashes on the Dual Xeon that we had to stop using it for motion capture tests. We used Messiah Studio for these tests to demonstrate how necessary it is to understand that stability issues are both hardware and software related.

You have to give it to IBM for creating a top-notch workstation that takes advantage of the power of the AMD Opteron chip and delivers it in a slim case that looks great in any office or studio. What is even more amazing is the AMD solution is also a much more cost-effective solution to the Intel Xeon solution with the same configurations. In other words, consider the AMD box over the Intel box if you want a killer power solution that is both blazing fast and cost-effective. IBM does a great job customizing Windows to give you the additional stability to run your apps. IBM provides a three-year on-site warranty that is very valuable in production environments where time is money. What else can I say? The AMD box is the right solution all the way across the board. Inside every 64-bit Opteron workstation is a superhero's power waiting to get out.

The IBM APRO comes with several configurations:

- AMD Opteron Processors 2.2 Ghz (1 or 2)
- Windows Professional XP or Red Hat Enterprise Linux WS 3 Standard (AMD 64)
- 64MB NVIDIA Quadro FX 1100 to 256MB NVIDIA Quadro FX 3000 cards
- AGP 8X graphics port.
- 6 (5) PCI slots
- Gigabyte Ethernet (integrated)
- 1 Gig to 2 Gigs of PC 3200 DDR SDRAM (with ability to go to 16GB)
- 36.4 GB Ultra 320 SCSI 10,000 drive
- CD-ROM 48 Max to CD-RW/DVD-ROM Combination 48X/24X/48X/16X Max
- Prices Range from $2619.00 to $6419.00 at IBM's preconfigured machines at their Web site at www.ibm.com.

For more information about IBM, go to www.ibm.com, and for more information about AMD and the Opteron chip, go to www.amd.com.

AKIKO ASHLEY IS ONE OF THE FOUNDERS OF LUMINETIK 3D ANIMATION AND VFX STUDIO. AKIKO HAS WORKED IN THE ENTERTAINMENT AND TECHNOLOGY INDUSTRY FOR YEARS. SHE STARTED HER CAREER WRITING GAMES FOR NINTENDO IN JAPAN IN THE EARLY 80'S. AFTER SHE LEFT JAPAN SHE WORKED IN THE MUSIC INDUSTRY DISCOVERING NEW CREATIVE TALENT. SHE DECIDED TO START HER OWN TECHNOLOGY CONSULTING FIRM IN NEW YORK CITY IN 1994. AKIKO JOINED LUMINETIK IN 1999. AKIKO WROTE LUMINETIK'S BUSINESS PLAN, MARKETING & PR STRATEGIES AND BUILT THE COMPANY'S TECHNOLOGY ALLIANCES. AKIKO IS AN EXECUTIVE PRODUCER ON SUCH RECENT PROJECTS AS VFX WORK ON THE CASTLEROCK/ WARNER BROS. MOVIE *TWO WEEKS NOTICE*, FOUR COMMERCIALS FOR ARM AND HAMMER FEATURING NEW YORK YANKEE JASON GIAMBI, AND CGI WORK FOR NATIONAL GEOGRAPHIC ON *SNAKE WRANGLERS*. LUMINETIK'S VISION IS TO BE A ONE STOP PRODUCTION SOLUTION, AKIKO IS CURRENTLY BUILDING THE MUSIC DIVISION OF THE COMPANY WITH PARTNER COLIN GIBBENS. LUMINETIK WAS ONE OF THE CO-SPONSORS OF NYDV EXPO AT MADISON SQUARE GARDEN IN 2003 AND PRODUCED THE COMMERCIAL FOR MIND SHARE VENTURES. FOR MORE INFORMATION ABOUT LUMINETIK, PLEASE GO TO ***WWW.LUMINETIK.COM.***

BY GREG LEMON

RIGGING: MASTER OF PUPPETS

RIGGING A VIDEO GAME CHARACTER USING THE FIGURE FROM THE TUTORIAL STARTING ON PAGE 24

Character rigging can be a difficult and frustrating task to 3D artists of all levels. Developing a stable, efficient rig takes knowledge, practice, and common sense. This article is designed to help further your knowledge of rigging, and assumes you have a basic understanding of concepts such as joint chains, IK and skinning. We'll be using Maya for all of our examples, but the topics discussed can be applied in almost all 3D software packages.

I'm going to talk about the rigging methods that I use that work for me. We'll use a golfer character to illustrate our ideas, starting from scratch and working towards a fully rigged character. There is almost always more than one way to do anything in 3D, so just because I like to do something one way doesn't mean it's the only way to do that particular task. Having an open mind about new 3D techniques and being able to think "outside the box" will help you grow immensely as a digital artist.

There are many qualities that a good rig has, but the main concepts I want to address are good naming conventions, fast speed, stability, pick ability, and zero ability. We'll also talk about how to correctly draw skeletal joints at odd angles, get good deformations quickly, and set up a curve-based rig that will act as a simple, intuitive control structure for our character.

When something breaks, and you can't tell what it is by looking at its name, you'll have a harder time finding and fixing it. By naming your nodes well, you can avoid many problems and help ensure you always know what part of the body you're working with. I use a naming convention that allows me to tell what type of node something is, its description, orientation and iterations.

TYPE_description_orientation_iteration

Examples:

JT_index_R_1 = first joint of the right index finger
CTL_head = controller for the head
PS_golfer = golfer polygon geometry

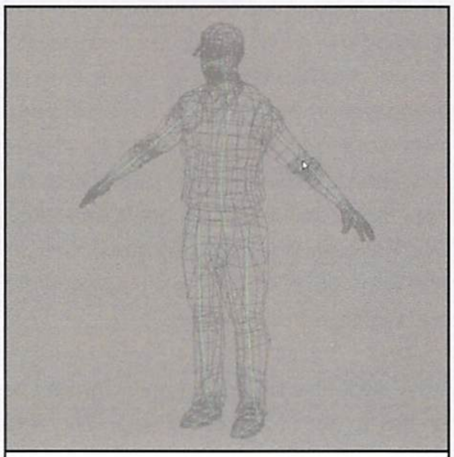
When we talk about a "fast" rig, we're talking about how heavily its network of interconnected nodes affects your playback speed and interactive manipulation. A slow rig will chug along at a slow frame rate, thus slowing down your workflow. Most 3D software these days can deal with fairly complex rigs and models and still play back in real time, but going overboard with a huge cloth simulation or super-dense model can slow things down to the speed of molasses. If you do need an effect as complex as the ones above, make sure you give yourself a way of hiding it if it's not needed. One common trick is to create pieces of low-res geometry from your high-res mesh and parent them to their respective bones. Once this is done, a switch is created that toggles the visibility of the low-res/high-res model, so that an animator can see the shape of their character without having to sacrifice speed.

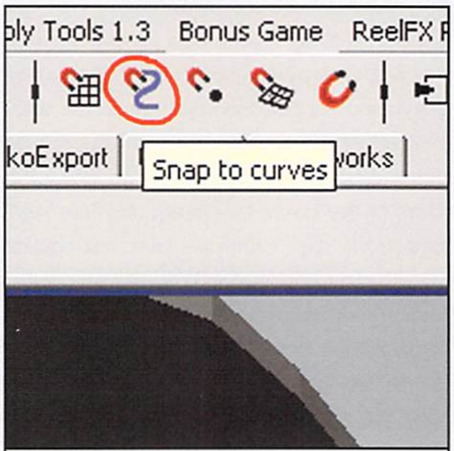
So you've got a fast rig, eh? So fast that you're almost done with your animation! Until ... your character explodes. Stability is a simple theory; your rig should not fall apart when animated. Ensuring stability in your rig is best learned through trial and error, but you can avoid some common problems by planning ahead. Think about how you're going to rig your character before you dive in. For the golfer, I thought about how his golf club would have to work with his hands, and vice versa. Analyzing this complex relationship helped me pre-visualize roadblocks and solve puzzles before they became problems. Another part of stability is knowing what works and what doesn't. As you become familiar with rigging, you will probably develop a workflow that you will use to set up most of your characters. If a technique works for you, use it. If not, don't bother. You don't have to use IK to make a great animation if you don't need it!

Pick ability refers to being able to easily select different parts of the body without a problem. Too much clutter or poorly designed controllers will be confusing when you're trying to tell the difference between your character's neck and wrist controls. One technique I like is using curves shaped like icons, such as arrows and cubes, as control objects. In addition, try to make sure that things you don't want to accidentally select are hidden and have their channels locked. Zero ability is an often overlooked but important concept. If you select all of your character's controls and "zero" each attribute, your character should return to the pose at which is was bound. This allows you to return to the bind pose if you need to in order to fix something that requires you to do so.

Our golfer is a biped, so we know what kind of skeleton he'll need right off the bat *(fig_golferModel)*. Before we start placing joints, we'll draw some curves in the side and front views that approximate the shape of his skeleton. Use a 3-degree CV curve for the spine and neck, and 1-degree linear curves for the arms and legs *(fig_skeletalCurves)*. We're dealing with a photoreal human, so we should pay attention to anatomy here, as we don't want to stray far from the shape of an actual human skeleton. The completed curves not only give us a great pre-visualization of what our skeleton will look like, but also allow us to use Maya's Snap to Curves feature *(fig_snapIcon)*, giving us the ability to draw our joints in perspective view right on the curves. This is especially helpful in drawing the arm joints, which are bent at an odd angle. Drawing them on curves will ensure that their orientations line up

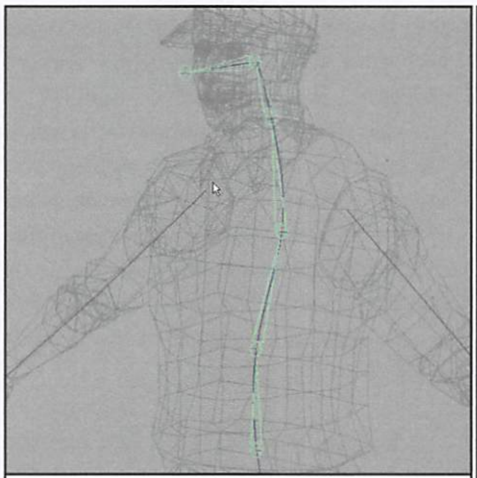
FIG_GOLFERMODEL – BIPED GOLFER

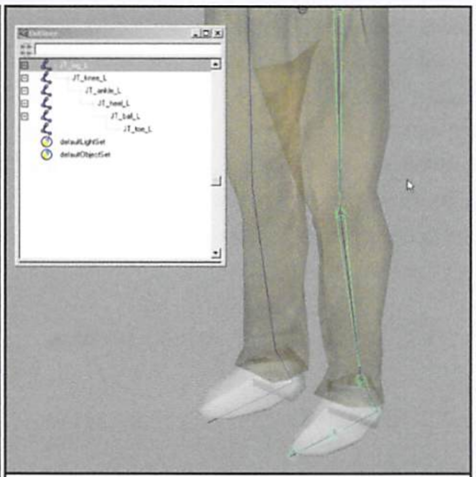

FIG_SKELETALCURVES – WIRE FRAME WITH CURVES



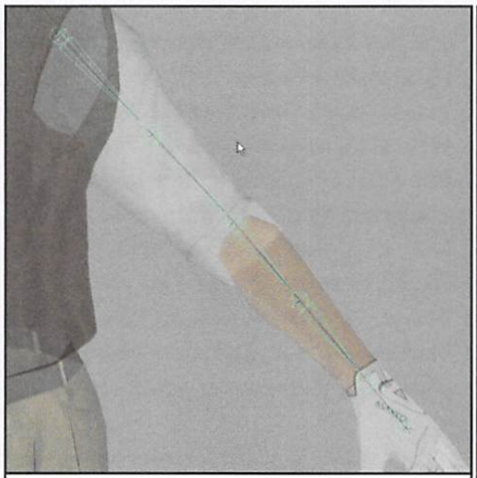
FIG_SNAPICON – MAYA'S SNAP TO CURVES ICON correctly and that the bones are pointing in the right direction. You'll want to draw the spine and each limb separately (draw the limbs for one side only, we'll mirror them later), and name each joint with a "JT_" followed by a description of its location and orientation, such as "JT_spine_1" and "JT_arm_R". See *(fig_spineJoints) (fig_leg_hierarchy)* and *(fig_armJoints)* for how our spine, legs, and arm joints look. I've used a 4-bone spine, which will work fine for this game-res character. We're going for photoreal here, but we don't need to replicate all of the joints of the human spine. Note that I've drawn an extra joint at the midpoint of the upper and lower arms *(fig_arm_hierarchy)*. These joints are there to help out the twisting motion in the shoulder and wrist areas. By animating the X rotation (twist) on these bones and tapering the weight maps between neighboring coaxial bones, we can achieve a gradual twist deformation in these areas, rather than a sharp twist at the shoulder and wrist. Once we make controllers for our character, we can use the Connection Editor to have one controller drive the vertical and horizontal motion in the shoulder and the twist in the bicep.

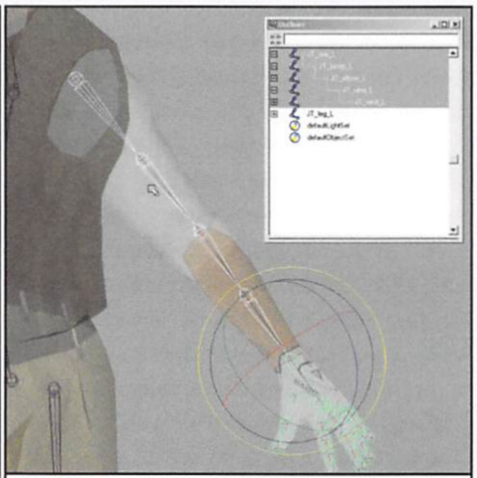
To draw the fingers, I'll use a technique similar to the curve-based one, but with planes instead. Create a NURBS plane and line it up so that it bisects the finger along its primary axis of movement *(fig_fingerPlane, fig_fingerPlane_bisect)*. Make the plane live using the magnet icon in Maya's status bar *(fig_makeLive_icon)*. Now, if you draw a joint in perspective view, it will snap to the plane *(fig_fingerPlane_bisect_makeLive)*. We can use this to easily draw finger joints in an area that would be otherwise difficult. When using this method, it helps to first draw a "dummy" joint off to the side of the finger (but still on the plane), and then continue to draw the joints of the chain. Un-parent the first "real" joint from the "dummy" joint and delete the dummy *(fig_fingerPlane_dummy_joint)*. Placing this first throwaway joint helps orient the next joint correctly when using this helpful method. Once you've completed a finger, click the magnet icon again to make the plane unlive. Repeat for the rest of the digits, until you have a complete hand skeleton.

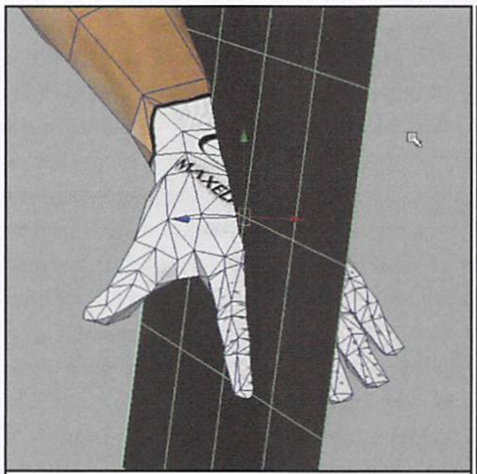

FIG_SPINEJOINTS

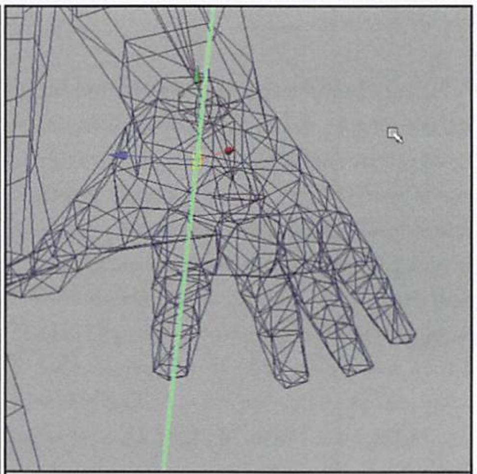


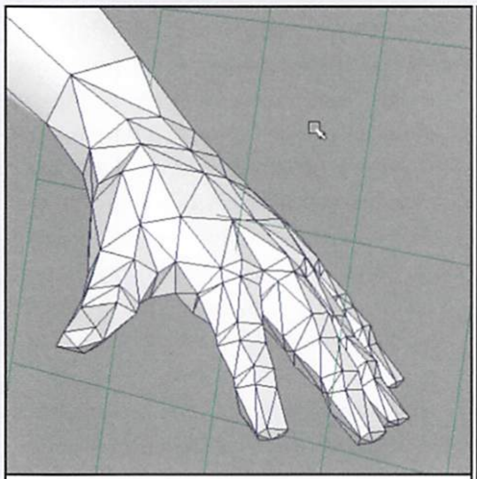
FIG_LEG_HIERARCHY

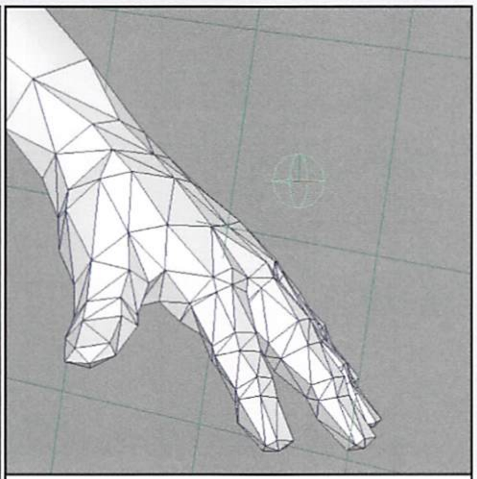

FIG_ARMJOINTS


FIG_ARM_HIERARCHY


FIG_FINGERPLANE


FIG_FINGERPLANE_BISECT

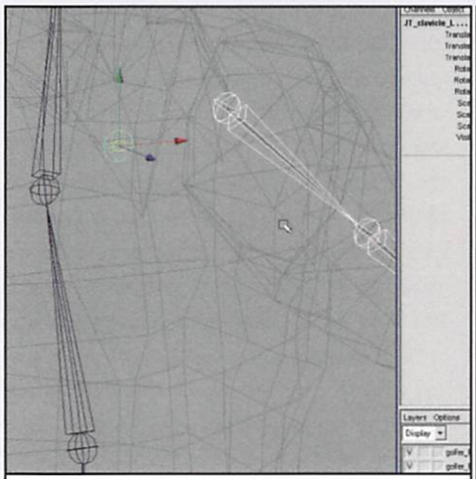

FIG_FINGERPLANE_BISECT_MAKELIVE

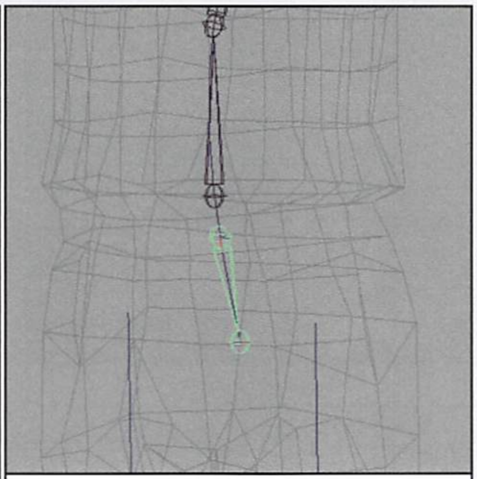

FIG_FINGERPLANE_DUMMY_JOINT

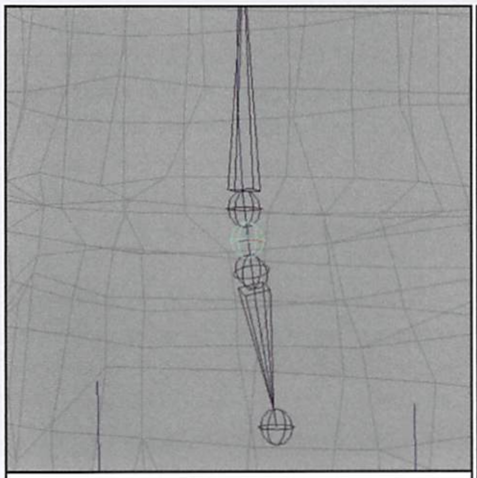
Create a single shoulder joint (not a full bone, just a joint in space) and position it near the character's spine, slightly below where the arm starts *(fig_clavicle_joint).*

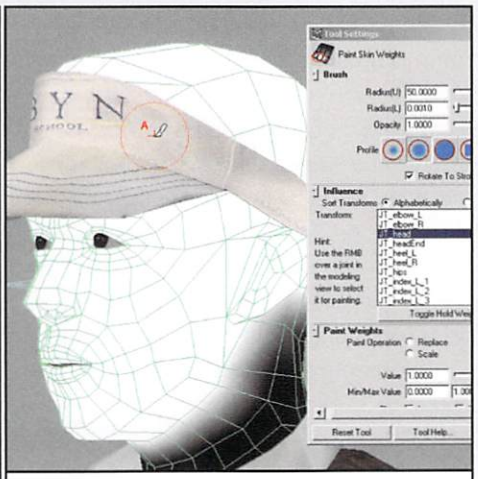
You can also create a hip joint below your character's spine, which will give you more control over your character's pelvis *(fig_hipJoints).* Parent both the first spine joint and the hip joint to a central joint, which will be your character's root *(fig_rootJoint).* Parent the fingers to the wrist, the arm to the shoulder to the upper spine, and the legs to the hips. Mirror the limbs to the other side using *Skeleton>Mirror Joint*). Maya 5's Mirror Joint function allows you to search and replace parts of your hierarchy names for the mirrored side, so you don't have to type "_L" over and over.

Now that we have our skeleton, it's time to bind our geometry to it. We'll use a Smooth Bind, which will allow us to use the Paint Weights tool later. Select all of your character's geometry, shift select the root, and go to the *Skin>Bind Skin>Smooth Bind* option box. Leave the settings at default, except for Max Influences, which we'll set at 1. This will skin each point directly to its closest bone, creating areas of full influence now that we'll smooth out later. We'll start with the head and work our way down the torso, using the Paint Skin Weights tool *(Skin>Edit Smooth Skin>Paint Skin Weights Tool)* to block solid areas of full weight for each bone *(fig_paintSkin_head)* *(fig_paintSkin_neck).* Paint the limbs on one side only, as we'll

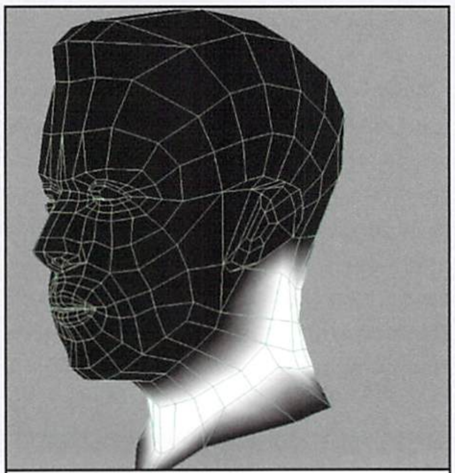

FIG_CLAVICLE_JOINT

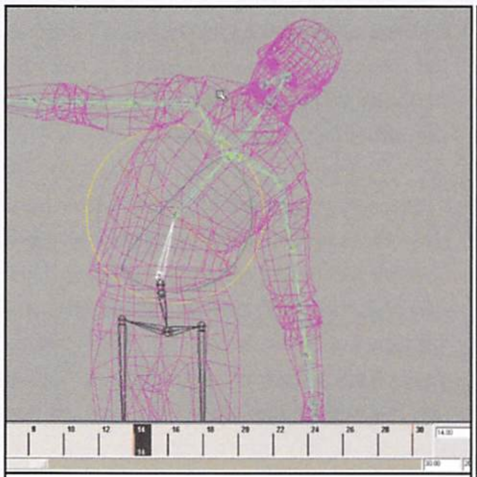

FIG_HIPJOINTS

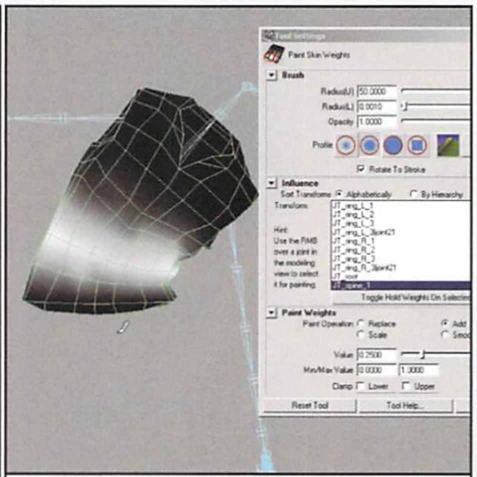

FIG_ROOTJOINT


Tool Settings
Paint Skin Weights
Brush
Radius(U) 50.0000
Radius(L) 0.0010
Opacity 1.0000
Profile
☑ Rotate To Stro
Influence
Sort Transforms ● Alphabetically
Transform: JT_elbow_L, JT_elbow_R, JT_head, JT_headEnd, JT_heel_L, JT_heel_R, JT_hips, JT_index_L_1, JT_index_L_2, JT_index_L_3
Hint: Use the RMB over a joint in the modeling view to select it for painting.
Toggle Hold We
Paint Weights
Paint Operation ○ Replace ○ Scale
Value 1.0000
Min/Max Value 0.0000 1.00
Reset Tool | Tool Help...

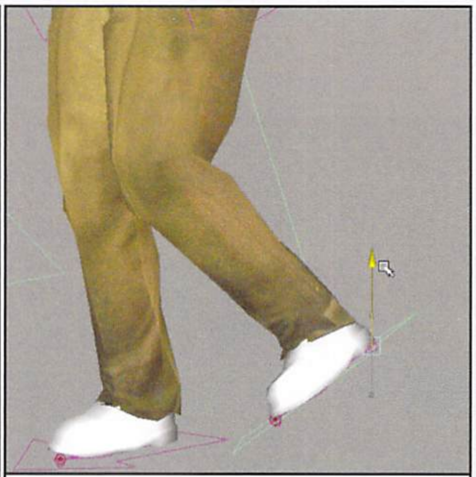
FIG_PAINTSKIN_HEAD


FIG_PAINTSKIN_NECK


FIG_PAINTSKIN_TORSOBEND


FIG_PAINTSKIN_SPINE

FIG_PAINTSKIN_TORSOBEND


FIG_PAINTSKIN_SPINE

mirror the skin weights later on. Make sure you are using the "Add" option under Paint Operation and that the value is set to one. You can interactively change your brush size while painting by holding the *B* key and left-click dragging with your mouse.

Many people wait until their character is completely set up before they animate anything. In truth, animating your character while skinning is a great way to speed up your workflow! By quickly doing some simple bending and twisting animations with your character, you can see how the deformations look while he's in action *(fig_paintSkin_torsoBend)*. You can also paint weights while your character is in problem-causing poses, which can fix nasty deformations very quickly.

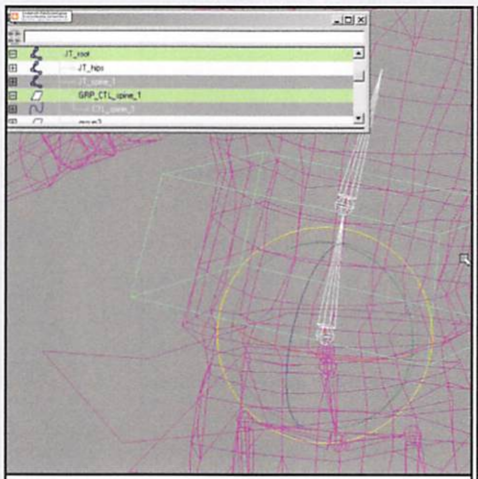
Once we've got the skinning looking good (on one side of the body, at least), it's time to go back in and blend the weights between different areas. Using Paint Weights, set your Paint Operation to add and paint some additional weight from neighboring bones on your maps *(fig_paintSkin_spine)*. This is better than using the Smooth function, which tends to smooth the weights out across the whole skeleton, resulting in messy super small influences on unwanted body parts. Clean up your work, and use Maya's Mirror Skin Weights tool to flip the weight to the other side. Make sure your character is in the bind pose when you do so, or the mirror operation may not work correctly.

At this point, we should have a fully functioning golfer with good deformations. Don't worry if the skinning isn't as good as it could be, as we can always go back and paint weights during the animation process. Now it's time to build the control structure that we'll use when we're animating. Now is a good time to think about the actions your character will need to perform during your animation, which can help you determine whether or not to use FK or IK.

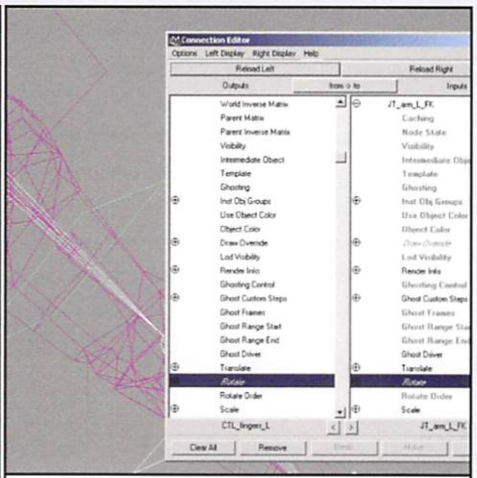
Forward kinematics, or FK, is based on the idea of rotating bones. You select your bone, rotate it, key it, and move on to the next one. The concept is much like that of a sock puppet, where you rotate your wrist and forearm to drive your character's motions *(fig_FK_example)*. In most situations, this is the best way to go, as it gives you precise control over each joint's rotations and allows you to really fine tune your motions. However, when you need the end of a limb to stay still while the rest of the body moves, FK can be a real nightmare. A good example of this is a walk cycle, where your character's feet need to stay in contact with the ground while the body moves forward. This is where IK comes in, as it allows automatically driving a joint chain by moving a target, called an IK effector. If the rest of the body moves, the foot will stay planted until the effector is moved. This is like working with a marionette, where the strings attached to the hands and feet cause the limbs to bend *(fig_IK_example)*. Both FK and IK can work well in different situations, and a good rigger will often find ways of incorporating both into a rig, allowing the animator to choose which one to use. We'll draw a simple IK system on each leg. Using the IK handle tool (make sure it is set to rotate plane instead of single chain), click on the first joint in the leg, then the ankle. Draw a second IK chain from the ankle to the toe.

What we have now is a low-level character rig that gives us a basic level of control for our character. We have everything we need in order to animate a basic walk: bones that our character is skinned to, and IK chains for the legs. While we could go ahead and start setting keys, it might be a good idea to consider developing the rig a bit further. Investing a bit more time into this will allow us to create a rig that will be more stable, fixable, and flexible than what we have now. Let's start with creating controllers, which are the nodes we'll select when we want to work with different parts of the body.

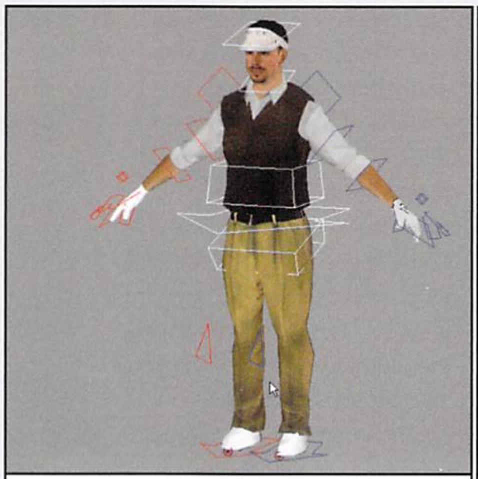
We'll use curve-based controllers (rather than locators) to drive the motion of our FK bones and IK systems. As I stated before, iconographic curves are more easily identified and selected than locators. I usually try to use simple geometric shapes that can be easily identified and selected for my controllers. Circles, squares, and arrows work just fine, and can all be easily drawn with the CV curve tool. After creating a control shape, name it "CTL_," followed by a description of its purpose and orientation, such as "CTL_spine_1." Group it by itself *(Ctrl + G)* to create another pivot, and use Maya's snapping function, or a snapping script, to move and align the group to the first spine joint. Parent the group to the parent of that joint, so that the controller and spine_1 joint exist in the same local space and share similar orientation. If the curve doesn't line up the way you want, select its points and move and rotate them into the shape you want, which won't affect the controller's position *(fig_example_spineController)*.



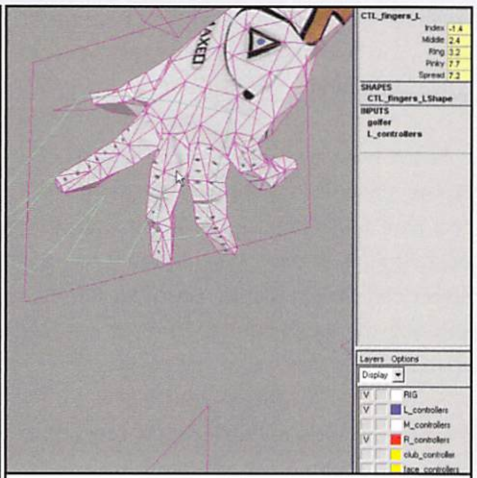
FIG_EXAMPLE_SPINECONTROLLER



FIG_CONNECTION_ED



FIG_RIGFINISHED



FIG_FINGERCONTROL

We can now use Maya's Connection Editor *(Window>General Editors>Connection Editor)* to create a direct relationship between the curve controller's rotations and the joint's rotations. Load the controller into the left window and the joint into the right, and select rotation for the controller and the joint *(fig_connection_ed)*. For the legs, we'll make arrow-shaped foot controllers and parent the IK chains to the controllers. We'll also add a "pole vector" controller, which will allow us to control the orientation of the knee. Place the controller directly in front of the knee, shift-select the leg to ankle IK, and go to *Constrain>Pole Vector*. Parent the controller to the foot controller, and repeat for the other side. Our rig is almost finished; see fig *(fig_rigFinshed)* for how our curve control structure looks.

Let's finish off the rig with some finger controls. Make another curve controller, and place it slightly above the hand. Using Maya's Add Attribute command (*Modify>Add Attribute*), add the following float attributes, each with a min, max and default of -10 10 and 0: index, middle, ring, pinky and thumb. Parent the control to the wrist joint, and use Maya's Set Driven Key to wire the finger rotations to the attributes. This enables us to quickly pose the hand using one controller, without having to wade through a mess of joints *(fig_fingerControl)*.

FIG_PAINTSKIN_SPINE

We now have a simple and effective rig for our character that can be reused for multiple animation projects. Additional controls for the face, club, and eyes can be added to help bring further life to out character. We can now quickly pose our character and start blocking out our animation *(fig_golferPosed)*. Our controls are zero able, so if we need to return him to the bind pose, we just need to zero out all our animatable attributes.

Here are some additional pointers that you can use to enhance your workflow:

Think about how you can use hierarchies when you animate. By creating a multi-level hierarchy with different controllers, you can separate different aspects of your movement onto different layers, allowing you to control the motion more precisely. You could have a layer that controls rotation parented to a layer that controls translation.

If you're doing facial animation, consider creating a camera that aims directly at your character's face. Parent the camera to your character's head joint, and look through it when you need to look closely at the face while animating.

Learn how constraints work and incorporate them into your rigs. Point, orient, and aim constraints are a valuable tool in your rigging arsenal. Keep in mind that an object can have multiple constraint targets, and that the weights of the targets can be animated.

Character sets are shapeless nodes that are a collection of all of your character's animatable attributes. When you create a character set *(Character>Create Character Set)*, you define a set of nodes and attributes that you want consolidated into the set. You can make the character set active by using the dropdown menu to the right of the range slider. When the character set is NOT active, pressing *S* sets a key on whatever you have selected, as it normally does. When the set it active, pressing *S* sets a key on all the attributes in the set (if you have an object selected that is not a member of the set, no keys will be set on it). This makes working in a pose-to-pose method very easy, as you can quickly key your whole character with the set on to do your blocking, and then refine the individual parts of the body with the character set off.

Set Driven Key is a powerful tool that any Maya rigger will want to learn. It employs a concept called "reactive animation," where the animation of one attribute is driven by the animation of another. A great example is using Set Driven Key to control finger animation with custom attributes. Set Driven Key, the Connection Editor, and the Expression Editor are powerful tools that you can use to build automated relationships between nodes.

Here's a really quick shortcut that can help you when you're animating. Click in the timeline with the left mouse button, and you'll move to that frame and the animation will update. Click the middle mouse button and you'll move to that frame ... but the animation won't update. Confused? Think of it as a way to travel backwards and forwards in time, without having the animation move with you. The best part? If you set a key on whatever you have selected, it will create a key with the displayed value intact.

GREG LEMON IS CURRENTLY A PART OF *WWW.ALCAZAR-ENTERTAINMENT.COM* WHICH HE, EDDIE ALCAZAR AND TWO OTHER INDUSTRY FRIENDS STARTED IN EARLY 2004. SINCE THEN THEY HAVE BEEN BUSY WORKING ON AAA GAME TITLES, COMMERCIALS AND OTHER 3D WORKS. "WE STRIVE TO PRODUDE WORK OF THE HIGHEST QUALITY, WHILE STILL HITTING OUR DEADLINES AHEAD OF SCHEDULE. WE ALWAYS FIND A BALANCE BETWEEN PUSHING THE LIMIT AND DELIVERING ON TIME"

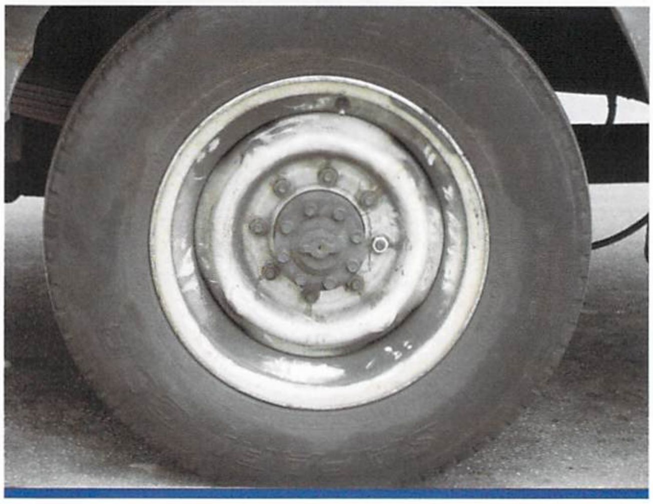
GREG BEGAN HIS 3D ANIMATION CAREER AS A PROFESSOR OF COMPUTER ART AT THE *SAVANNAH COLLEGE OF ART AND DESIGN*. HE TAUGHT THERE FOR A YEAR AND A HALF. FROM THERE, GREG SIGNED ON AS AN ANIMATOR/RIGGER ON *X 2: X-MEN UNITED* WHERE HE WORKED ON SEVERAL OF MYSTIQUE'S MORPHING SEQUENCES. FROM THERE, GREG MOVED INTO THE WORLD OF GAMING, WORKING ON CINEMATIC ANIMATIONS FOR EA'S *MEDAL OF HONOR* EXPANSION PACK.

GREG RESIDES IN SAN FRANCISCO, AND LIVES WITH HIS GIRLFRIEND. HIS ONLINE PORTFOLIO CAN BE SEEN AT ***WWW.GREGLEMON.COM***

BY DAN ABLAN

Building from Images with Modo

AN INTRODUCTORY TUTORIAL



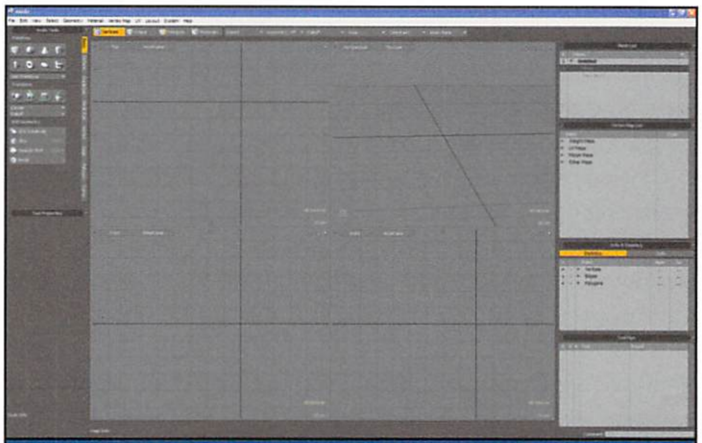


FIGURE 1– THE DEFAULT MODO INTERFACE.

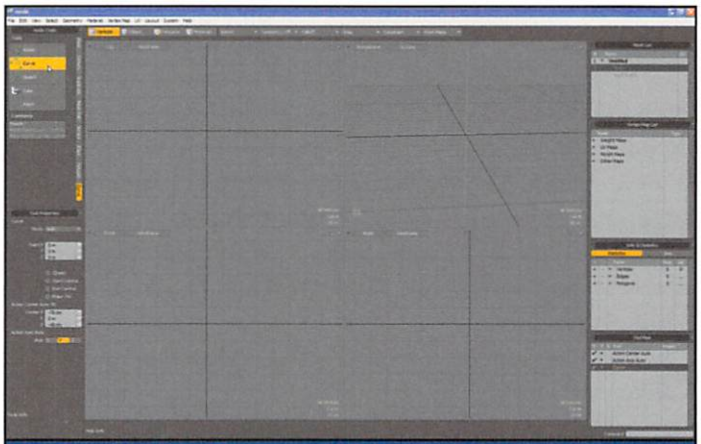


FIGURE 2– MODO ALLOWS YOU TO BUILD OBJECTS WITH CURVES. SELECT THE CURVES TOOL AFTER FIRST CHOOSING THE CURVES PANEL

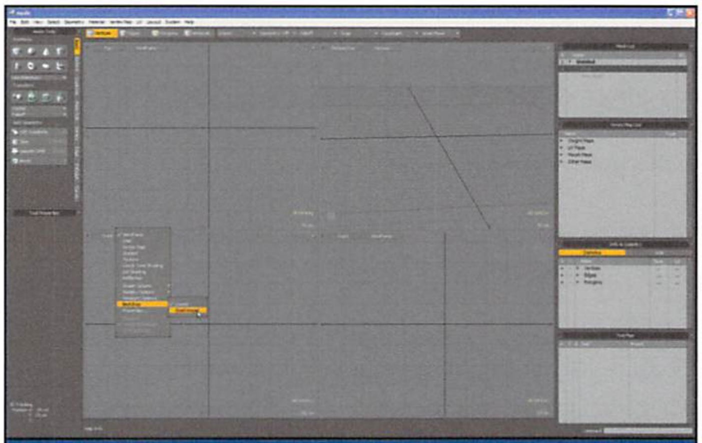


FIGURE 3– QUICKLY ADD A BACKDROP IMAGE IN MODO FROM THE VIEWPORT DISPLAY OPTIONS

After two long years, Luxology, Inc. has released their first product, modo. It is a highly anticipated 3D modeling application that is one of the most advanced polygonal and subdivision surface modeling tools ever created. Aside from having one of the most flexible and easy to use interfaces on the market, modo has a very clean and easy workflow. We'll be adding modo tutorials in future issues of HDRI 3D, so be sure to check them out. But instead of talking about this program, we thought we'd include an introductory tutorial so you can see just how this program works.

You'll start this tutorial in a default modo layout. Modo gives you the ability to create any possible layout arrangement you can think of, and access it at anytime. I've been able to set it up on a dual-screen system with a quad view and tools in one monitor, while the other monitor has just a full frame perspective view. *Figure 1* shows a default layout when modo is started.

You can change the arrangement of panels and views to anything you like. Modo may default to a single perspective view; if so, change it to a quad view from the top right of the panel.

1 At the top left of the modo interface you'll find the modo Tools list. Modo allows you to create with primitives, points, or curves, as you would with most modelers. Click the Curve button to change the modo Tools to Curve options. This is the tall vertical list of tools lined up to the left of the top viewport. Once you click curves, select the Curve tool, as shown in *figure 2*. You can see that you have the option of using Bezier, Sketch, or a Tube option.

2 Modo allows you to use colored backdrop images for reference to create 3D models. In the bottom left viewport, click on the viewport display icon, which is the second label at the top left of each viewport. Once the options appear, go down to the Backdrop option and choose Load Image, as in *figure 3* on the next page. Load the oldwheel.jpg image from 3dgarage.com/downloads, or use one of your own.

COOL TIP: Another easy and cool way to add images to your backdrops in modo is to simply drag and drop them from another directory on your system.

3 With the image loaded in the backdrop, go to View at the top of modo and choose Adjust Backdrop Image. On the lower left of modo, you'll see the Backdrop Image Controls appear, as in *figure 4*.

4 One thing that's cool with these tools is that you can interactively adjust the image. Click directly on the image and you'll see a bounding region appear. Click and drag a corner to size the image. Click in the center to move around, and use the Ctrl key to constrain movements. You can also set the Display Resolution depending on the strength of your video card. Conversely, you can click and drag on the sliders in the backdrop for properties to rotate, set Transparency levels, and more. Experiment! When ready, set Transparency to 50%. Press the spacebar to drop the tool.

5 Move your mouse cursor over the backdrop image and press 0 on your numeric keypad. This will make the view full screen. Then, press *a* to fit it to view.

6 From the Basic tool set, choose the Cylinder tool, as shown in *figure 5*.

7 In the viewport, hold the Ctrl key and drag out a disc over the image. If you have trouble seeing the geometry, simply go to View at the top of modo, and choose Adjust Backdrop Image – then change the Transparency. Once you draw out the cylinder shape, you'll see that it's a bit chunky. Don't drop the tool yet, and in the Tool properties on the left, up the sides to 50.

8 Once the cylinder is to your liking, press the spacebar to drop the tool.

9 Press 0 on your numeric keypad to return to a quad view.

10 You'll see that because you created a cylinder, it's extended out down the Z-axis. Press the spacebar a few times so that you bring modo to Edges mode (shown at the top of the viewports). Then, with the right mouse button, lasso select around eight rows of polygons, as shown in *figure 6*.

11 With the rows of the object selected, simply press the delete key on your keyboard to remove them.

12 Now here's a cool feature – in the Perspective view (top right), add the backdrop image to the background, and size it up accordingly to match your original placement in the bottom left view. To figure out what size your other image is, go to View at the top of modo, and choose Adjust Backdrop Images. Then, click on the image in the front view – look at the tool properties on the left. Note the size. Then, click on the image in the Perspective view, and enter the same size value. Next, press the spacebar to jump to Polygon mode, and then in the top right viewport, select the front of the disc. *Figure 7* shows the image now in Perspective view.

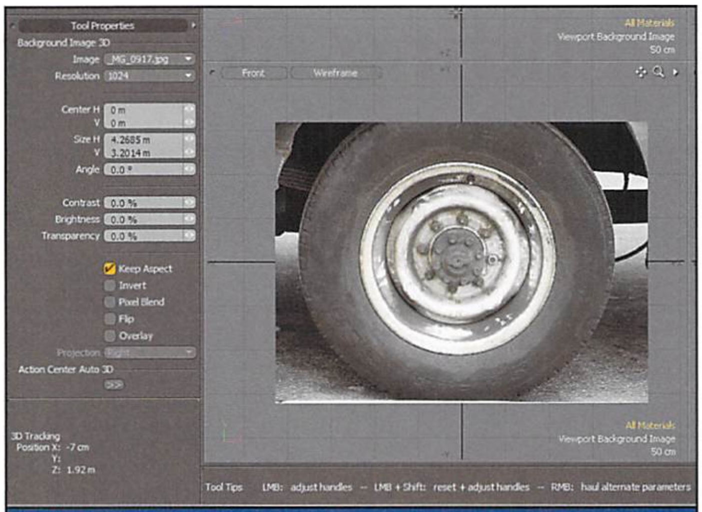

Tool Properties
Background Image 3D
Image _MG_0917.jpg
Resolution 1024
Center H 0 m
V 0 m
Size H 4.2685 m
V 3.2014 m
Angle 0.0 °
Contrast 0.0 %
Brightness 0.0 %
Transparency 0.0 %
✓ Keep Aspect
Invert
Pixel Blend
Flip
Overlay
Projection Right
Action Center Auto 3D
3D Tracking
Position X: -7 cm
Y:
Z: 1.92 m
Tool Tips LMB: adjust handles – LMB + Shift: reset + adjust handles – RMB: haul alternate parameters

FIGURE 4– BACKDROP IMAGES HAVE A LOT OF CONTROL IN MODO

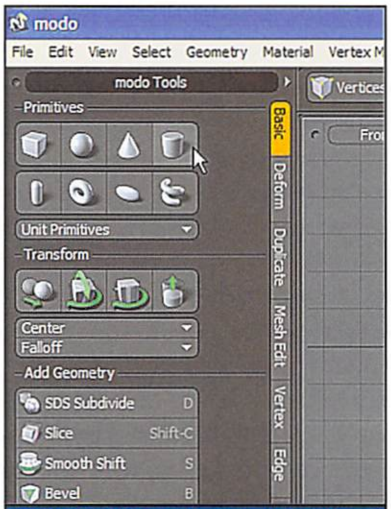


FIGURE 5– MODO OFFERS A WIDE RANGE OF PRIMITIVE SHAPES TO HELP YOU BEGIN MODELING MODO

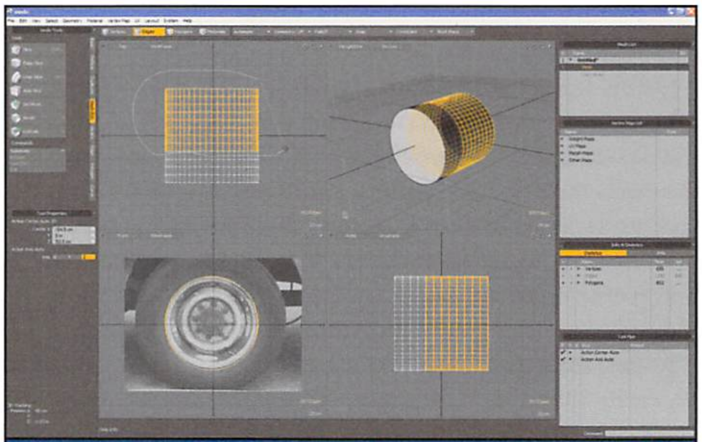


FIGURE 6– EDGE TOOLS AND SELECTIONS ARE ONE OF MODO'S STRONG POINTS, AND IT'S EASY TO QUICKLY SELECT AND REMOVE UNWANTED GEOMETRY

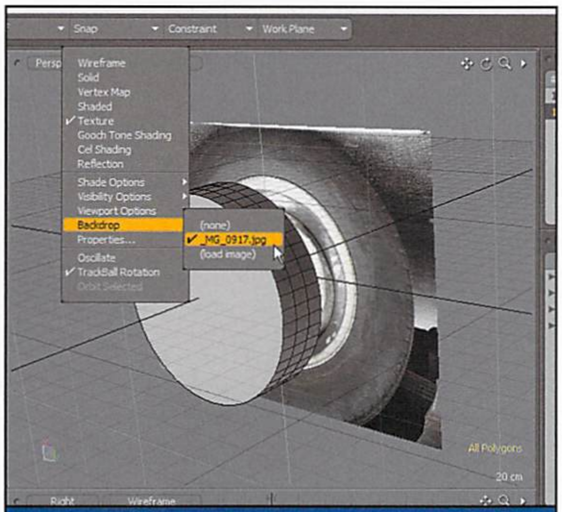


FIGURE 7– YOU CAN ADD AN IMAGE TO A PERSPECTIVE VIEW FOR ADDED MODELING REFERENCE

COOL TIP: When adjusting backdrop images, you can change the way the image is projected. Notice in the Backdrop Image tool properties panel, you can change projection to front, right, top, bottom, and so on.

13 Change the Perspective view to Wireframe view. Make sure the first polygon on the disc is selected (be sure you're in Polygon mode at the top of modo). Then press the *b* key to activate the Bevel tool, as in *figure 8*.

14 When you're working in Bevel mode (or any other), clicking on the selection brings up handles. Click the blue handle in this case to shift the bevel, and the red handle to inset. Bevel the selection to match the background image so that you create the first edge of the wheel.

15 Next, press *b* to turn off the tool, then press *b* to turn it on again. Now bevel the same selection inward. *Figure 8b* shows the operations.

16 Now repeat the above steps a few more times so that you continue beveling, using the backdrop images as guides. *Figure 9* shows the results.

17 Save your work by pressing *Ctrl-S*. Then, press to set a material name for the wheel. Enter something original like, "wheel." To set a color, you can simply click and drag on the values, or click once to call up your system's Color Corrector. Give it a bit of Specularity and you're good to go.

18 At the top of modo is the Material dropdown. Select this to choose Material Editor, as shown in *figure 10*.

19 In this panel, you can adjust and assign various materials to your models.

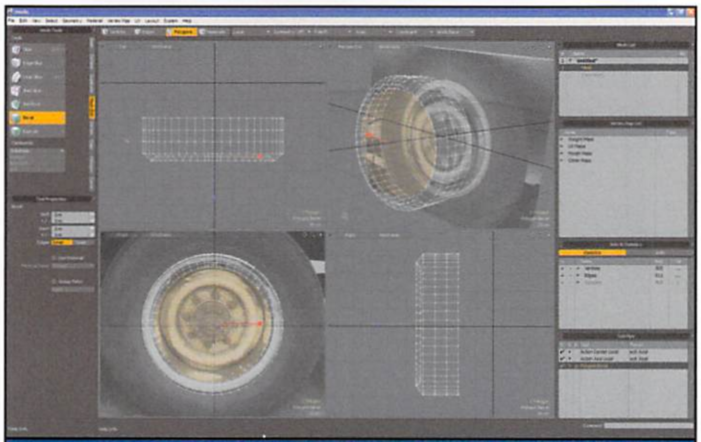


FIGURE 8– BEVELING IS JUST ONE WAY TO BEGIN CREATING SHAPES FROM PRIMITIVES

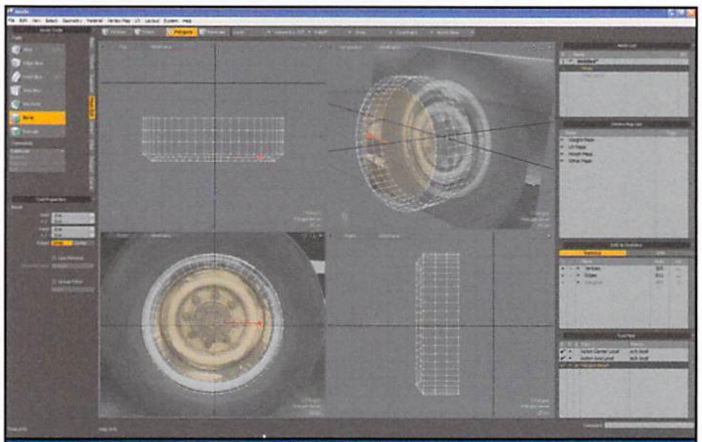


FIGURE 8B– AFTER TWO BEVELS, YOU CAN BEGIN TO SEE THE SIMPLE DISC TAKING SHAPE

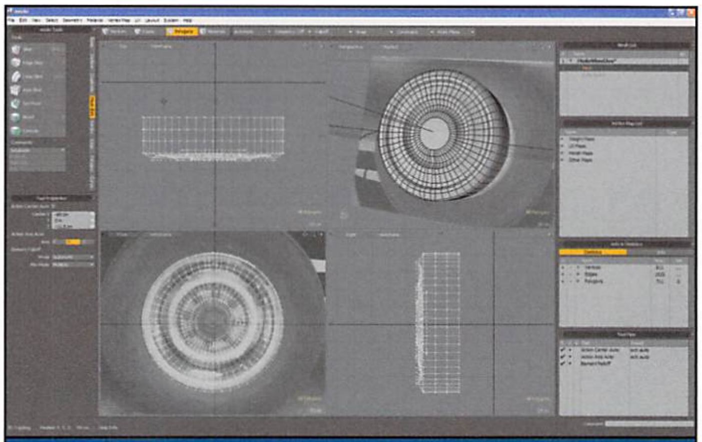


FIGURE 9– YOU CAN ADD AN IMAGE TO A PERSPECTIVE VIEW FOR ADDED MODELING REFERENCE

20 At the top of the Material Editor you'll see M, T, I – M for material, meaning the material name you're working with. In this case, you've only created one single material, wheel. If you had more, you could choose them from the dropdown list. The T is for a new Texture, and I for Image. Select T, and from the dropdown to the right, choose New Texture. Load the oldwheel.jpg image, or the image you used for the wheel reference. Click OK, and you'll see the image mapped on the wheel as in *figure 11*.

21 Close the Material Editor, and in the Perspective view, you can click and drag on the red and green handles to easily adjust the size of the image to line up with your model. Click and drag the light blue handle to move the image. *Figure 12* shows the placement in the Perspective view.

22 From the viewport render style selection, you can change the shade options to not show wireframes, making your image map easier to see. *Figure 13* shows the selection.

The next step would be to build out the tire itself, and join it with the wheel. You can do this using a combination of bevels, either on the polygons or the edges. Additionally, you can try building with the Lathe tool from a simple curve. Try creating a curve by going to the Curve panel (the tall vertical panel to the right of the tools) and then, using either just Curve or Bezier Curve, build the shape of a tire. Then use the Lathe command.

But there are multiple ways to build objects in modo. This tutorial was about as basic as you can get, but it can give you an understanding of the workflow and interface. Our future tutorials will really get into modo deeper, through the use of edges, subdivision surfaces, use of the modo tool pipe and so much more. Please send your modo suggestions to us so we can create the tutorials you want. Until then, happy modeling!

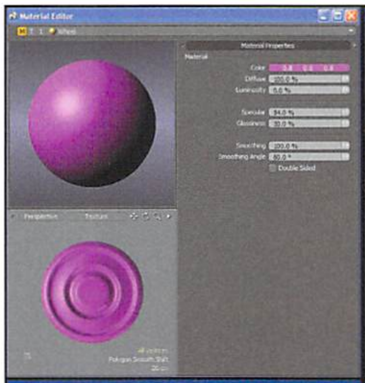


FIGURE 10– THE MATERIAL EDITOR OFFERS UNIQUE CONTROL FOR SURFACING AND TEXTURING YOUR OBJECTS

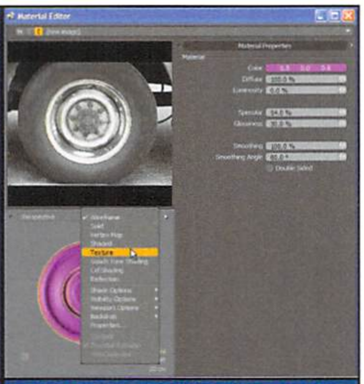


FIGURE 11– THE MATERIAL EDITOR ALLOWS YOU TO CREATE AN IMAGE MAP FOR THE COLOR, DIFFUSE, SPECULARITY, AND MORE FOR YOUR MODELS

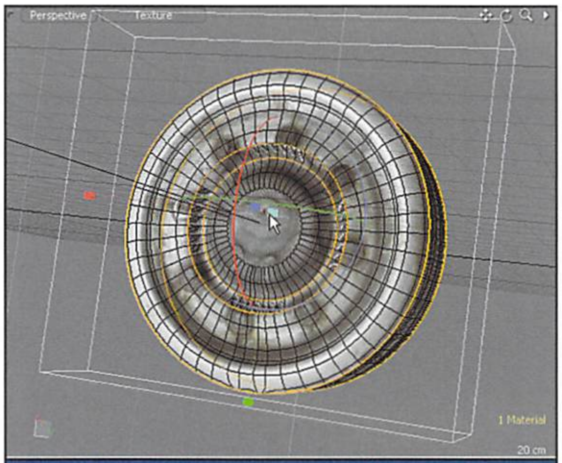


FIGURE 12– ONCE AN IMAGE MAP IS APPLIED, YOU CAN EASILY MANIPULATE IT IN MODO'S VIEWPORTS

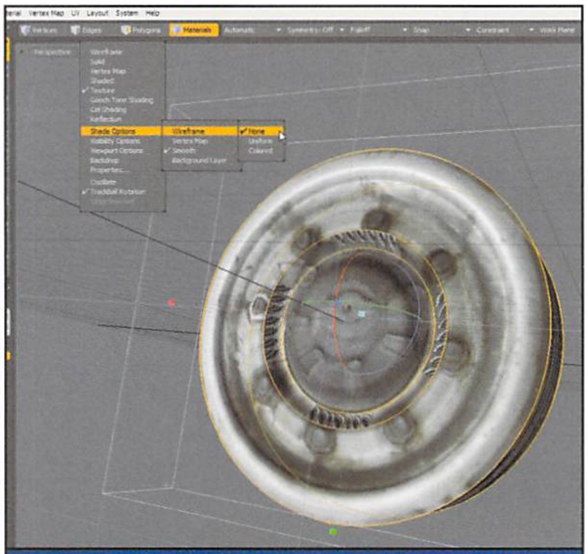


FIGURE 13– VIEWPORT RENDER STYLES ARE WIDE RANGING. HERE, YOU CAN SEE THAT THE VIEWPORT HAS A TEXTURE VIEW VISIBLE, BUT THE WIREFRAMES CAN ALSO BE TURNED ON OR OFF AT THE SAME TIME

DAN ABLAN IS PRESIDENT OF AGA DIGITAL STUDIOS, INC. IN CHICAGO, IL., AND FOUNDER OF 3DGARAGE.COM. HE HAS WRITTEN EIGHT BOOKS FOR *NEW RIDERS PUBLISHING* ON 3D AND COMPUTER GRAPHICS, AND HAS RECENTLY PUBLISHED LIGHTWAVE [8]: KILLER TIPS (WITH RANDY SHARP). DAN HAS CREATED A 20 HOUR LIGHTWAVE COURSE ON CD-ROM, YOU CAN FIND OUT MORE ABOUT THE COURSE AT 3DGARAGE.COM. DAN IS ALSO THE EDITOR-IN-CHIEF OF HDRI 3D MAGAZINE.

BY KENNETH IBRAHIM

COMMANDING AN AUDIENCE: THE MAYA API COMMAND PLUG-IN

API INTRODUCTION

The Maya API (Application Programmer Interface) is a C++ based interface that provides a great deal of convenience and power to the Maya user. Daunting at first, many shy away from its potential and speed and instead work exclusively with MEL to develop the various tools needed for a project. The intent here is to take some of the mystery out of the use of the API and to demonstrate by example what can be relatively easily accomplished. The API provides access to most of Maya's underlying data structures and offers many built-in data manipulation methods, some of which appear in various MEL commands and some of which don't. It also offers a vast speed improvement (approximately 10x) over MEL, as API plug-ins are compiled rather than interpreted and won't have the overhead many MEL commands do. Because plug-ins are compiled, another program called a "compiler" must be used to create the plug-in from the source code. Compilers are free software in the world of Linux (*gcc* being one I've used recently), but *Microsoft Visual Studio* for Windows is rather costly, and I'm not sure what the situation is on the Macintosh. This article doesn't cover compilers or the compilation process; rather, it describes the development of a plug-in via its source code independent of the compilation environment. The API provides for cross-platform compatibility, and getting a plug-in developed on Linux to compile and run on Windows, for example, is effortless. Very little change, if any, is required for the source code. The Maya distribution ships with extensive API documentation, including an introduction, full C++ class descriptions, compiler issues, and a number of examples. It also provides a growing FAQ section which clears up a lot of development questions. I highly recommend familiarizing yourself with that information.

The API allows for development of a number of different plug-in types, including the following:

MEL Command—Implements new functionality with undo/redo capability

Dependency Graph (DG) Node—Implements nodes that take data as inputs and calculates new values for outputs

Locator Node—Same as DG node type with additional support for OpenGL drawing into the Maya viewports

Shader Node—Implements shading nodes such as materials, textures, etc.

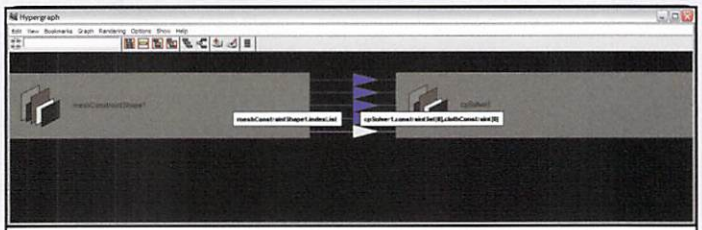
Shape Node—Implements new geometry types with selectable and modifiable components

File Translator—Provides an interface to read and write proprietary data via the *File>Open/Save* menus

Manipulator—Allows for custom manipulators to modify attributes of node plugs

Deformer—Implements a new deformer type fully integrated into the Maya architecture

In this example, we will develop a MEL command plug-in that will work in conjunction with a small MEL wrapper script to improve user workflow when using Maya Cloth constraints.


Hypergraph
Edit View Bookmarks Graph Rendering Options Show Help
meshConstraintShape1
cpSolver1
meshConstraintShape1.indexList
cpSolver1.constraintSet[0].clothConstraint[0]
IMAGE 1– THE "INDEXLIST" VERTEX INDEXES DATA PLUG

THE COMMAND

I was working quite a bit with Maya Cloth on a certain project (involving a big green guy) in which I made heavy use of cloth constraints, which are composed of particular sets of vertices. After having laboriously picked the points I wanted to use in the constraint, I would often make the mistake of creating the constraint without first saving the set of vertices as a *Quick Select Set* for later use. If I then needed to add or remove vertices from the constraint, I was up against a painful task. Even adding a single vertex to the constraint would involve deleting the current constraint, reselecting all of the vertices I originally had plus the additional vertex, and then creating the new constraint. With constraints composed of a small, easily selected set of vertices this might be acceptable, but remembering which points were part of a densely packed region was impossible. So I did a little investigating in the *Hypergraph* and discovered that there is a certain connection between the constraint shape node and the Cloth solver node called "indexList" *(shown in Image 1)* that sounded promising as a list of the vertices involved. Using the *getAttr* command with the *–type* flag confirmed that the attribute was an array of integers; the indexes to the vertices of the constrained cloth mesh, I supposed. So I decided to create a command that would simply pass back the list of constrained vertices that I could then deal with via a simple MEL wrapper script to create the modified constraint. We'll see the MEL script after discussing the plug-in code.

Each type of plug-in is based on a provided class and requires certain methods (C++ functions) in the class to be overwritten. For a command type plug-in, the base class is called *MPxCommand* and the set of methods are as follows (those marked * are optional):

constructor—called when node is created

destructor—called when node is actually deleted

doIt—performs the actions of the command but often is used to parse the argument list and then call redoIt

redoIt*—performs the actions of the command

undoIt*—undoes the actions of the command

isUndoable*—determines if the command supports undo

creator—called when a new instance is instantiated

We'll describe each of these methods as we get to them in the code.

SOURCE CODE AND EXPLANATION

We begin by including all of the header files we'll need. This will include the *MFnPlugin* and *MPxCommand* headers for this type of plug-in, as well as some of the other common headers, including *MArgList* for retrieving the arguments passed to the plug-in. The *MSelectionList*, *MFnDagNode*, *MDagPath*, and *MPlug* classes provide the mechanism to extract the desired DAG objects from the Maya scene file. Because this plug-in deals with arrays of integers (the vertex indexes), we call upon the API's *MFnIntArrayData* class to access this data.

```
#include        <maya/MFnPlugin.h>
#include        <maya/MPxCommand.h>

#include        <maya/MArgList.h>
#include        <maya/MFnDagNode.h>
#include        <maya/MDagPath.h>
#include        <maya/MPlug.h>
#include        <maya/MSelectionList.h>
#include        <maya/MFnIntArrayData.h>
```

The next step is to create our command as a subclass of the MPx-Command class.

```
class getCpConVerts : public MPxCommand
{
    public:
                                        getCpConVerts();
            virtual                     ~getCpConVerts();

            MStatus                     doIt(const MArgList &args);
            MStatus                     redoIt();
            static void  *creator();

    private:
            MString                     conName;
};
```

As you can see, I've named the command getCpConVerts as a contraction for "get cloth plug-in constraint vertices" and I've added a single instance attribute called conName that I'll use to identify the constraint. I haven't defined the isUndoable and undoIt methods, as I chose not to implement "undo" functionality for this example. Although this command is not set to be "undoable," I've gotten into the practice of dividing the workload between the doIt and redoIt methods. Generally, you want to use the doIt method to parse any argument list you might have and set any internal variables that will be accessed later, and then call the redoIt method to perform the actual work. This puts the code into a good position to add "undo" functionality later should you decide to take on the task via the undoIt method.

Next we define the constructor and destructor methods as well as the creator method.

```
// constructor & destructor
getCpConVerts::getCpConVerts() {}
getCpConVerts::~getCpConVerts() {}

// creator
void *getCpConVerts::creator()
{
        return new getCpConVerts;
}
```

There is no need to do anything in either the constructor or destructor methods other than to define them, and the creator method simply returns a new instance of this command that, in this case, will be destroyed immediately after execution, since we haven't defined "undo" functionality.

```
MStatus getCpConVerts::doIt(const MArgList &args)
{
        MStatus status;

        // retrieve the arguments
        conName = args.asString(0, &status);

        status = redoIt();
        return status;
}
```

Our doIt method only does the groundwork of the command; it parses the argument list and stores the resultant string into our instance variable. The *MArgList* class provides convenience routines to fetch indexed items from the list as specific types. Here I'm retrieving the only argument we expect and interpreting it as a string. I don't error check the result as the front end MEL script makes sure that a single, valid string is passed to the command. The status variable is used to error check most other method invocations along the way, printing out appropriate error messages at each step. Once we've retrieved our parameter, the name of the cloth constraint we're interested in, we call the redoIt method to perform the real work.

```
MStatus getCpConVerts::redoIt()
{
    MStatus         status;

    // put the surface on the selection list and retrieve the dep node
    MSelectionList sList;
    MObject conObj;
    MDagPath conDagPath;

    status = sList.add(conName);
    if (!status) {status.perror("adding constraint to sel list"); return
status;}

    status = sList.getDagPath(0, conDagPath, conObj);
    if (!status) {status.perror("retrieving constraint from sel list");
return status;}
```

The first part of the method illustrates a very common procedure for accessing an object in Maya from the API when the object's name, as opposed to any of its attributes, is given. Since this is a command plug-in, we don't have any plugs (the pipes for making connections between node attributes) to pole for data as we would with a node-type plug-in. Instead we have the name of the node as a string. So what we do is create an instance of a selection list via the *MSelectionList* class and use the add method of the class to put the object on the list. Wildcards can be used with this method, so you can add multiple items at once if desired. Before going on, it's worthwhile noting that another way to have written this plug-in is to have the user select the actual cloth constraint node and then call the command without passing any names. In that case, a "current" selection list would already exist and we would then simply extract the selected item from the list. I chose to write the plug-in with the parameter so that I would need to do less error checking in the code with the idea in mind that any checking and validation of the data would be done in the front end MEL script that we'll look at shortly. The Maya API documentation has some very good examples of selection methods described under the "Selection" chapter, so be sure to check it out. It shows you how to iterate over objects, select objects by name, etc.

```
        // get handle to the "indexList" array attr
        MFnDagNode nodeFn(conDagPath);
    MPlug indexListPlug = nodeFn.findPlug("indexList", &status);
        if (!status) {status.perror("retrieving indexList attr plug");
return status;}
```

Next, I retrieve the first and only element on the selection list via the getDagPath method. I use this method to get a unique path to the constraint DAG node, which I can then use to find the plug I want. To accomplish this, I create an instance of an *MFnDagNode* object, initializing it via the DAG node, path I have in the conDagPath variable. Finally, I fetch the plug of interest by name using the findPlug method of the *MPlug* class, leaving me with the indexListPlug instance.

```
    MObject indexListPlugObj;
    status = indexListPlug.getValue(indexListPlugObj);
        if (!status) {status.perror("retrieving indexList obj from
plug"); return status;}
        // get handle to the attr array
        MFnIntArrayData indexListAttrArray(indexListPlugObj);
```

To get to the actual array of integers, I need to create an instance of the *MFnIntArrayData* class using a *Mobject*, which I produce using the getValue method on the indexListPlug instance. I'm finally at the data I want! Wheh!! Remember that I knew from the beginning that I was dealing with an array of integers and by searching through the alphabetical API class listing, I found the *MFnIntArrayData* class and figured that I would eventually use it to access the real data. The rest of the code is just the path to get there.

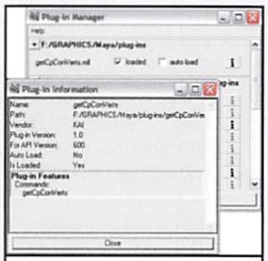
```
    long numVerts = indexListAttrArray.length();

    // return the array of int indexes to the vertices
    MPxCommand::clearResult();
    for(int i=0; i<numVerts; i++)
        MPxCommand::appendToResult(indexListAttrArray[i]);

    // return status
    return status;
}
```

At last we get to the real heart of the code, which is simply a for loop that returns all of the array data (the vertex indexes) to the user, where it can be stored in a MEL variable or simply examined in the *Script Editor* window. The mechanism to return data from a command is the *MPxCommand* class' appendToResult method. To be safe, I also call the clearResult method to make sure I'm starting off with a clean slate. As a note, if you're only going to return a single result as opposed to an array of data, you can use the setResult method. We end the redoIt method by returning the required status value as we did for the doIt method, hopefully indicating success. This method demonstrates a bit of the convoluted nature of the API, in which you have to kind of onion peel your way to the actual data you want via various classes and methods.

We complete the plug-in by defining the necessary initialize and uninitialize routines. If you don't include these functions, the plug-in will not load into Maya. The initialize function is used to register any commands, nodes, devices, etc. that your plug-in provides. Likewise, the uninitialize function de-registers anything that you registered with the initialize function. Both of these routines can be either a C or C++ function.

```
extern "C" MStatus initializePlugin(MObject obj)
{
    MStatus             status;
    MFnPlugin     plugin(obj, "KAI", "1.0", "Any");

    status = plugin.registerCommand("getCpConVerts", getCpCon-
Verts::creator);
    if (!status)
    {
        status.perror("registerCommand");
        return status;
    }

    return status;
}

extern "C" MStatus uninitializePlugin(MObject obj)
{
    MStatus             status;
    MFnPlugin     plugin(obj);

    status = plugin.deregisterCommand("getCpConVerts");
    if (!status)
    {
        status.perror("deregisterCommand");
        return status;
    }

    return status;
}
```

When initializing the plug-in, we create an instance of the *MFnPlug-in* class in which we specify attributes that show up when clicking on the info button next to the plug-in name in the *Plugin Manager* window as shown in *Image 2*. This information includes the vendor name (here my initials "KAI"), the version, and the required API version. We then proceed to register the command. When un-initializing, we simply de-register the command. Here now is the complete source code listing which is also available online at *www.hdri3d.com/resources*.



```
Name:              getCpConVerts
Path:              F:/GRAPHICS/Maya/plug-ins/getCpConVer
Vendor:            KAI
Plug-in Version:   1.0
For API Version:   600
Auto Load:         No
Is Loaded:         Yes
Plug-in Features
  Commands:
    getCpConVerts
```

IMAGE 2– MAYA'S PLUG-IN MANAGER WINDOW SHOWING THE LOADED PLUG-IN

```
// INCLUDES
#include          <maya/MFnPlugin.h>
#include          <maya/MPxCommand.h>

#include          <maya/MArgList.h>
#include          <maya/MFnDagNode.h>
#include          <maya/MDagPath.h>
#include          <maya/MPlug.h>
#include          <maya/MSelectionList.h>
#include          <maya/MFnIntArrayData.h>

// COMMAND CLASS DECLARATION
class getCpConVerts : public MPxCommand
{
    public:
                                          getCpConVerts();
        virtual                          ~getCpConVerts();

        MStatus                          doIt(const MArgList &args);
        MStatus                          redoIt();
        static     void          *creator();
    private:
        MString                         conName;
};


// MEMBER FUNCTION DEFINITIONS

// constructor & destructor
getCpConVerts::getCpConVerts() {}
getCpConVerts::~getCpConVerts() {}

// creator
void *getCpConVerts::creator()
{
    return new getCpConVerts;
}

// doIt : fetches node name and then calls redoIt
MStatus getCpConVerts::doIt(const MArgList &args)
{
    MStatus status;

    // retrieve the arguments
    conName = args.asString(0, &status);

    status = redoIt();
    return status;
}

// redoIt : returns the list of constrained vertices
MStatus getCpConVerts::redoIt()
{
    MStatus         status;

    // put the object on the selection list and retrieve the dag path
    MSelectionList sList;
    MObject conObj;
    MDagPath conDagPath;

    status = sList.add(conName);
    if (!status) {status.perror("adding constraint to sel list"); return
status;}

    status = sList.getDagPath(0, conDagPath, conObj);
    if (!status) {status.perror("retrieving constraint from sel list");
return status;}

    // get handle to the "indexList" array attr
    MFnDagNode nodeFn(conDagPath);
MPlug indexListPlug = nodeFn.findPlug("indexList", &status);
    if (!status) {status.perror("retrieving indexList attr plug"); return
status;}

MObject indexListPlugObj;
status = indexListPlug.getValue(indexListPlugObj);
    if (!status) {status.perror("retrieving indexList obj from plug");
return status;}

    // get handle to the attr array
    MFnIntArrayData indexListAttrArray(indexListPlugObj);

    long numVerts = indexListAttrArray.length();

    // return the array of int indexes to the vertices
    MPxCommand::clearResult();
    for(int i=0; i<numVerts; i++)
        MPxCommand::appendToResult(indexListAttrArray[i]);

    // return status
    return status;
}

// initialize / register
extern "C" MStatus initializePlugin(MObject obj)
{
    MStatus                 status;
    MFnPlugin       plugin(obj, "KAI", "1.0", "Any");

    status = plugin.registerCommand("getCpConVerts", getCpCon-
Verts::creator);
    if (!status)
    {
        status.perror("registerCommand");
        return status;
    }

    return status;
}

// uninitialize / deregister
extern "C" MStatus uninitializePlugin(MObject obj)
{
    MStatus                 status;
    MFnPlugin       plugin(obj);

    status = plugin.deregisterCommand("getCpConVerts");
    if (!status)
    {
        status.perror("deregisterCommand");
        return status;
    }

    return status;
}
```

I mentioned earlier that I would use a front end MEL script to run the command in order to leave the plug-in as lean and focused on its task as possible. It is defined as a global proc so that it can be invoked from anywhere within Maya and performs a number of tasks in the following order:

–checks to make sure that something is selected
–checks whether the selected node is a cloth constraint and complains if it isn't
–locates the cloth constraint's associated cloth mesh and complains if not found
–loads the plug-in if it isn't already loaded
–selects the mesh vertices returned by the plug-in

Here is the MEL code which is also available online at *www.hdri3d.com/resources*.

```
global proc selectCpConVerts()
{
    // return if no objects selected
    string $selObjs[] = `ls -sl`;
    if ((size($selObjs) == 0))
    {
        print("No cloth constraint selected!\n");
        return;
    }

    // find the actual cloth constraint node
    string $clothConNode = "NONE";
    string $nodeType = `nodeType $selObjs[0]`;
    if ($nodeType == "transform")
    {
            string $childNodes[] = `listRelatives -c $selObjs[0]`;
            string $childNode;
            for ($childNode in $childNodes)
            {
                string $childNodeType = `nodeType $childNode`;
                if (gmatch($childNodeType, "cp*Constraint"))
                    $clothConNode = $childNode;
            }

            if ($clothConNode == "NONE")
            {
                print("No cloth constraint selected!\n");
                return;
            }
        }
        else if (gmatch($nodeType, "cp*Constraint"))
            $clothConNode = $selObjs[0];
        else
        {
            print("No cloth constraint selected!\n");
            return;
        }

        // find the mesh cloth object whose vertices we will select
        string $parent[] = `listRelatives -p $clothConNode`;
        $parent = `listRelatives -p $parent[0]`;

        string $clothMeshNode = "NONE";
        string $kids[] = `listRelatives -s -c $parent[0]`;
        string $kid;
        for ($kid in $kids)
            if (`nodeType $kid` == "mesh")
                $clothMeshNode = $kid;

        if ($clothMeshNode == "NONE")
        {
            print("Can't find corresponding cloth object!\n");
            return;
        }


        // load the plugin if it's not already installed
        if (!(`pluginInfo -q -l "getCpConVerts"`))
        {
            print("Loading the getCpConVerts plugin...\n");
            loadPlugin getCpConVerts;
        }

        // select the vertices at last!
        int $indexes[] = getCpConVerts($clothConNode);

        select -cl;
        int $i, $index;
        for($i=0; $i<size($indexes); $i++)
        {
            $index = $indexes[$i];
            select -add ($clothMeshNode + ".vtx[" + $index + "]");
        }
    }
```

Hopefully, you'll find the MEL code fairly straightforward. It's really just making use of a few commonly used commands such as ls, listRelatives, gmatch, nodeType, and select to validate the user's selection so that the plug-in doesn't have to. I create such MEL wrapper scripts for most of the plug-ins I write, especially for node-type plug-ins, to automate the creation and connection of the node in the *dependency graph*. This is a simple API example, which has hopefully provided some insight into how data is organized internally in Maya and how to figure out how to retrieve it using the available classes. It also demonstrates the utility of using front end MEL scripts to wrap the plug-in both for developer and user convenience. Again, the online documentation is fairly robust, including a very helpful FAQ section that deserves a good perusing. All you need now is a compiler and you're on your way! Once compiled, you can test the plug-in by creating a cloth object, assigning any cloth constraint to it (i.e. a mesh constraint), selecting the constraint as in *Image 3*, and then running the *selectCpConVerts* wrapper script. You should now see that in place of the constraint, the vertices contained in it are now selected as in *Image 4*. You'll need to have the compiled plug-in in some path defined in the MAYA_PLUG_IN_PATH environment variable, or you'll need to load it manually using the *Plug-in Manager* window accessed from the *Window>Settings/Preferences>Plug-in Manager* menu. Happy constraining!

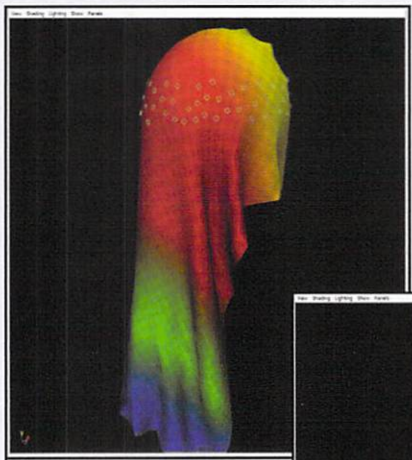


IMAGE 3– THE CLOTH CONSTRAINT SELECTED

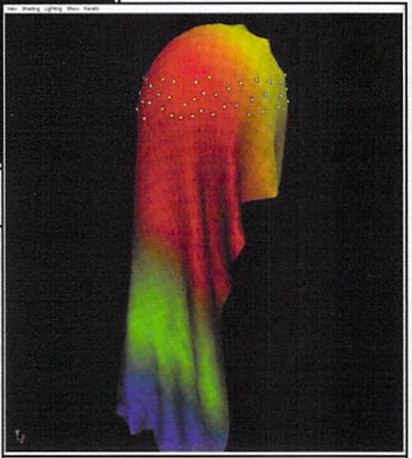


IMAGE 4– THE CONSTRAINT'S VERTICES SELECTED BY THE PLUG-IN

KENNETH IBRAHIM'S CAREER IN CG STARTED IN JAPAN IN THE MID 90'S WHILE WORKING AT SEGA CREATING GRAPHICS FOR TV COMMERCIALS AND THEME PARKS AS WELL AS PROGRAMMING SOFTWARE TOOLS FOR LARGE ARCADE GAME SYSTEMS AND LOCATION-BASED ENTERTAINMENT RIDES. HIS VOICE ALSO APPEARS IN A FEW GAMES, NOTABLY *SEGA RALLY CHAMPIONSHIP*. HE THEN JOINED THE ALIAS/WAVEFRONT TOKYO OFFICE WHERE HE DEMO'ED, TRAINED, AND SUPPORTED MAYA SINCE ITS ALPHA PHASE. SINCE RETURNING TO THE US IN '97, KEN HAS WORKED IN FX AND LIGHTING ON A NUMBER OF FEATURE FILMS INCLUDING *SHREK, FINAL FANTASY, MATRIX REVOLUTIONS, X2, PETER PAN,* AND *I, ROBOT.*

BY HUNTER WOLF

FAKING MULTIPLE CAMERAS IN LIGHTWAVE

HUNTER WOLF EXPLAINS HOW TO CREATE THE ILLUSION OF MULTIPLE CAMERAS WITH SINGLE CAMERA RENDERING.

One of my favorite things about using computers is that you can sometimes find homemade workarounds to limitations in the software. Take Photoshop, for example; there are a million and one ways to make a cloud. Even if a feature you need is included in your favorite software application, you can often times find faster ways to perform a task to suit your needs.

I really enjoy making films in 3D, but one thing I wish LightWave had is the ability to render an entire scene from as many cameras as I have set up. Imagine a scene with two characters speaking, and you want the camera to be on the first character, then switch to the other during the render process. LightWave allows you to render just one camera at a time. Then you have to save the scene as a new version, move your camera, and render the new shot.

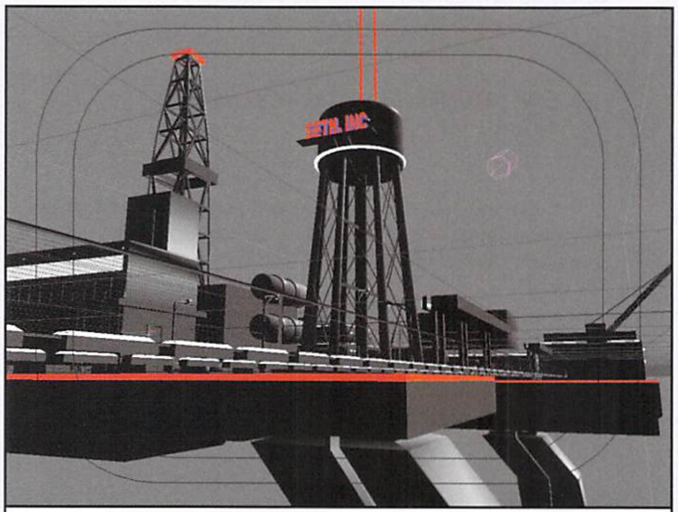
Of course, in reality, it is best to render very short scenes this way, and then put them in chronological order in a non-linear film editing application, such as Final Cut Pro or Premiere Pro (you can use any film editing software to achieve this, even iMovie and Windows Movie Maker).

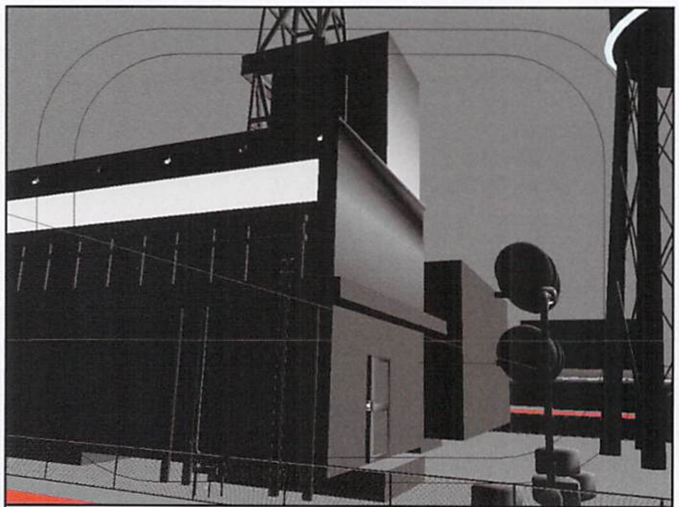
Fortunately, the Graph Editor offers a solution to the camera limitation, allowing you to change how the camera's curves are interpolated, so that you can move the camera, providing the illusion of rendering from multiple cameras.

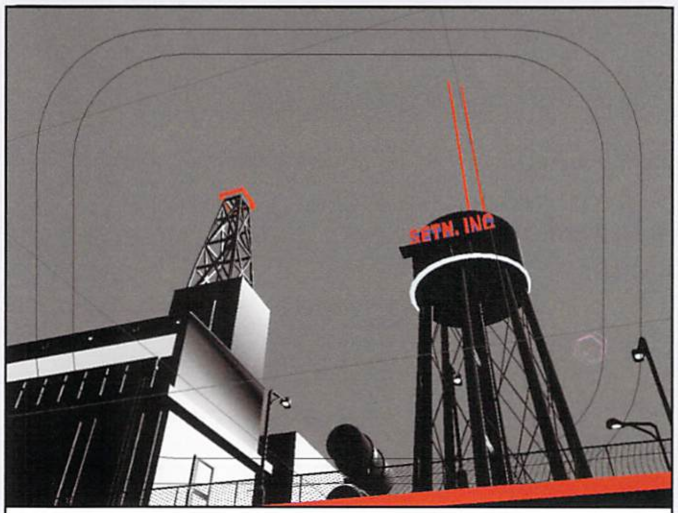
We'll use the scene at the right for our example. It's an offshore oil platform I built in Modeler for a film I'm working on. The character is having flashbacks, remembering that he was at this oil platfrom sometime in his past. For the scene, I need the camera to quickly move from one angle to the next, but without a lot of camera movement. Each shot will take one second (30 frames) of time to complete, followed by an immediate shot of the third flashback angle, which occurs over a one-frame duration.

In the first shot, on Frame 0, Top Right, I have the camera set up for an establishing shot, to give the audience a sense of where they are in the movie.

On Frame 30, Bottom Right, I move the camera to a close-up shot of the platform manager's office.


FRAME 0– OUR ESTABLISHING SHOT


FRAME 30– A TIGHT SHOT OF THE PLATFORM MANAGER'S OFFICE



FRAME 61– "WORM'S EYE VIEW" SHOT

Finally, on Frame 61, Top Left, I move the camera to a low angle (also called a "Worm's Eye" or "Ant's Eye" view).

If you were to play this file, using the Preview controls, you'd notice that the camera is anticipating motion, thus swinging too much.

This is because, by default, the incoming curves, which control the motion of the camera, are based on Bezier splines. Sometimes these splines are too organic, providing motion that is too fluid and almost floaty. Selecting the camera in the scene, then opening up the Graph Editor, displays this motion in effect. (See Figure 1, center Left).

For the flashback sequence, I want abrupt changes in the camera angles, like you'd see if watching a slide show. So, to do this, we need to change these splines to flat, or linear, curves. The Graph Edior scares a few people because it looks so unwieldy, with all the curves, grids and stuff. Trust me; once you get to know it, the Graph Editor will be your best buddy.

Shift-select all of the camera's channels in the Curve Bin. Next, right-click and drag a selection around all of the keys in the Curve Window. Now that they are all highlighted, we can change their incoming curve type in one fell swoop.

Change the incoming curve, down in the Curve Control panel, from Bezier Spline to Linear. You will notice that the curves have become straight lines in the Curve Window. (See Figure 2, bottom Left).

When you make a new preview of the scene, the camera will move from one angle to the next in quick cuts, without any rotation of the camera. You can use this technique whenever you want to produce animations that look like they were cut in a non-linear editing application.

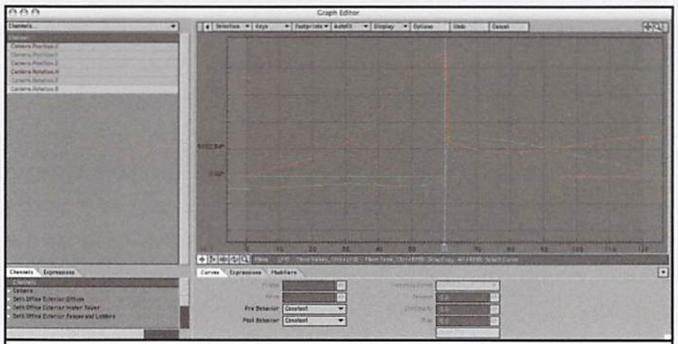


FIGURE 1– THE GRAPH EDITOR

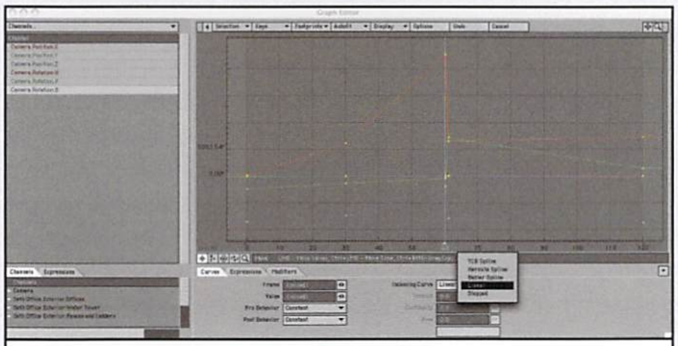


FIGURE 2– NOTE THAT THE CURVES HAVE BECOME STRAIGHT LINES

HUNTER WOLF (AKA: DWAYNE J. FERGUSON) IS A TELEVISION ANIMATION ART DIRECTOR/ SCRIPTWRITER (MUTANT LEAGUE), AUTHOR (KID CARAMEL: PRIVATE INVESTIGATOR, AND THE SUCCESSFUL HAMSTER VICE COMIC BOOK SERIES). HE IS JUMPING UP AND DOWN AT THE RELEASE OF HIS SHORT FILM *BLACK ZERO: MERCENARY ANT* TO DVD. HIS WEBSITE IS **DIEHARDSTUDIO.COM**

BY ERIC KELLER

Quick and Dirty Cloth

When attacking the problem of creating cloth effects for characters using Alias' Maya, the most obvious solutions seem to be a rig involving either Maya's Cloth module or an implementation of soft-body dynamics and springs. However, an experienced computer animator knows that there are often a great many solutions to almost any problem and that the most sensible solution is the one that gets the job done appropriately. So, really, when attacking the problem of creating cloth, or any problem for that matter, the first real task is to get a handle on what the problem actually is. If the problem is to create convincing cloth for a low polygon character that animates quickly and easily, chances are the Cloth module or soft-body dynamics will be overkill. This is especially true if we want a rig that's easy to work with and that doesn't have to be super realistic – just realistic enough to add a little extra life to our character's motion.

This tutorial will demonstrate one way to add just a little cloth motion to a character's sleeve. Truly dynamic simulators are super-cool (there's no denying that), but then you often have to deal with caching data and a rig that can be less responsive. The technique I describe in this tutorial is designed to be fast and easy, even as it loses a little realism. It gets the job done. Remember that it's only one of many possible solutions; I'm hoping it will inspire you to envision your own quick and dirty workarounds. This rig uses influence objects and a handy little mel script called lagLocator.mel. It is pretty straightforward, quick to set up, and very easy to adjust while animating.

This tutorial assumes you are familiar with setting up skeletons, binding geometry to the skeletons using smooth-bind, installing and using mel scripts, and the basics of influence objects. We all know that there are many ways to rig a character, and your preferred method may be different from mine, but that's no reason why you couldn't apply this technique to part of your favorite rig.

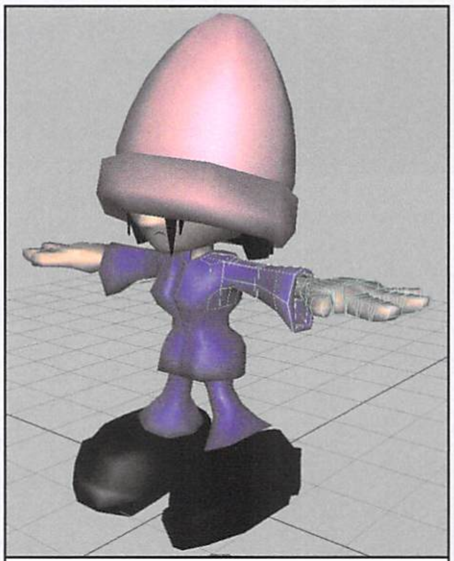


FIGURE 1– CHARACTER WITH ARM GEOMETRY SEPARATED

I'll be using one of my low polygon game-style characters as an example. This clothing rig works particularly well with low poly characters, but the same principles could be applied to more complex geometry. My character is a sort of manga styled girl, she's a bit on the tomboy side; she enjoys break dancing, snowboarding, combat with laser swords, and fuzzy pink critters that shoot lightning out of their ... but I digress. Her outfit consists of a floppy blue jump suit with a hood, big shoes, and a giant pink hat. I'm focusing on her arm/sleeve geometry, which I have duplicated and separated from the rest of her body (highlighted in figure 1). First, I'll briefly describe how I set up her arm.

BACKGROUND ON THE ARM SET UP

The arm geometry is comprised of the sleeve and the hand modeled as two separate objects (wireframe in figure 2). The

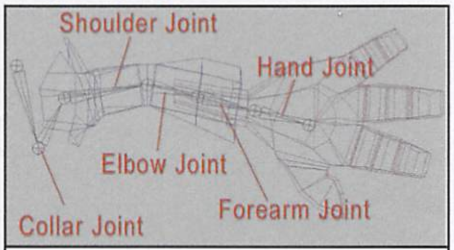


FIGURE 2– JOINT SETUP FOR SLEEVE AND HAND GEOMETRY

hand geometry stops just above the wrist. The skeleton starts where the collarbone would be connected to a spine joint, and then there is the shoulder, the elbow, the forearm, and finally the hand. I've left the finger joints off the hand for the sake of simplicity. This skeleton is a fairly typical arrangement and one of many possible arm setups. The forearm is broken into two joints to properly recreate wrist rotation on the X-axis. X rotation of the wrist actually takes place in the forearm, rather than at the wrist, where the two bones (radius and ulna) rotate around each other. As far as this tutorial goes, that's not an essential part of the rig.

The geometry is smooth bound to the skeleton. I've set up all of the weighting for the vertices in a very rudimentary way by using the Component Editor. I generally enter weights for all of the bound geometry and then add influence objects where necessary. You can also use the paint weight tools if you want a more organic and intuitive tool for joint weighting. It's very simple for this example: the vertices at the end of the sleeve have a weight value of 1 set to the forearm joint. The next row up (moving in the direction from the end of the sleeve towards the shoulder) is set at 0.8 to the forearm, and 0.2 to the elbow. The next row is 0.8 to the elbow and 0.2 to the shoulder. The row at the very center of the elbow is split 0.5 to the elbow and 0.5 to the shoulder. The row above the shoulder is set 0.8 to the shoulder and

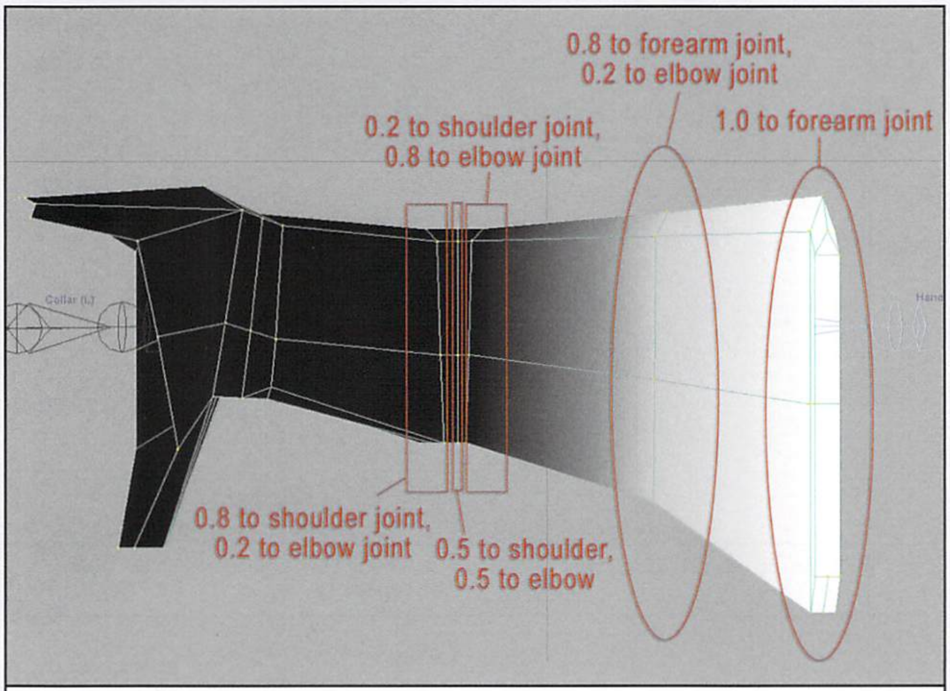


FIGURE 3– VALUES FOR SMOOTH BOUND SKIN WHICH I ENTERED MANUALLY INTO THE COMPONENT EDITOR

0.2 to the elbow. The vertices around the shoulder are set up in a similar fashion to the elbow weights I've described, but with the weights being shared by the shoulder and collar joints (see figure 3).

SETTING UP INFLUENCE OBJECTS FOR THE ELBOW AND SHOULDER, A QUICK REVIEW

My joint weighting tends to be very simplistic since I usually go back and fine tune with influence objects anyways. If you've never used an influence object, I recommend checking out a more in-depth tutorial, but I'll briefly describe the way I do it so you can follow along later when I describe how I use them for cloth effects. I would recommend the Gnomon Workshop DVDs on this subject for further study. Most often, influence objects are used to correct for joint compression problems. They are somewhat tedious to rig, but the payoff is big if you want your geometry to deform correctly when animating the joints.

Here are the basic steps I use for adding an influence object to the inside of the elbow:

1. I create a piece of polygon geometry. The shape isn't really important, so a plane will do nicely. To make it more obvious what part of the geometry is being affected by the influence object, I use the Create Polygon tool and point snap so that the vertices match the vertices of the arm geometry. I also shade it bright red so it stands out from the character's geometry.

2. I delete history on the new piece of geometry.

3. I center the pivot point of the geometry.

4. I rename it something that will stick out in the Component Editor; in this case I use all caps and named it "L_ELBOW_INSIDE_INF_OBJ."

5. I parent this piece of geometry to the elbow joint.

6. I select the arm geometry, and then the influence object, and then go to (from the Animation Menu mode) *Skin>Edit Smooth Skin>Add Influence* and choose the option box. I use these settings:

— Use Geometry should be unchecked
— Drop-off can be at 0.1 (doesn't matter)
— Lock Weights should be checked
— Default Weight should be at zero

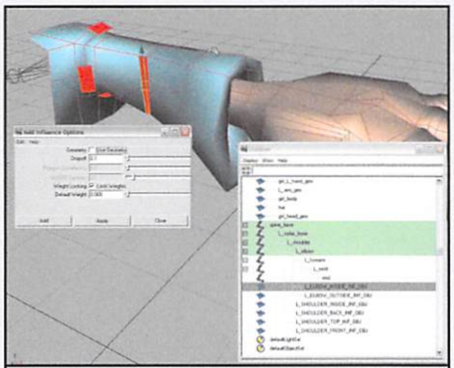


FIGURE 4– INFLUENCE OBJECTS ADDED TO THE ELBOW IN A FAIRLY TYPICAL SETUP

As I mentioned before, this really isn't a tutorial devoted to influence objects, so I'm not going to explain these options. For the most part, the default weight is set to zero so I can set the weights manually in the Component Editor.

7. Now, when I select the L_ELBOW_INSIDE_INFOBJ, the arm geometry wireframe turns purple, indicating that my influence object has an output connection to the arm geometry (see figure 4)

8. When I select the vertices of the arm geometry and open the Component Editor, I check to make sure L_ELBOW_INSIDE_INFOBJ is listed and that it has weight values of zero (note: sometimes columns with zeros in them are hidden by default in the Component Editor, so check the Options menu in the Component Editor if things don't look right. If the Component Editor isn't updating properly, you can try toggling the Hide Zero Columns option on and off to get the Editor to behave.).

9. Now I manually enter the weight values into the Component Editor, just as I did for the joints. I create a drop-off by sharing the weight values with the skeleton joints and other influence objects when appropriate. Generally for the vertices on the inside of the elbow, the middle row is set to 1.0 for the influence object. The rows on either side are spilt with 0.5 going to the influence object and 0.5 going to the closest joint. After I set these weights, I pull the influence object away from the arm geometry and make sure it's deforming these vertices correctly (see figure 5 *Next Page*).

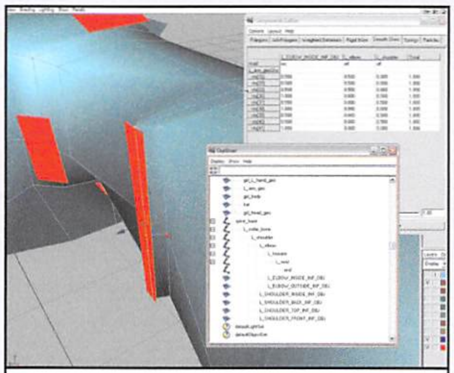


FIGURE 5– THE INFLUENCE OBJECT IS PULLED AWAY FROM THE ARM GEOMETRY JUST TO MAKE SURE IT'S WORKING CORRECTLY

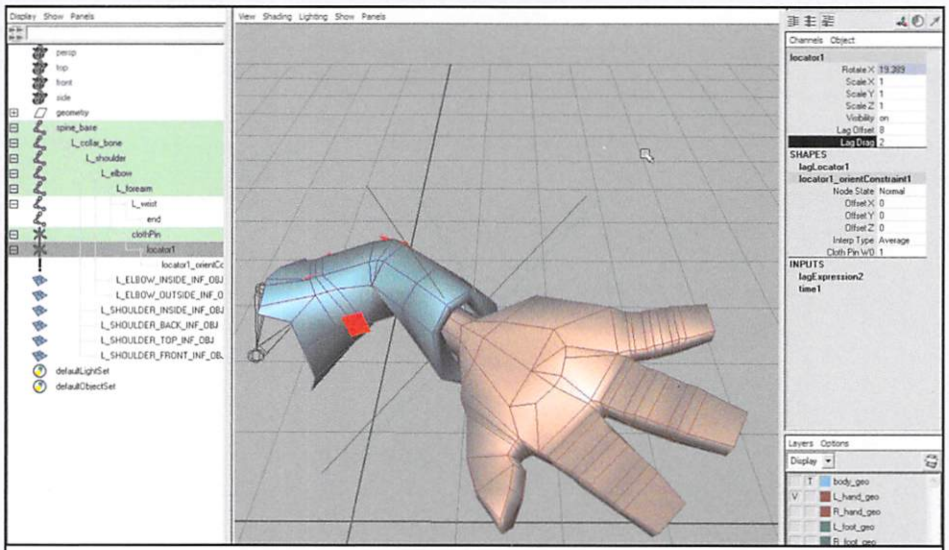


FIGURE 6– THE LAG LOCATOR IS SHOWN IN GREEN, LAGGING BEHIND THE CLOTHPIN LOCATOR PARENTED TO THE ELBOW

10. The last step is to use driven keys to translate, rotate, and scale the influence objects in such a way as to properly deform the inside of the elbow as the elbow joint rotates. I use the elbow joint as the driver and the influence object as the driven. I set up additional influence objects on the outside of the elbow and around the shoulder. Once a piece of geometry is turned into an influence object, its output options are automatically set so that it will not render.

SETTING UP AND USING LAGLOCATOR.MEL

The lagLocator.mel script is one of those little gems you find while digging through the hundreds of free mel scripts on highend3d.com.

(http://www.highend3d.com/files/dl.3d?group=melscripts&file_loc=lagLocator-v1.0-.mel&file_id=2160).

The script was written by Pete Shinners and can be found under the animation section in the highend3d.com mel scripts animation directory. The script is simple to use and has potential for hundreds of applications. It's not as glamorous as Maya's Cloth Module or soft-body dynamics, but it is a real time saver. Essentially, it creates a locator that is parented to whatever object you have selected when you run the script. When you animate the parent, this locator will lag behind it in time and space. It has two custom attributes that you can control in the Channel box. These are Lag Offset and Lag Drag.

Lag Offset controls the number of frames the locator will drag behind. In other words, it's a temporal lag. Lag Drag will create the fake dynamics we're looking for by lagging in distance (it's a spatial lag). As you would expect, a higher Lag Drag value will create a larger drag distance for a more exaggerated effect. To use the script, put it into your scripts directory, source the script, select an object and type "lagLocator" in the command line. That's all there is to it. Select the Lag Locator (which will be parented to your object and most likely named locator1), and in the Channel box, you'll find the Lag Offset and Lag Drag fields. Try creating an empty scene and use the script on an object just to get the hang of it.

ADDING THE LAGGING LOCATOR TO THE SKELETON

OK, now on to the point of this tutorial: faking cloth movement. The rig I'm using is pretty simple:

1. I created a locator.
2. I named it "clothPin."
3. I parented it to the forearm joint.
4. I entered zeros in the Channel box for the Translate and Rotate values so that the locator would move to the joint's pivot point.
5. I move the clothPin locator down towards the wrist.
6. With "clothPin" selected, I ran the lagLocator script.
7. The Lag Locator script only accounts for translation, not rotation, so to fix this I created an orient constraint between locator1 and the clothPin. This means that you'll have a lag in the clothes when the arm moves, but no lag when the forearm twists. This can always be fixed with some well-placed keyframes when it comes time to animate. I haven't tried writing a script that would take rotation into account, but I'm guessing it could be done fairly easily.

When you replicate this rig, try setting some keyframes on the arm and move it around. The locator won't update as you set the keys, but it will when you play it and when you scrub through the timeline (and you don't have to cache anything!). Figure 6 should give you the gist of the rig at this point.

CREATING THE CLOTHING INFLUENCE OBJECTS

The final part of the setup is to create some influence objects that will deform the end of the sleeve. I created eight polygon objects that match the geometry of the sleeve's end using the create polygon tool with Point Snapping on. To add them as influence objects, I:

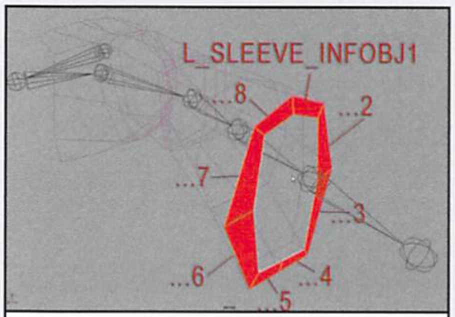


FIGURE 7– THE EIGHT INFLUENCE OBJECTS ARE CONTROLLING THE END OF THE SLEEVE

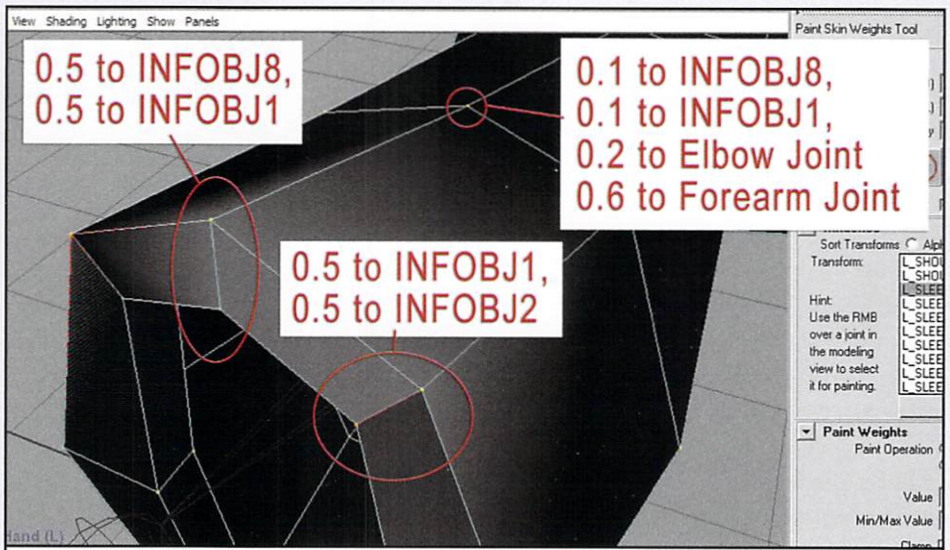


FIGURE 8– A DROP-OFF HAS BEEN CREATED BY SHARING WEIGHT VALUES BETWEEN INFLUENCE OBJECTS AND THE FOREARM AND ELBOW JOINTS

1. Created each polygon.
2. Deleted history on these polygons.
3. Centered their pivots.
4. Renamed each on in a clockwise manner starting from the top. The new names are: "L_SLEEVE_INFOBJ1, L_SLEEVE_INFOBJ2, L_SLEEVE_INFONJ3 ... LSLEEVE_INFOBJ8. See figure 7
5. I added these objects as influence objects to the sleeve geometry with the same settings described in the previous section.
6. I grouped them together.
7. I renamed the group "L_SLEEVE_INFOBJ_group" and centered its pivot.

The next step is to weight the vertices of the sleeve to the influence objects. To do this, I selected the three vertices that comprise the edge of the polygons at the end of the sleeve as shown in figure 8. I went into the Component Editor and created the drop-off effect by setting the weights of the vertices. The vertices on the inside of the sleeve and all the vertices around the edge of the sleeves are set to 1.0 for the influence objects. More specifically, they are split evenly between the adjacent influence objects (0.5 and 0.5 for each). The row of vertices further up the sleeve share their weights between the forearm joint and the influence objects. I entered a value of 0.1 and 0.1 split between the two influence objects and then 0.2 to the elbow and 0.6 to the forearm. Once all this setup is finished, the fun part begins.

ANIMATING THE ARM

The final step is to parent the "L_SLEEVE_INFOBJ_group" to the clothPin locator. Once I start putting keyframes on the joint rotations (or using inverse kinematics), the cloth motion starts to become obvious, and it does look pretty cool. During the process of setting up the keyframes, things may look a little messed up, but once the animation is rewound and played, things should start to look floppy (in a good way). As I mentioned before, scrubbing through the animation also shows the floppiness in action and no caching of data is needed.

The cloth motion can easily be tweaked by adjusting the Lag Offset and the Lag Drag channels for locator1. Also, since the influence objects that comprise the edge of the sleeve are grouped and parented to the clothPin locator, keyframes can be set on the group, or on the individual influence objects, in order to correct for any weird intersections, to simulate stretching, or to compensate for the fact that there's not sufficient drag when the forearm rotates. At this point, it's possible to add a little realism using good old keyframes, or even by using driven keys and some artistic sensibility to make up for what we lost by not using a dynamic simulation like Maya Cloth.

OTHER RIGS

The ideas in this tutorial are a good foundation for experimentation. The combination of influence objects and the lag locator script could be applied to other parts of the clothing or even to add just a little jiggle to a pot belly or some fleshy jowls. By point constraining the end of an IK chain to a lagging locator, you could add a quick bit of floppiness to pony tails, stray tentacles, or sections of a skirt.

For even more tools to simulate cloth, check out a couple handy scripts in the Maya 6 Bonus tool pack available on the Alias Web site. There's a script for adding multiple influence objects at the same time and a script for adding secondary motion to joint chains. Both of these could come in handy combined with the ideas in this tutorial.

I hope this tutorial has inspired you. Please feel free to contact me if you have any questions.

ERIC KELLER HAS SPENT THE LAST SIX YEARS AS A SCIENTIFIC ANIMATOR AT THE HOWARD HUGHES MEDICAL INSTITUTE IN CHEVY CHASE, MARYLAND. WHEN HE'S NOT RENDERING MOLECULAR DATA AT WORK HE ENJOYS CREATING CHARACTER ANIMATION AT HOME IN WASHINGTON DC. HIS TRAINING IS IN MUSIC AND ART. HIS PRIMARY TOOLS ARE MAYA, ZBRUSH, PHOTOSHOP, AND AFTER EFFECTS, INSPIRATION COMES FROM HIS WIFE ZOE, THEIR TWO DOGS AND WAY TOO MUCH TV.

BY DEUCE BENNETT

EXPLOSIONS WITH HYPERVOXELS

Hello there everyone – well, a new magazine, a new article – and I feel that an introduction is in order. Everyone calls me Deuce. It is a nickname that I picked up in the film industry while I was working with my father doing physical special effects. I have always felt that having the background in actual film production has helped me enormously doing CGI – and I would suggest to anyone who is in school studying CGI, at least take a photography class or find some other way to get behind a real camera. You learn so much so fast there that directly translates to the computer.

Now, one of my past passions was destroying things – yes, I blew up buildings, cars, people – I WAS the life of the party. But liabilities, insurance, and the supreme knowledge that I had in my grasp the power to bring an unhappy end to many people's life stories made me feel all the more comfortable to do these same things now in the computer. After all, the only way I can hurt someone with my computer is to throw it at someone and be lucky enough that they don't duck!

Recently, I have been doing many animated visualizations for Future Combat Systems for military contractors and the military directly. Now, talk about a reason for learning ways of blowing up CGI vehicles and such! I have recently hit upon my favorite sprite settings for Hypervoxels for doing a fireball, and I would like to show that to you now, with some explanations along the way.

Ready? Let's get to it!

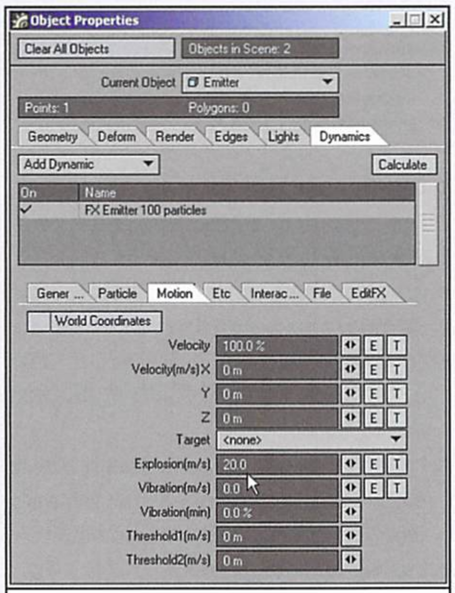
OK, first, let's think a bit about explosions – primarily fireballs. OK – so explosions are primarily a fast affair, with hot gases rising, burning from an orange-red to a black smoke afterwards. Scale is key, as something REALLY BIG moves slooooooow. So a fast moving fireball will be a small pop, like a gas can in a car, to a HUGE fireball representing a spaceship or a large airplane. The settings I'm going to be discussing here should work for "average" fireballs – if you need smaller, or larger, I hope to explain things in such a way that you can make your adjustments as you need.

I'm going to take some liberties here and make some assumptions. I assume that you will know how to add particles, and that you know what the differences in partigons, particle emitters, and Hypervoxels emitters are. If you don't, there are many beginner tutorials and lots of information out there to get you up to speed.

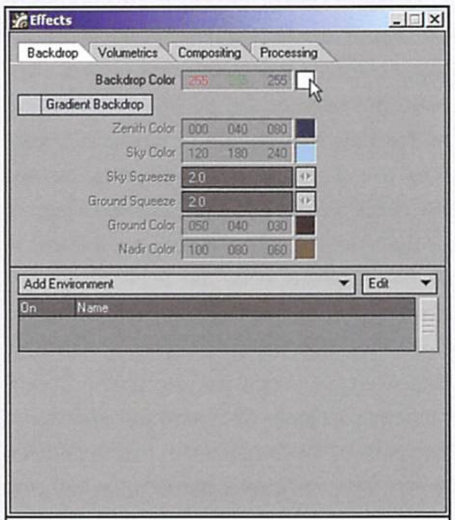
Oh, and one more thing. I want to tell you right away, here and now, that working with Hypervoxels is more mysticism and voodoo than, say, modeling a car. When you are working with Hypervoxels, trying to do something like clouds, or fireballs, or anything as ephemeral as these – there are no hard and fast rules. There are only guidelines, and it's up to you as the artist to make decisions on "what looks good to you." What I hope to do is to explore more the "why" than "how" – that way, you're more ready to do your own explorations into making things look the way you want them to.

OK. What would an explosion be without some particles? Particles, specifically Hypervoxel emitters, are what all my fireballs are made from.

Let's add a Hypervoxels emitter, make it real small, say, 50mm each dimension – and since a fireball is a fast moving affair, let's set our emitter to 100 particles per FRAME, and only 100 total particles. Only other thing to set would be to make it an explosion, of about 20m, in the Motion Tab. Now, do two things – first, hit "Calculate" – you will quickly see the particles moving in their explosion – I do this, because initially, the particles will not "come from the center" – they seem to be born "outside" the point of emission; this just catches the calculations up. Second, set your background to white – this way, you should be able to see all of our work in VIPER.


Object Properties
Objects in Scene: 2
Current Object: Emitter
Points: 1 Polygons: 0
Dynamics tab
On ✓ Name: FX Emitter 100 particles
Motion tab
Velocity: 100.0 %
Velocity(m/s) X: 0 m
Y: 0 m
Z: 0 m
Target: <none>
Explosion(m/s): 20.0
Vibration(m/s): 0.0
Vibration(min): 0.0 %
Threshold1(m/s): 0 m
Threshold2(m/s): 0 m

20M EXPLOSION SPEED

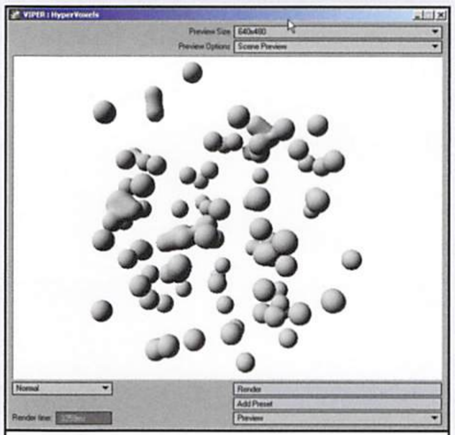

Effects
Backdrop tab
Backdrop Color: 255 255 255
Zenith Color: 000 040 080
Sky Color: 120 180 240
Sky Squeeze: 2.0
Ground Squeeze: 2.0
Ground Color: 050 040 030
Nadir Color: 100 080 060

SET BACKDROP COLOR TO WHITE SO WE CAN SEE WHAT WE'RE DOING

Now the real work begins. Open Hypervoxels and activate our emitter.

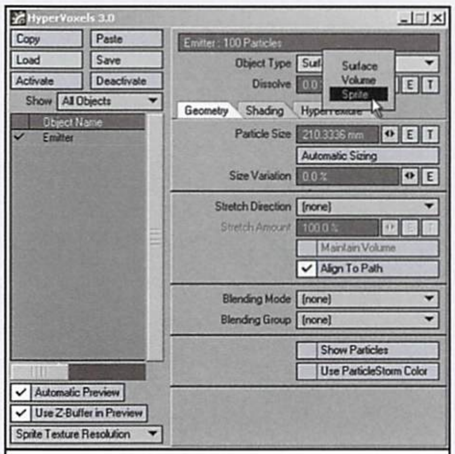
I like to think of each particle with the voxels applied as a single kernel of popcorn. This is especially true in working with explosions. If you can remember back, there used to be a method for creating an explosion look by morphing spheres with textures on them. Each sphere would grow throughout the explosion and dissolve out. It didn't look *that* good then, but the concept is still kind of the same in Hypervoxels.

Now let's activate VIPER. Depending on what frame you activate your VIPER, your screen might not look like mine (I was on frame 3 when I made that screenshot).



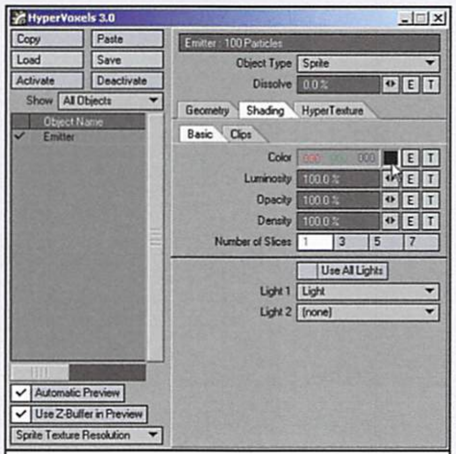
BLOBS! NOT WHAT WE'RE AFTER

First things first. We are going to be using Sprite Mode, so turn that from Surface to Sprite Hypervoxels.



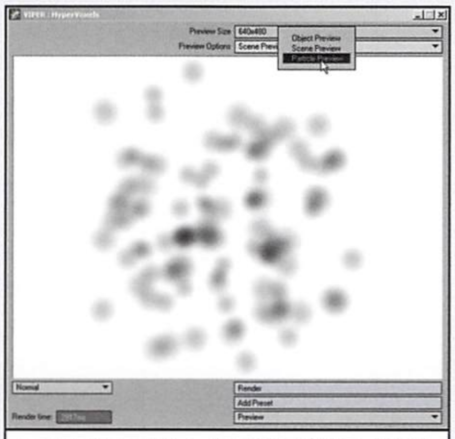
SPRITE MODE – THAT'S FOR US

You'll practically see your Voxels disappear, as the default color is a light grey. Change that to a full black; it will make it easier to see our work.



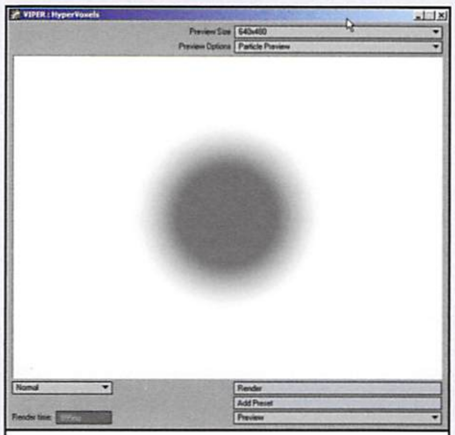
MAKE 'EM BLACK – SHOWS UP AGAINST WHITE

Let's work on our texture. For this, I would turn VIPER to Particle Preview, as it's easier to see the settings that we make on a single voxel.

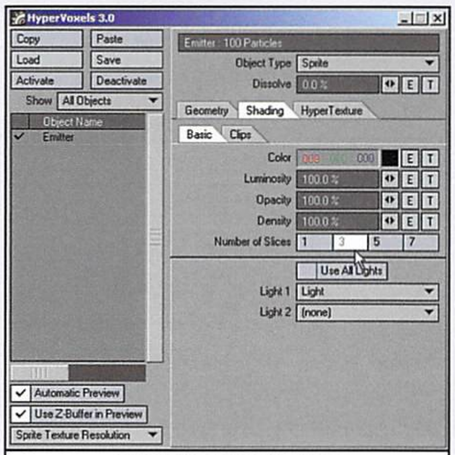


PARTICLE PREVIEW – UP CLOSE WITH A SINGLE PARTICLE

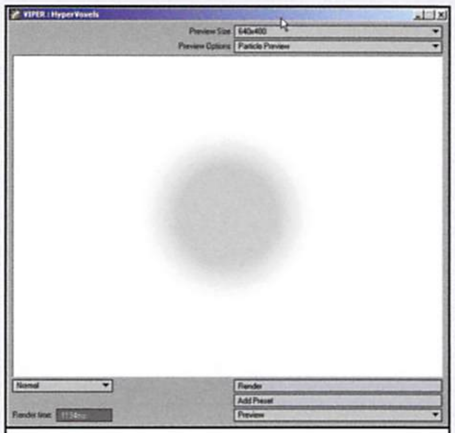
Now, a word on Sprite Mode. What it really is, or rather what you should think of it as, are slices of a volumetric hypervoxel. Imagine that you have a cotton ball; if you slice a thin piece out of the center, then you have "one slice" – but if you slice one on each side of that, you have three slices – and so forth up to a maximum of seven. I use a three slice because it gives a little more density, but not so overpowering for my tastes. When you do this, you'll see the VIPER adjust, and they are a little softer now.



ONE SLICE SPRITE

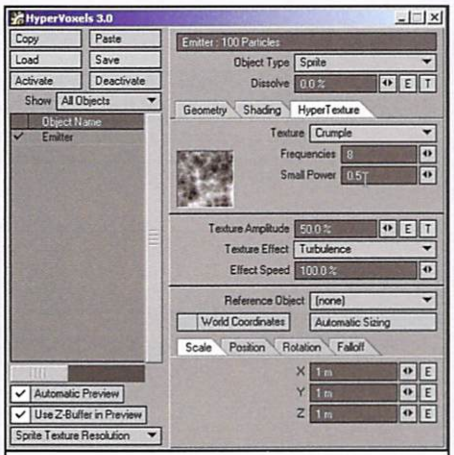


CHANGE TO THREE SLICE



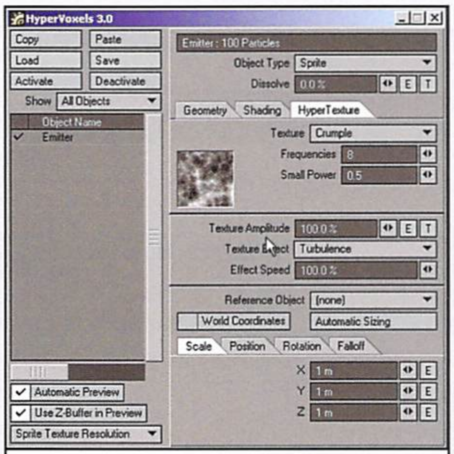
SAME PARTICLE – WITH THREE SLICES

Click on the HyperTexture Tab, and make that Crumple Texture active. Because of the detail that I want to see in the fireball, I turned up the Frequencies to 8, but I didn't want a ton of small details, so I lowered the Small Power to around .5 – a way to make it your own, or to make different fireballs unique, is to change these settings.


HyperVoxels 3.0 — Emitter : 100 Particles; Object Type: Sprite; Dissolve: 0.0 %; HyperTexture: Texture: Crumple; Frequencies: 8; Small Power: 0.5; Texture Amplitude: 50.0 %; Texture Effect: Turbulence; Effect Speed: 100.0 %; Reference Object: (none); Scale X: 1 m, Y: 1 m, Z: 1 m; ✓ Automatic Preview; ✓ Use Z-Buffer in Preview

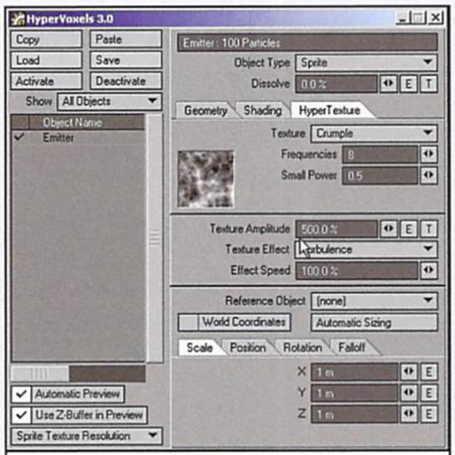
MORE FREQUENT CRUMPLE

Since a fireball is a moving, organic thing, we need to make our texture move, so let's do that with the Texture Amplitude and Texture Effect area. The Texture Amplitude modifies the Crumple Texture that we have applied. Change it from 50% to 100% and look at VIPER.

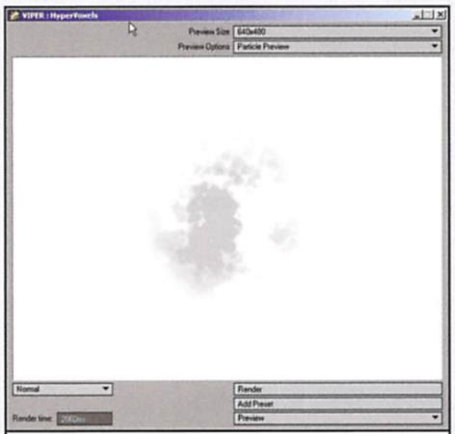

HyperVoxels 3.0 — Emitter : 100 Particles; Object Type: Sprite; Dissolve: 0.0 %; HyperTexture: Texture: Crumple; Frequencies: 8; Small Power: 0.5; Texture Amplitude: 100.0 %; Texture Effect: Turbulence; Effect Speed: 100.0 %; Reference Object: (none); Scale X: 1 m, Y: 1 m, Z: 1 m; ✓ Automatic Preview; ✓ Use Z-Buffer in Preview

HIGHER AMPLITUDE TEXTURE

It's getting that grittiness in there better now, isn't it? Now, here's a little of that voodoo. Most places in LightWave that you would think, "Oh, 100% is the maximum" – isn't the maximum. Change that value to 500% and look at VIPER.

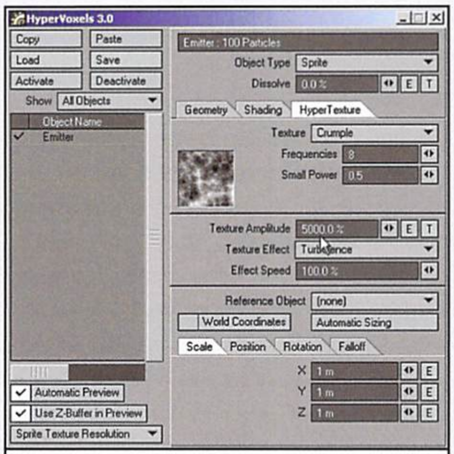

HyperVoxels 3.0 — Emitter : 100 Particles; Object Type: Sprite; Dissolve: 0.0 %; HyperTexture: Texture: Crumple; Frequencies: 8; Small Power: 0.5; Texture Amplitude: 500.0 %; Texture Effect: Turbulence; Effect Speed: 100.0 %; Reference Object: (none); Scale X: 1 m, Y: 1 m, Z: 1 m; ✓ Automatic Preview; ✓ Use Z-Buffer in Preview

HIGHER STILL

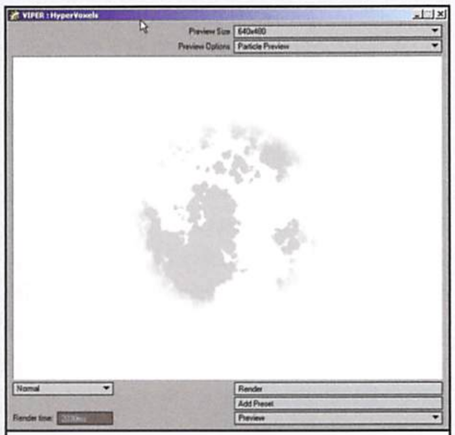

VIPER : HyperVoxels — Preview Size: 640x480; Preview Options: Particle Preview

OK, GETTING INTERESTING NOW

See what happens? Now that Crumple Texture has more character, more like puffs "inside" one voxel. Change it to 5000% and look.

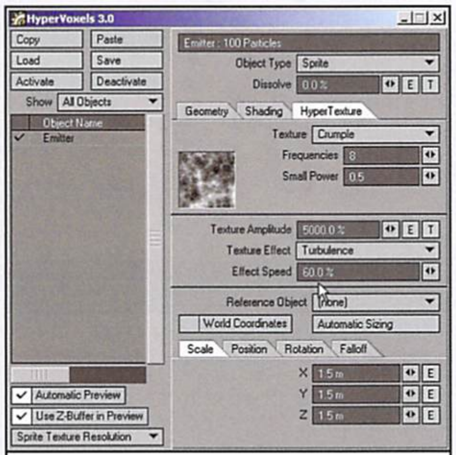

HyperVoxels 3.0 — Emitter : 100 Particles; Object Type: Sprite; Dissolve: 0.0 %; HyperTexture: Texture: Crumple; Frequencies: 8; Small Power: 0.5; Texture Amplitude: 5000.0 %; Texture Effect: Turbulence; Effect Speed: 100.0 %; Reference Object: (none); Scale X: 1 m, Y: 1 m, Z: 1 m; ✓ Automatic Preview; ✓ Use Z-Buffer in Preview

THIS GOES TO ELEVEN

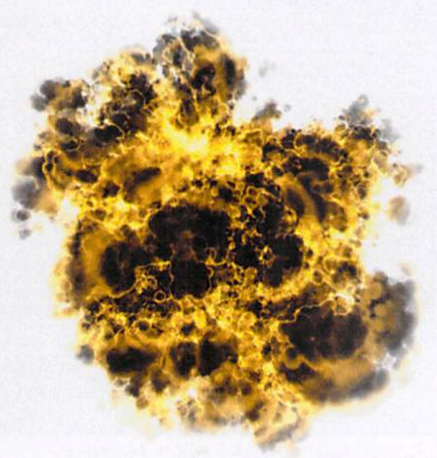

VIPER : HyperVoxels — Preview Size: 640x480; Preview Options: Particle Preview

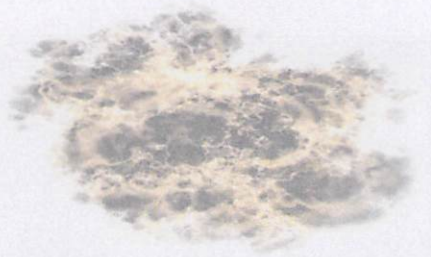
CRUMPLED !!

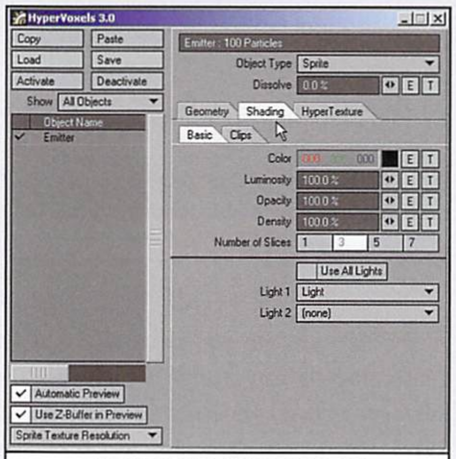
It now has nice edges inside and soft edges outside. While we're here, for my taste, and this changes on the scale of my explosions, change the Effect Speed to 60% and change the Texture Scale to 1.5 on X, Y and Z.


HyperVoxels 3.0 — Emitter : 100 Particles; Object Type: Sprite; Dissolve: 0.0 %; HyperTexture: Texture: Crumple; Frequencies: 8; Small Power: 0.5; Texture Amplitude: 5000.0 %; Texture Effect: Turbulence; Effect Speed: 60.0 %; Reference Object: (none); Scale X: 1.5 m, Y: 1.5 m, Z: 1.5 m; ✓ Automatic Preview; ✓ Use Z-Buffer in Preview

TURN UP THE EFFECT SPEED





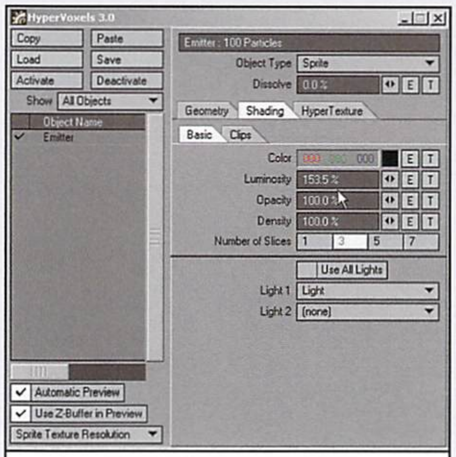


LET'S WORK IN THE SHADING TAB

Now let's work in the Shading tab.

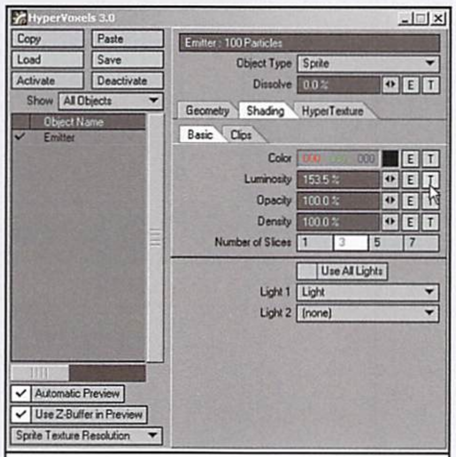
For what we're going to be working with, we really have only the three controls other than color – Luminosity, Opacity and Density. Luminosity you should know from working with surfaces; this relates to "how hot is our fire."

Well, I first approached this with "Well, it's hotter than 100%, that's for sure" – so change that value to 150-ish (remember that voodoo?). Sometimes, just grabbing the spinner and rolling up to "a higher number" works too.



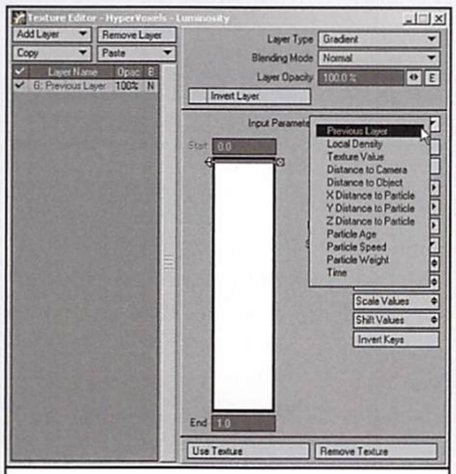
TURN ON THE LUMINOSITY

Right – it's hot, but it's not "always the same hot," now is it? Of course not. So we need to modify this, but not simply with an envelope. Hit the "T" button to add a Texture to this.



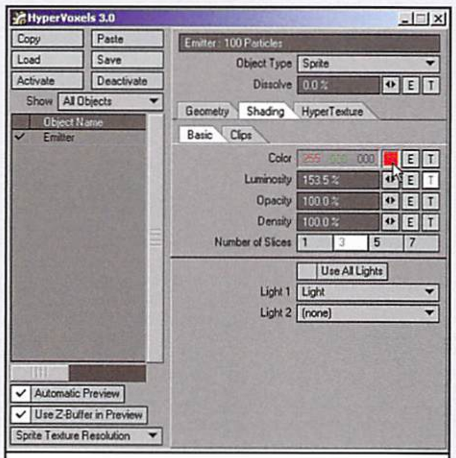
GET THAT "T"

Right. Now, a gradient would be good to use here, as it has the ability to "read" our scene and modify the changes based on inputs that we can give it. So, change to a gradient, and let's look at the options available to us.

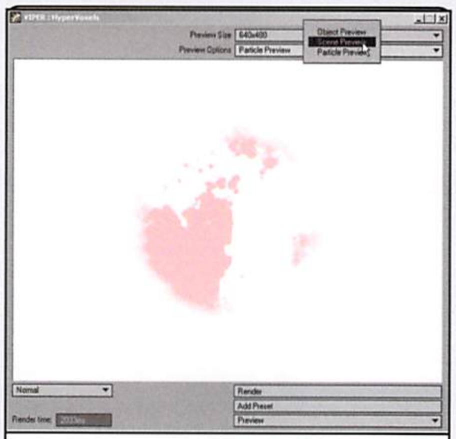


CHOICES CHOICES

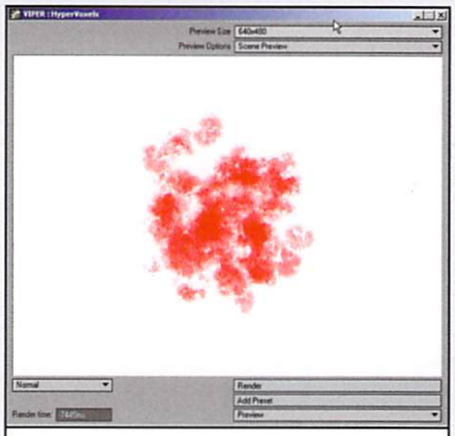
Out of this list, "Local Density" makes the most sense, as what we are wanting is to modify the "hotness" of our voxels based on "how dense our voxels are" – right? In order to really see the effect, we're going to need to change our voxel color from black to a lighter color, as "a hot black is still black" – understand? Alrighty then, let's make it red, and change our VIPER preview to Scene preview, and back up to about frame 1. Then go back into our Gradient Texture on Local Density. Here's where you are going to need to look at VIPER as we make changes.



AHHH, I LIKE RED THINGS

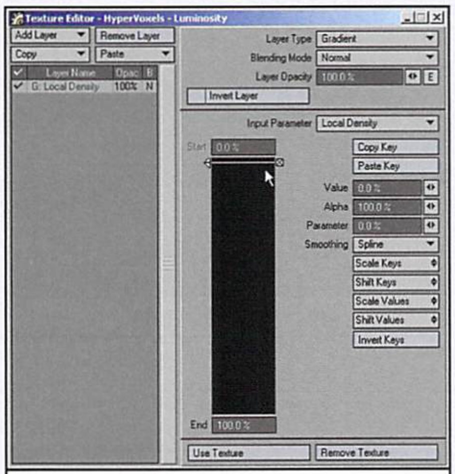


LET'S SEE HOW THE SCENE LOOKS

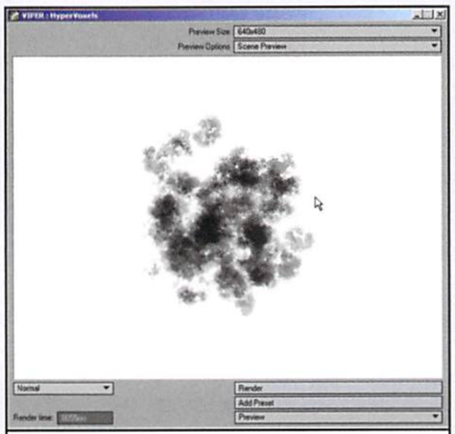


CALL IN CSI – WE HAVE A SPLATTER

The gradient is from 0% to 100% – this is the density of the voxels, with 0% being the least dense, and 100% being the most dense. So, with no other keys on the gradient, make the first and only key 0% Value and look at VIPER.


Texture Editor - HyperVoxels - Luminosity
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
G: Local Density 100% N
Input Parameter: Local Density
Start: 0.0 %
Value: 0.0 %
Alpha: 100.0 %
Parameter: 0.0 %
Smoothing: Spline
End: 100.0 %

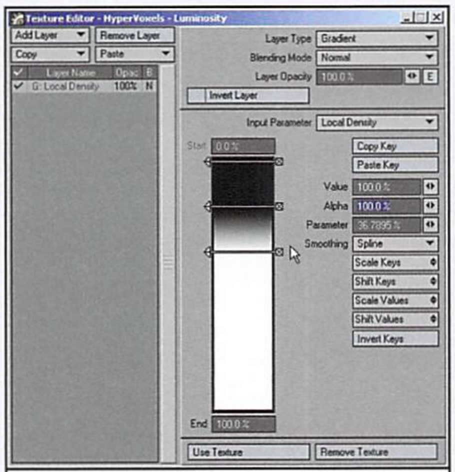
TURN DOWN LUMINOSITY



BACK IN BLACK

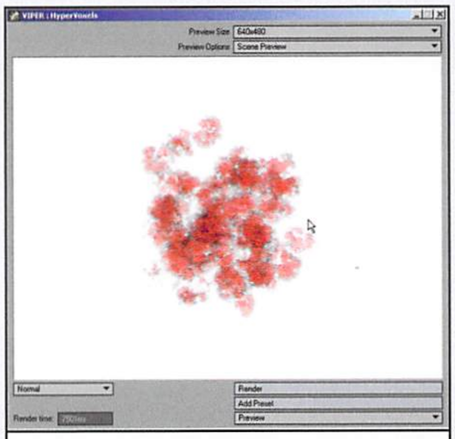
What happened to our red voxels? Well, they are "cool" or "cold" now. 0% Luminosity turned them off – and off is black.

OK, so we see this – now, let's understand "Local Density" – add two more keys on the gradient, with the next one down still at 0%, and the next 100%. Look at the screenshot to help.


Texture Editor - HyperVoxels - Luminosity
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Input Parameter: Local Density
Start: 0.0 %
Value: 100.0 %
Alpha: 100.0 %
Parameter: 36.7895 %
Smoothing: Spline
End: 100.0 %

UNDERSTANDING LOCAL DENSITY

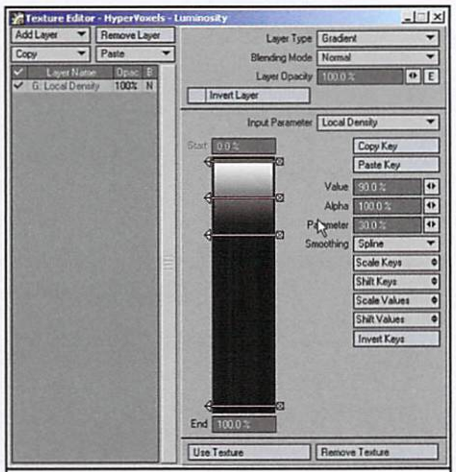
Now, with these keys – look at VIPER.



RED ON THE INSIDE, BLACK ON THE OUTSIDE

You see, where the voxels are "thicker" or more dense, they are "hot," and where they get thin out to the edges, they are black or "cool." This is what we're going to use, but because of the way fireballs look, we're going to reverse this, so that the thicker parts of the voxel are black. This will give us the "hot outer" over the black. Yet again, this is where some voodoo comes in, and the knowledge of LightWave being able to accept higher than 100% in values helps.

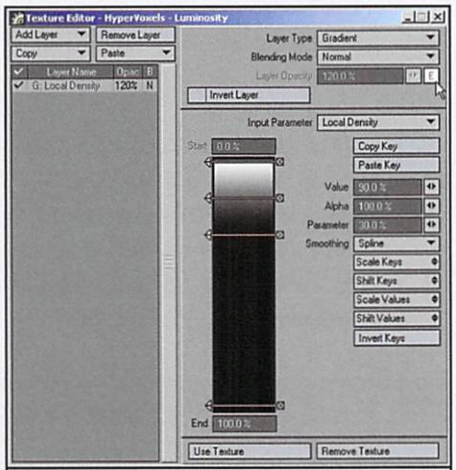
Bring out your magic, and set your keys something like this – my value for the top is 2500%, the second is 800%, and the third is 90%, with the total bottom being 0%.


Texture Editor - HyperVoxels - Luminosity
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Input Parameter: Local Density
Start: 0.0 %
Value: 90.0 %
Alpha: 100.0 %
Parameter: 30.0 %
Smoothing: Spline
End: 100.0 %

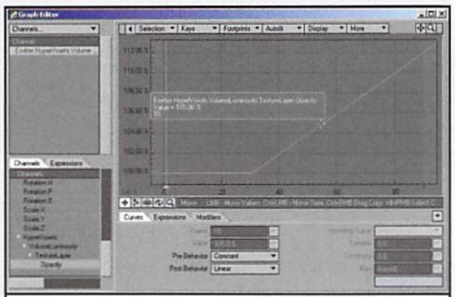
INVERT THEM

One last area to talk about here – and that is that we can change the Opacity of this layer as well over time. This will modify "how" these keys affect the voxels as the time of the explosion gets later.

Click on the "E" button beside the Layer Opacity – and put in a curve in the Graph Editor. Keyframe 100% at frame zero and at frame 30, then a Linear Constant and Forever Change through a keyframe at 55 at 105%. This will make it go darker as time goes on.


Texture Editor - HyperVoxels - Luminosity
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 120.0 %
G: Local Density 120% N
Input Parameter: Local Density
Start: 0.0 %
Value: 90.0 %
Alpha: 100.0 %
Parameter: 30.0 %
Smoothing: Spline
End: 100.0 %

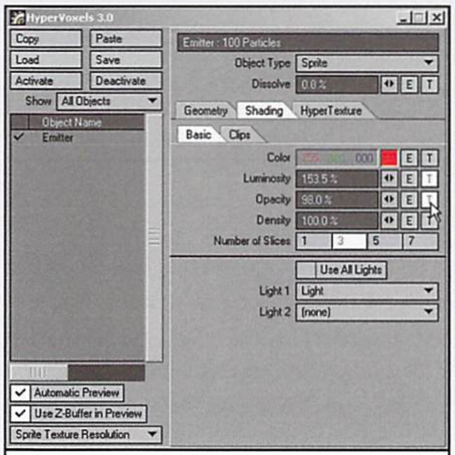
AND THE ENVELOPE SAYS . . .



THIS WILL MODIFY OUR MODIFY OVER TIME

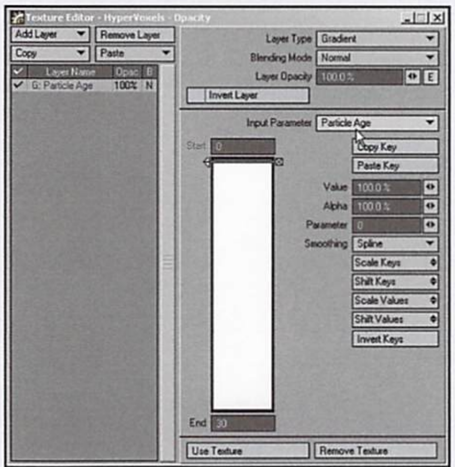
Now for Opacity. Opacity is one of those "mystic" areas of Hypervoxels that for the longest time I did not understand – but now I think of it like transparency. 0% Opacity means "it isn't there" – and 100% Opacity means it's opaque. Got it? Glad I could help. Now, as an explosion goes on in its life, it changes from gasses to thick smoke. So, as the particles age, we want them to get thicker and thicker.

Voodoo again – (laughing) – OK – I set 98% for my base, and then opened the "T" button.


HyperVoxels 3.0 — Emitter : 100 Particles; Object Type: Sprite; Dissolve: 0.0 %; Color: 255 000 000; Luminosity: 153.5 %; Opacity: 98.0 %; Density: 100.0 %; Number of Slices: 1 3 5 7; Light 1: Light; Light 2: (none); ✓ Automatic Preview; ✓ Use Z-Buffer in Preview

OPACITY – AND THE TEXTURE

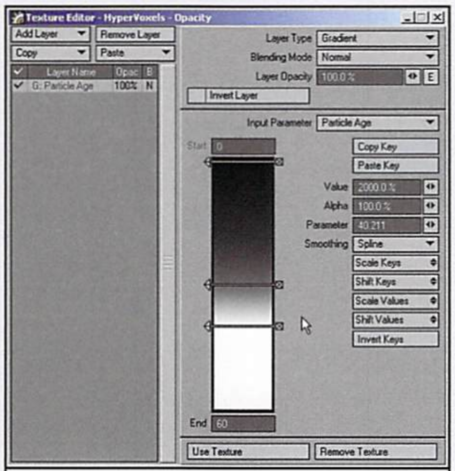
Choose a gradient, and then base that gradient on Particle Age.


Texture Editor - HyperVoxels - Opacity — Layer Type: Gradient; Blending Mode: Normal; Layer Opacity: 100.0 %; Input Parameter: Particle Age; Start: 0; Value: 100.0 %; Alpha: 100.0 %; Parameter: 0; Smoothing: Spline; End: 30

AS YOU GET OLDER, YOUR OPACITY CHANGES

I changed the end to be around 60 frames, then added keys.

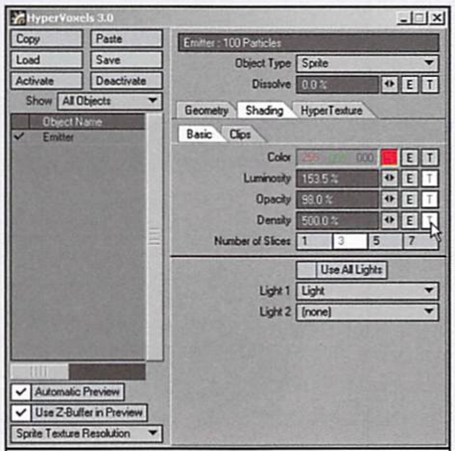
Bearing in mind, this is a "tune to taste" – my settings start with the 98%, then around frame 30 I went to 700%, finishing with 2000% by frame 40. What this is doing, is that by frame 40, we're getting very, VERY opaque. By this time of the explosion, we're going to be in mostly smoke, and we want it to be very opaque.


Texture Editor - HyperVoxels - Opacity — Layer Type: Gradient; Blending Mode: Normal; Layer Opacity: 100.0 %; Input Parameter: Particle Age; Start: 0; Value: 2000.0 %; Alpha: 100.0 %; Parameter: 40.211; Smoothing: Spline; End: 60

SET YOUR KEYS HERE

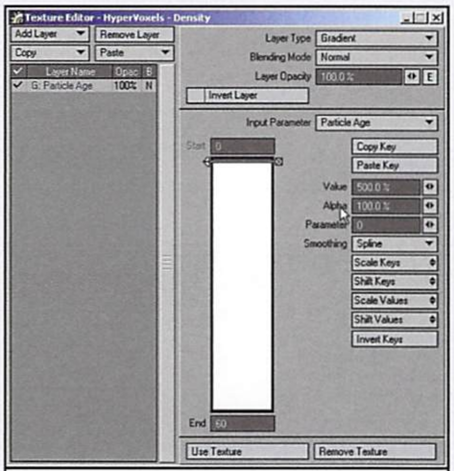
Last one. Density. This is very much like Opacity, but this is dealing with the voxels and how dense the voxels are. Remember the "Local Density" setting? This is directly related. 0% dense is gone, 100% dense is "all there." Ready? Good.

Crank the Density all the way to 500%. Then add the "T" to it.


HyperVoxels 3.0 — Emitter : 100 Particles; Object Type: Sprite; Dissolve: 0.0 %; Color: 255 000 000; Luminosity: 153.5 %; Opacity: 98.0 %; Density: 500.0 %; Number of Slices: 1 3 5 7; Light 1: Light; Light 2: (none); ✓ Automatic Preview; ✓ Use Z-Buffer in Preview

GOSH YOU'RE DENSE

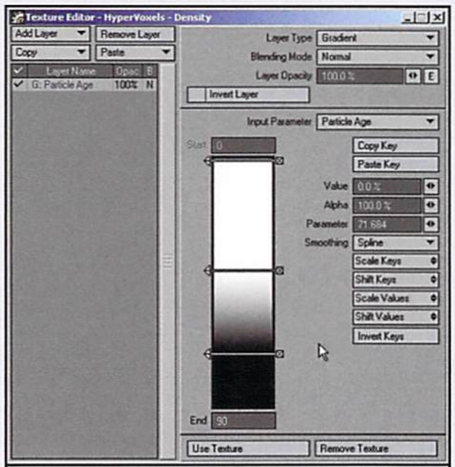
Gradient, Particle Age, and set the first key to 500%.


Texture Editor - HyperVoxels - Density — Layer Type: Gradient; Blending Mode: Normal; Layer Opacity: 100.0 %; Input Parameter: Particle Age; Start: 0; Value: 500.0 %; Alpha: 100.0 %; Parameter: 0; Smoothing: Spline; End: 60

AS YOU GET OLDER, YOUR DENSITY CHANGES TOO

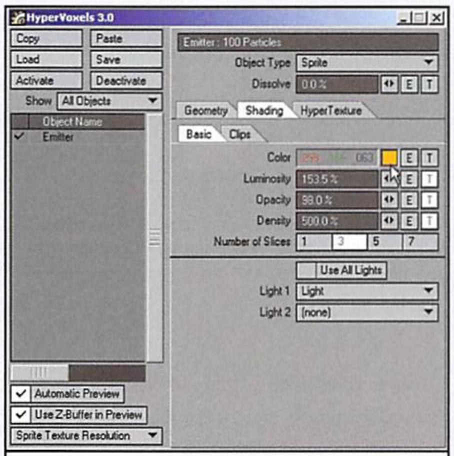
Now, over time, think of what we want the voxels to do. We want them to be more dense as the explosion runs, then less and less dense as the fireball goes away. This is the balancing act that we do with the Opacity – we're saying become more and more opaque, but less and less dense. Hey, no one said it would be easy to understand! Just trust me.

We're starting with 500% Density; we want this to stay throughout the time of the explosion's "meat," the heart of the explosion. So this value will stay the same, 500%, through about frame 40, as this is the time that we have decided the event will be. Then it will start dying out, being gone (0%) by frame 70-ish.


Texture Editor - HyperVoxels - Density — Layer Type: Gradient; Blending Mode: Normal; Layer Opacity: 100.0 %; Input Parameter: Particle Age; Start: 0; Value: 0.0 %; Alpha: 100.0 %; Parameter: 71.684; Smoothing: Spline; End: 90

DENSITY OVER TIME

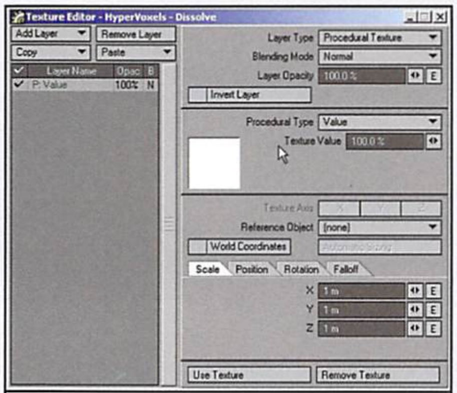
OK, let's get our color in there; this will be the base color of the fire. Here's where you can have some creative fun, as this indicates what kind of fuel is involved. Perhaps it's an alcohol; this could give it a green color, or blue, – but I'm after something along the lines of a good ol' fashioned gasoline fireball. So I'm going to set my base color around reddish-orange.


HyperVoxels 3.0
Emitter : 100 Particles
Object Type: Sprite
Dissolve: 0.0 %
Geometry / Shading / HyperTexture — Basic / Clips
Color: 063
Luminosity: 153.5 %
Opacity: 98.0 %
Density: 500.0 %
Number of Slices: 1 3 5 7
Use All Lights
Light 1: Light
Light 2: (none)
Show: All Objects
✓ Emitter
✓ Automatic Preview
✓ Use Z-Buffer in Preview
Sprite Texture Resolution

NOW FOR FIRE COLOR

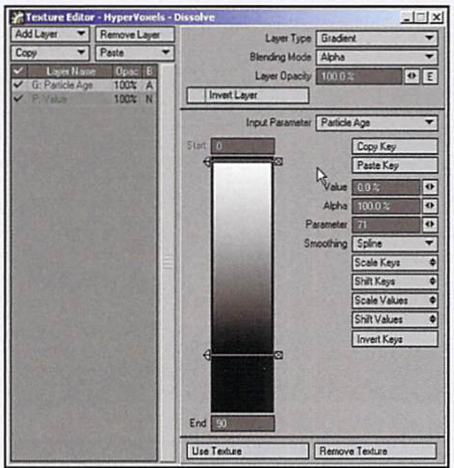
Now, our explosion life is about 70 frames. We need to have our voxels dissolve out by that time, but I want to have more control than simply putting in an Envelope. So click on the "T" button beside Dissolve.

First thing we're going to add is a Procedural texture called "VALUE" – and make it 100%. I know what you're probably thinking, "Why not just set it to 100% in the Hypervoxel panel instead of here?" – well, that setting on the main panel for Dissolve is an all overriding setting. We couldn't do what we're going to do with it set there. Just follow, and you'll see.


Texture Editor - HyperVoxels - Dissolve
Layer Name: P: Value — Opac 100% — B N
Layer Type: Procedural Texture
Blending Mode: Normal
Layer Opacity: 100.0 %
Invert Layer
Procedural Type: Value
Texture Value: 100.0 %
Reference Object: (none)
World Coordinates
Scale / Position / Rotation / Falloff
X: 1 m
Y: 1 m
Z: 1 m
Use Texture / Remove Texture

THIS HAS ADDED VALUE

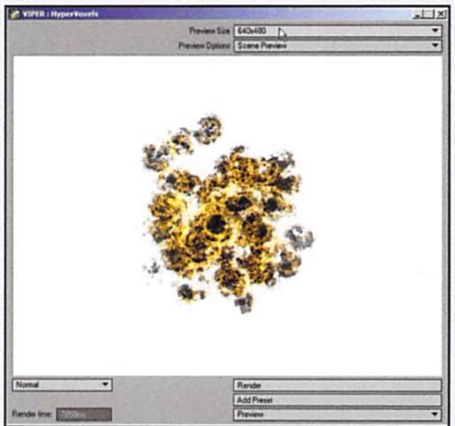
Now, on top of that, add a Gradient, based on Particle Age, then set the Start key at 68% and set another key at around 71 to 100%. Why 71? Seemed right to me. See screenshot if you're lost with my description – then, set this to an ALPHA layer. Ahhhh, see why we needed the Value? We needed to have this gradient on top of a base that was "totally dissolved out." No amount of reversing the layer or anything would give this control without the Value added.


Texture Editor - HyperVoxels - Dissolve
Layer Name: G: Particle Age — Opac 100% — B A
Layer Name: P: Value — Opac 100% — B N
Layer Type: Gradient
Blending Mode: Alpha
Layer Opacity: 100.0 %
Invert Layer
Input Parameter: Particle Age
Start: 0
Value: 0.0 %
Alpha: 100.0 %
Parameter: 71
Smoothing: Spline
End: 90
Copy Key / Paste Key / Scale Keys / Shift Keys / Scale Values / Shift Values / Invert Keys
Use Texture / Remove Texture

SETTING THE GRADIENT ALPHA

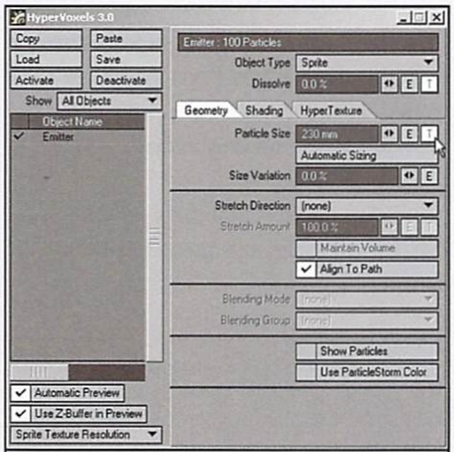
As the particles' ages increase, they will dissolve out over their age, being gone at or around 71 frames.

Hey, take a look at VIPER! Looking pretty good? I think so! But it needs that final touch. The particles right now, or rather the voxels right now, are staying the same size all through the animation. Let's make that a little more "organic."



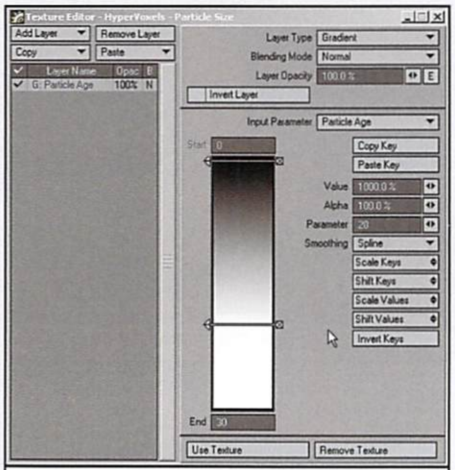
LOOKING KINDA LIKE A FIREBALL

I modified my Start Particle size to 230mm; depending on your use or scene, you might look at changing this; then I clicked that magic "T" button.


HyperVoxels 3.0
Emitter : 100 Particles
Object Type: Sprite
Dissolve: 0.0 %
Geometry / Shading / HyperTexture
Particle Size: 230 mm
Automatic Sizing
Size Variation: 0.0 %
Stretch Direction: (none)
Stretch Amount: 100.0 %
Maintain Volume
✓ Align To Path
Blending Mode: (none)
Blending Group: (none)
Show Particles
Use ParticleStorm Color
Show: All Objects
✓ Emitter
✓ Automatic Preview
✓ Use Z-Buffer in Preview
Sprite Texture Resolution

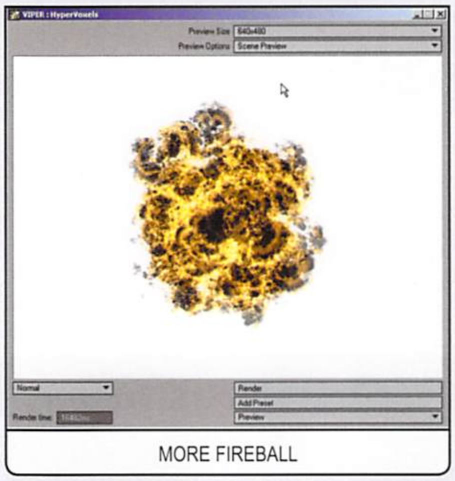
YES, MAKE IT BIGGER

You probably guessed, gradient based on Particle Age. But this works a little differently. Most places that you have been adding a gradient override the value on the outside panel. Meaning, that a texture overrides the setting that you have on the Main voxel panel. Particle Size is different. The values you add here do indeed modify the value on the voxel panel. So, with a 230mm Starting Size, I set my starting percentage to 10%, for "very small," then by frame 20, the biggest that they will get, I set the value to 1000% for "very, very big." This is really cool, because if you change your scale on the main panel, then the settings in here don't have to change.


Texture Editor - HyperVoxels - Particle Size
Layer Name: G: Particle Age — Opac 100% — B N
Layer Type: Gradient
Blending Mode: Normal
Layer Opacity: 100.0 %
Invert Layer
Input Parameter: Particle Age
Start: 0
Value: 1000.0 %
Alpha: 100.0 %
Parameter: 20
Smoothing: Spline
End: 30
Copy Key / Paste Key / Scale Keys / Shift Keys / Scale Values / Shift Values / Invert Keys
Use Texture / Remove Texture

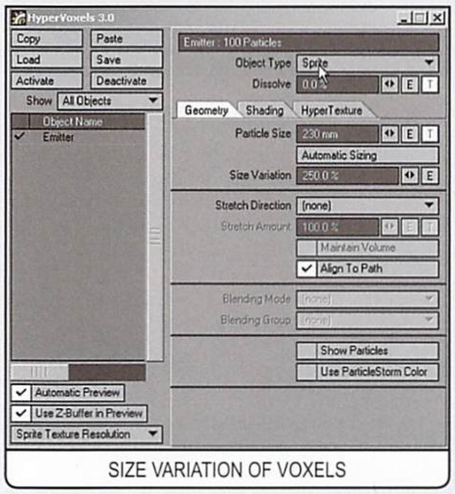
AS YOU GET OLDER, YOU GET BIGGER

And hey, look at VIPER now! *GRIN*

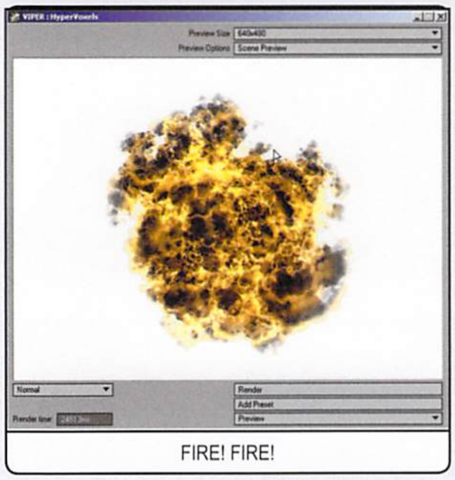

VIPER : HyperVoxels
Preview Size: 640x480
Preview Options: Scene Preview
Normal
Render
Add Preset
Preview

MORE FIREBALL

OK, final tweaks. Size variation is wonderful. It makes all the voxels different. I like a large value here; I use 250%

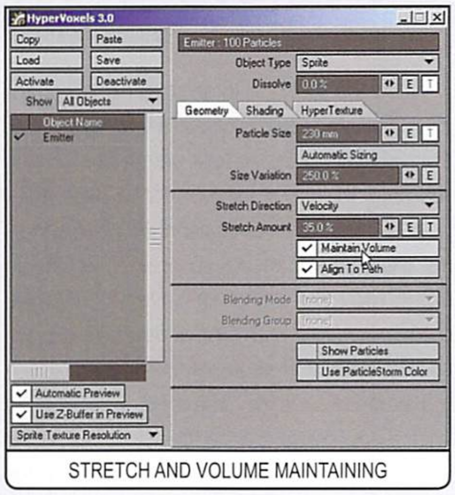

HyperVoxels 3.0
Copy | Paste
Load | Save
Activate | Deactivate
Show: All Objects
Object Name
✓ Emitter
Emitter : 100 Particles
Object Type: Sprite
Dissolve: 0.0 %
Geometry | Shading | HyperTexture
Particle Size: 230 mm
Automatic Sizing
Size Variation: 250.0 %
Stretch Direction: (none)
Stretch Amount: 100.0 %
Maintain Volume
✓ Align To Path
Blending Mode: (none)
Blending Group: (none)
Show Particles
Use ParticleStorm Color
✓ Automatic Preview
✓ Use Z-Buffer in Preview
Sprite Texture Resolution

SIZE VARIATION OF VOXELS

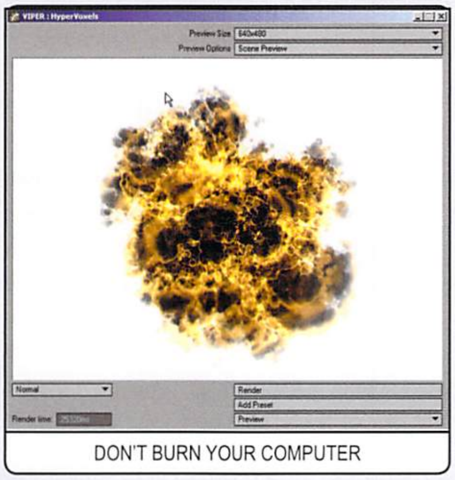

VIPER : HyperVoxels
Preview Size: 640x480
Preview Options: Scene Preview
Normal
Render
Add Preset
Preview

FIRE! FIRE!

Stretch Direction I use Velocity, about 30-45% and maintain Volume. Adjust to taste; this is just me.


HyperVoxels 3.0
Copy | Paste
Load | Save
Activate | Deactivate
Show: All Objects
Object Name
✓ Emitter
Emitter : 100 Particles
Object Type: Sprite
Dissolve: 0.0 %
Geometry | Shading | HyperTexture
Particle Size: 230 mm
Automatic Sizing
Size Variation: 250.0 %
Stretch Direction: Velocity
Stretch Amount: 35.0 %
✓ Maintain Volume
✓ Align To Path
Blending Mode: (none)
Blending Group: (none)
Show Particles
Use ParticleStorm Color
✓ Automatic Preview
✓ Use Z-Buffer in Preview
Sprite Texture Resolution

STRETCH AND VOLUME MAINTAINING


VIPER : HyperVoxels
Preview Size: 640x480
Preview Options: Scene Preview
Normal
Render
Add Preset
Preview

DON'T BURN YOUR COMPUTER

I hope that after you finish reading this for the first time, then once through with LightWave running, you will have two things. One is a fireball preset that you can use, and two is the understanding of how we got there so you can modify and make your own.

Now, just don't get any burning embers in your eye – but that's another story for another time.

JACK "DEUCE" BENNETT II IS A FREELANCE CGI ARTIST WITH A BACKGROUND IN PHYSICAL SPECIAL EFFECTS FOR MOTION PICTURES AND TELEVISION. DEUCE HAS BEEN WORKING IN THE FILM INDUSTRY HIS ENTIRE LIFE, AND HAS SUCH MOVIES AS *ROBOCOP*, *LONESOME DOVE*, AND *JIMMY NEUTRON: BOY GENIUS* TO HIS CREDIT, AS WELL AS TV SHOWS LIKE *WALKER, TEXAS RANGER*. DEUCE HAS BEEN USING COMPUTERS SINCE HE WAS 9, AND STARTED OFF WRITING HIS OWN GRAPHIC PROGRAMS. HE IS A UNIQUE COMBINATION OF PHYSICAL KNOWLEDGE AND VIRTUAL KNOW-HOW.

BY DAN ABLAN

COOL EFFECTS & More in Mirage

Bauhaus Software has created a spectacular program that can help you in your daily work. Sure, that's a pretty general statement, but when you learn what Mirage can do, you can find a way it can be used in your work.

Say you have an image sequence you need to compile and save out to Quicktime, or an AVI, or any other compressed format. Mirage will do that for you. Perhaps you need to blend two animations or video clips together; you can do that as well. But that's nothing, as Mirage is great for film, video post, broadcast, 2D Animation, 3D Sweetening, Pre-visualization, and even Matte Painting. Bauhaus Mirage is a unified environment that simplifies the creation of animated graphics and special effects. Combining real-time video paint, animation, and effects functionality into a single product, Mirage centralizes the workflow for visual effects production in film, video, broadcast, and cartoon/2D animation. To begin, it's hard to say what the true power of Mirage is, as it can be used for so many things. One of the first things you'll discover with Mirage is that it has a Sub-Pixel Accurate Paint system. This paint system can be animated, to create simple things like animated signatures, to moving images over video, and even fully animated drawings and cartoons. Another cool feature of Mirage is the FX Stack, where you can apply multiple effects to an image or sequence. Beyond that, Mirage offers a powerful keying system for compositing. You should also know that Mirage offers an integrated particle system, which is an excellent way to add additional effects to your 3D sequences or video clips.

But instead of just listing the cool things Mirage can do, how about working through a simple tutorial on your own? This basic tutorial will demonstrate the cool volumetric lighting effect offered in Mirage. It's one of my favorites.

Now, most people think of using this effect through text, which is of course possible – but you've seen that. A few years ago, I was teaching the artists at ABC-TV in New York how to use LightWave, and they were showing me some of their awesome After Effects work. They had a shot of a News helicopter flying over New York City, and as I watched it, streaks of light bled out of the aircraft, which transitioned to their animated logo. It was such a cool effect, but as always, it was done with an expensive third-party plug-in for After Effects. So, when I saw that Mirage had a volumetric lighting feature, I immediately thought of these types of effects. Going one step further, Mirage can do it faster. *Figure 1* shows the final effect produced by Mirage.

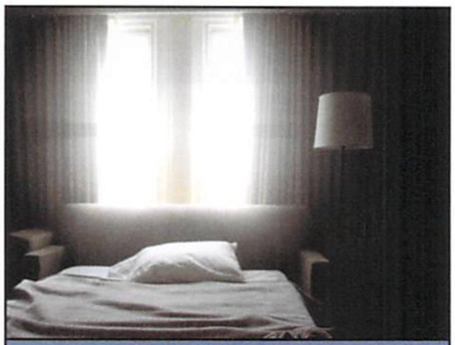


FIGURE 1– VOLUMETRIC LIGHTING ISN'T JUST FOR TEXT, BUT FOR GREAT LOOKING IMAGE ENHANCEMENTS

1 Begin by firing up Mirage. You'll need to set the initial product resolution to whatever image you choose to use. Conversely, you can also set a larger resolution, as Mirage will allow you to adjust the project size later. Once Mirage has been started, adjust the interface so that your primary view is the project workspace, making the timeline smaller and out of the way, such as in *figure 2*.

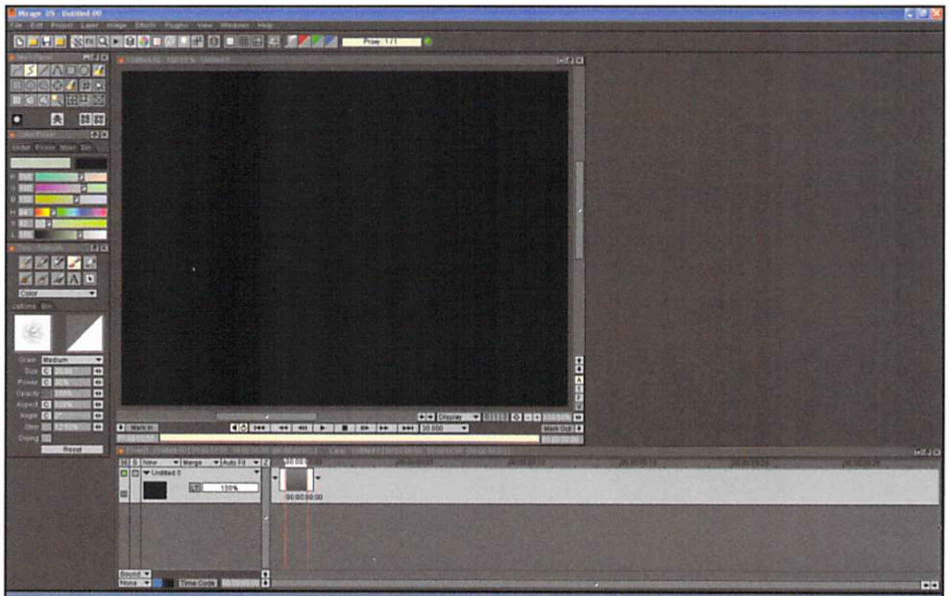


FIGURE 2– SET UP YOUR PANELS IN MIRAGE TO SOMETHING LIKE THIS, SO YOU CAN SEE YOUR ENTIRE WORKSPACE

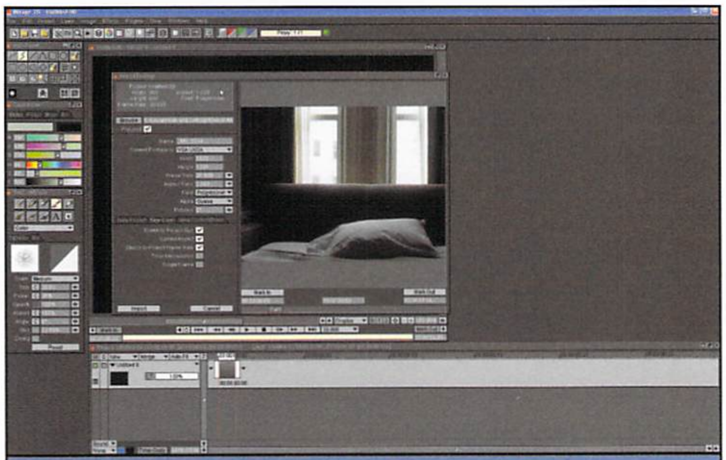


FIGURE 3– IMPORTING IMAGES OR CLIPS AUTOMATICALLY OPENS THE IMPORT FOOTAGE PANEL

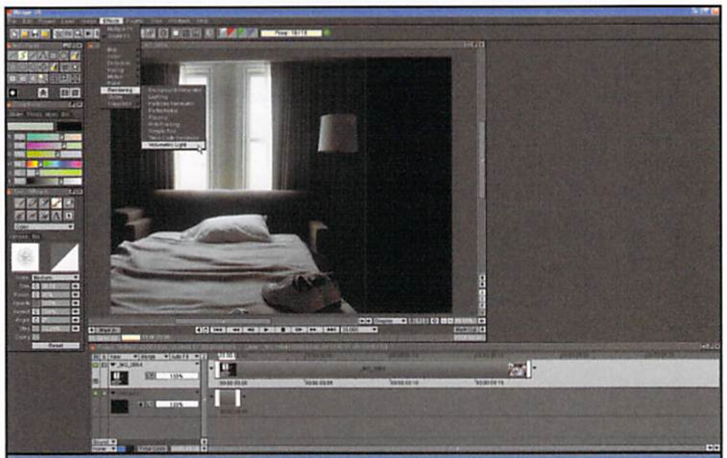


FIGURE 4– ADDING THE VOLUMETRIC LIGHT EFFECT TO YOUR IMAGE IS AS EASY AS A SELECTION

2 The next step is to load an image. You can do this by going to the File dropdown at the top left of the interface and selecting Load. But what is even easer is to simply open your folder with an image you want to use and drag it into the Mirage workspace. Once you import an image or clip, you'll see the Import Footage panel, as shown in *figure 3*.

3 In this panel, you can choose to convert the footage or clip, use an Alpha channel and more. For now, just click Import at the bottom of the panel. Your image will be placed in the project area. You'll also see the clip entered into the Layers panel.

4 Now, go up to the Effects dropdown from the top of the interface, then go to Rendering and choose Volumetric Light. *Figure 4* shows the selections.

5 You'll note that there are other effects in this area you can experiment with on your own. For now, you'll see the FX Stack open with the Volumetric Light effect added, as in *figure 5*.

6 When you add the Volumetric Light effect, you'll immediately see what it's doing on your image. You'll also see a marker labeled "center." You can click the green square and move the volumetric's origin. For this tutorial, I'm using an image of a room with two windows. You can download this image from www.3dgarage.com/downloads. Click and drag the green square to the center of the windows. This will make the volumetric light generate from that position. *Figure 6* shows the change.

7 Now let's focus on the FX Stack. At the top, you'll see a check mark with the name of the Effect you're working with. Right now, you only have one effect, but down the road, as you have multiple effects in the FX Stack, this setting will be beneficial, as you can turn an effect on or off quickly and easily. Note that with the FX Stack open, you can quickly add additional effects (or remove them) from the controls at the top of the panel.

8 Move down to the center of the FX Stack and you'll see the Channel listing. Here, set this to Luminosity, rather than the default setting. What Mirage will do is create the volumetric light from the luminous portions of the images; in this case, the windows. *Figure 7* shows the change and effect.

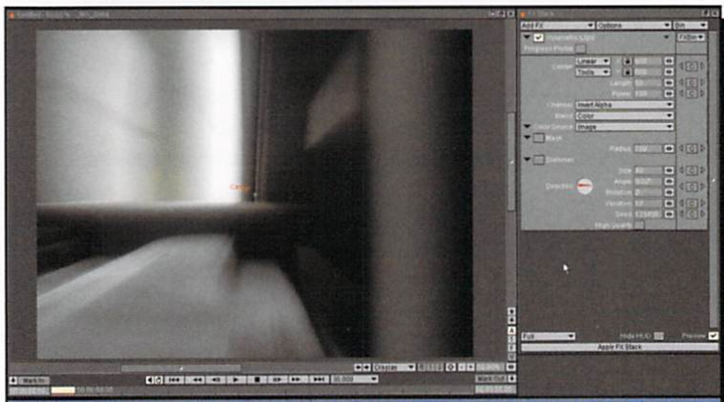


FIGURE 5– IMPORTING IMAGES OR CLIPS AUTOMATICALLY OPENS THE IMPORT FOOTAGE PANEL

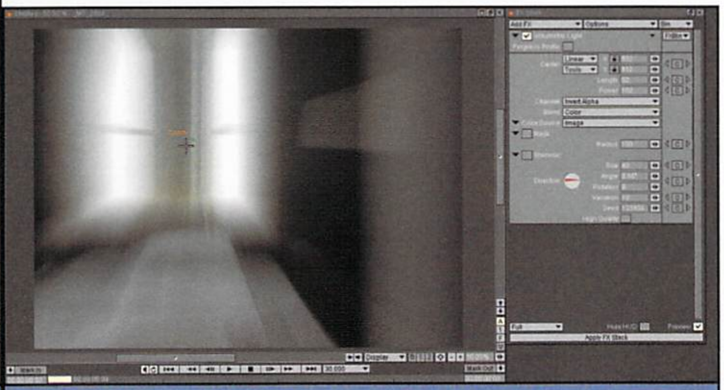


FIGURE 6–MOVE THE GREEN SQUARE TO THE CENTER OF THE WINDOWS SO THAT THE VOLUMETRIC IS GENERATED FROM THIS POSITION.

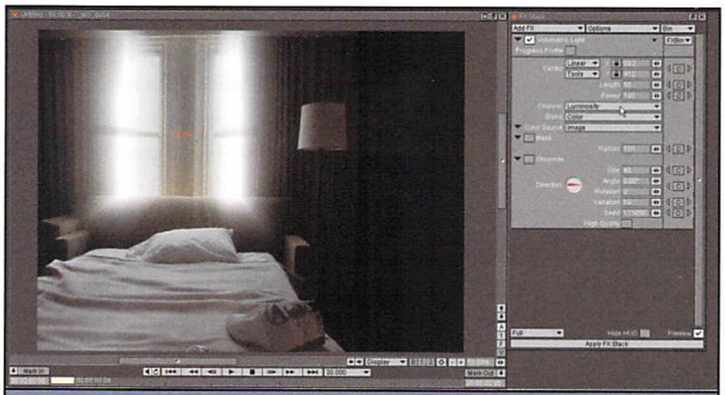
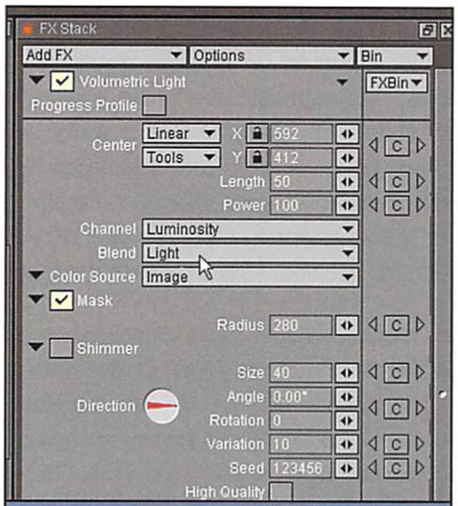


FIGURE 7– CHANGING THE CHANNEL SETTING TO LUMINOSITY MAKES THE VOLUMETRIC LIGHT STREAK FROM THE WINDOWS

9 The next category beneath the Channel setting is Blend. This allows you to change the way the volumetric mixes (blends) with the rest of the image. You can choose to add to the image, subtract, multiply, and so on. For this image, try setting this to Light, so that you have the effect as in *figure 8*.

10 You can change the color of the volumetric by setting the Color Source to Image, Gradient or Color. For now, allow the image to color the volumetric.

11 You might have noticed that the volumetric, although streaking from the window, also slightly streaked from the lampshade. Because the lamp has some luminous quality compared to the rest of the image, the volumetric effect applies here as well. Therefore, click the Mask button in the FX Stack. You can easily mask the volumetric effect with this setting, and you can adjust the mask by clicking and dragging the red control handles right on the image, or setting a value in the FX Stack. For this image, a value of 280 works well. *Figure 9* shows the change.



FX Stack
Add FX | Options | Bin
Volumetric Light | FXBin
Progress Profile
Center: Linear X 592; Tools Y 412
Length 50
Power 100
Channel Luminosity
Blend Light
Color Source Image
Mask
Radius 280
Shimmer
Size 40
Direction: Angle 0.00°
Rotation 0
Variation 10
Seed 123456
High Quality

FIGURE 8– BLEND THE VOLUMETRIC BY SETTING THE VALUE TO LIGHT

12 Because a mask is now in place, set the Length value to 80, and the Power to 145. The Power setting increases the amount of volumetric, while the Length increases the edge of the volumetric, essentially softening its falloff.

13 One of the last things you can do is apply a Shimmer. This basically adds streaks to the volumetric light, which you can then vary with Rotation, Size and more.

14 Finally, click Apply FX Stack at the bottom of the FX Stack panel to apply the effect. Then save out the image from the File menu.

The volumetric lighting effect is very easy to set up as you've seen here, and Mirage's fast interface allows you to quickly see your work, which makes setting this effect up sort of fun. But taking this tutorial one step further, you can animate all of the values! So, imagine this project taken to the next level. The photo of the room is instead video. As the camera moves in towards the window, the volumetric light effect is increased with an animated Power and Length setting. A few simple keyframes can suddenly bring a regular image or video sequence to life. Mirage is so powerful that it's hard to express without a 1000 page book, and there are so many possibilities that we'll continue to provide Mirage tutorials here in HDRI 3D.

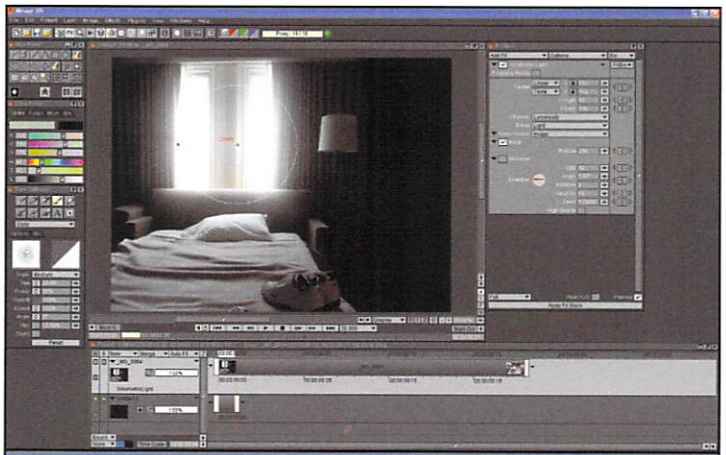
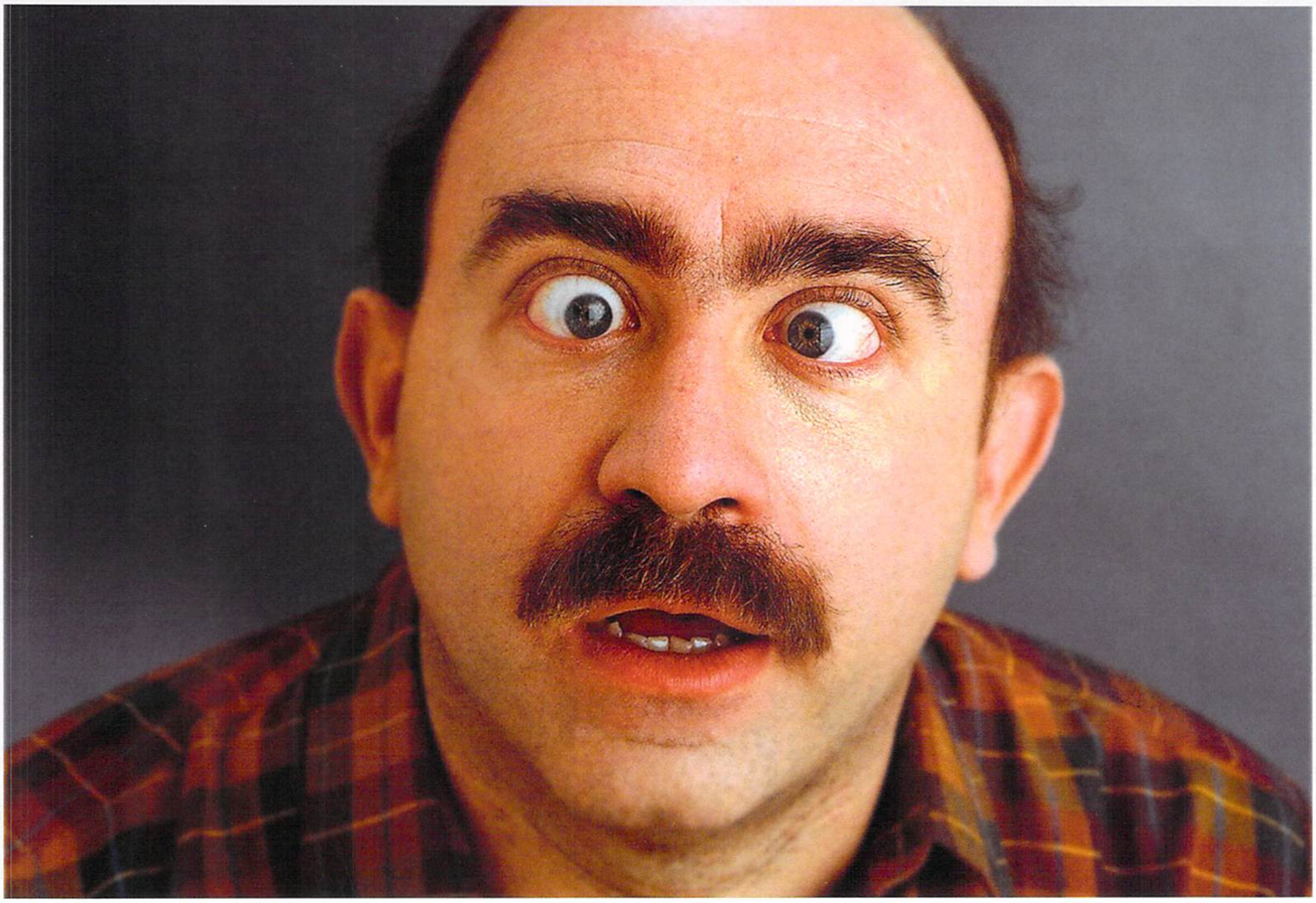


FIGURE 9– MASKING THE VOLUMETRIC HELPS CONTROL ITS EFFECT

DAN ABLAN IS PRESIDENT OF AGA DIGITAL STUDIOS, INC. IN CHICAGO, IL., AND FOUNDER OF 3DGARAGE.COM. HE HAS WRITTEN EIGHT BOOKS FOR *NEW RIDERS PUBLISHING* ON 3D AND COMPUTER GRAPHICS, AND HAS RECENTLY PUBLISHED LIGHTWAVE [8]: KILLER TIPS (WITH RANDY SHARP). DAN HAS CREATED A 20 HOUR LIGHTWAVE COURSE ON CD-ROM, YOU CAN FIND OUT MORE ABOUT THE COURSE AT 3DGARAGE.COM. DAN IS ALSO THE EDITOR-IN-CHIEF OF HDRI 3D MAGAZINE.



GET FOCUSED!

with the

High Quality. DVD-ROM. 60+ hours.

3Dgarage.com

3D | GARAGE™

LightWave® 3D Courseware

from Dan Ablan

Training from the author!

From NASA to the US Army, to NBC and FOX, to hobbyists, people around the globe are *finally* understanding the power of LightWave® 3D. See for yourself why the 3D Garage™ courseware is rated the best training you can find.

Download free tutorials now at **3D GARAGE.com**

AGA DIGITAL

AGA DIGITAL STUDIOS, INC. 1305 Wiley Road Suite 125 Schaumburg, IL 60173
847.755.4488 p 847.755.6401 f www.agadigital.com | www.3dgarage.com

Others promise. We deliver.

MIRAGE

PINNACLE SYSTEMS

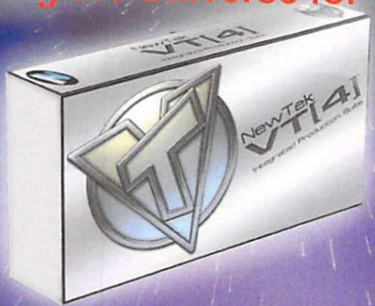
ANARK

LEITCH

DIGITAL FUSION

T.S. Computers

Serving the Universe for Over 15 Years



Workstations
Render Nodes
File Servers
Training

New!! 800 MHz FSB XEONS
VT [4] - Fusion 5 - Liquid 6
Ultra - DV Rack
Get Your Upgrades Now

11300 Hartland Street North Hollywood, CA 91605 (818) 760-4445 www.vgn.com

---

HDRI 3D

M · A · G · A · Z · I · N · E

Go to www.hdri3d.com for more info!

ONLINE SPECIAL OFFER

❑ US $49 ❑ Rest of World $68

Subscribe to HDRI 3D Magazine online before December 31, 2004 to get this great discount!
Six issues a year

HDRI 3D presents real world insights and emerging trends in digital production from the digital artist's perspective.

Subscribe today and ***SAVE!***

Lighting the way for digital artists

---

encompass *media*

97 South 5th Ave.
West Reading, Pa 19611

Integrated Production Suite

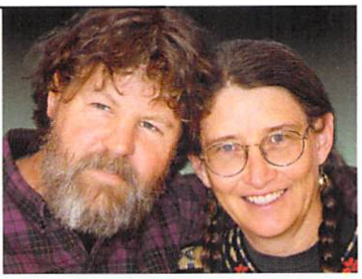
VT[4]TM Systems
Now Available!

NewTek

For Custom Built VT[4]TM Systems
Call 610.372.1010 or Toll Free 800.570.7300
www.encompassmedia.net
Authorized NewTek Dealer Since 1989

A RESOLUTION ON THE RESOLUTION OF RESOLUTION

BY ANDREA CARVEY



I finally got it! Well, I think I got it. Perhaps not entirely, but I think I've pretty much figured out the whole resolution thing. This has been a problem for me for a very long time – something that others apparently knew by osmosis or having been abducted by aliens and programmed with alien information during their "visit" (which explains why many of these people are a little "different"). At any rate, everyone else seemed to have fewer problems with it than I did.

So, why is it so confusing, one might ask? Gee, let's see! I'd hear the word "resolution" just about every time I turned around. "Hey, what's the resolution of that image?" Or, "That's a cool monitor. What's its resolution?" Or "To get that to look good in print, you'll have to render it at a higher resolution." Or "I'm going to set the camera's resolution for video, so the image will be smaller." OK, do you get the problem?

I'd run into big time difficulties when I'd try to use an image as a texture for a model. Things would be going along fine, then ... "Huh? What the heck?! Why is it the size of Nebraska?" After trying vainly to correct the situation (even resorting to math), I'd end up brute forcing it by sizing the image down 3000%. Not very elegant. Another common way to entertain myself, rather than go out and enjoy a beautiful day, was to try to print an image that I'd rendered. "It looked so great on the screen! What happened?" "Maybe the printer wasn't working, again. It just printed out a blank page." "No, wait. What is that tiny speck in the center?" "Ahh! Apparently the resolution was all wrong!" Again.

After spending an hour or two trying to figure it out myself, I'd slink into Brad's studio hoping that he had some "free" time (sure) to go over "resolution" with me – yet again. (For those of you who don't know, Brad is my own personal technology wizard, without whom I would be happily, blissfully, and peacefully digging up ancient treasures.) I'd ask Brad what I was doing wrong, and he'd patiently try to explain about pixel size, number of pixels on a monitor, how they can be changed, how to increase and decrease resolution, etc. etc. etc., until my eyes would glaze over and I'd be transported into my favorite TV commercial for Corona beer – my own mini-vacation – complete with palm trees, sandy beach, lapping water, soft tropical breeze, quiet, dreamy ... "Do you get it, now?" he'd ask. Oh crap! What the heck was he talking about? I can't ask him to go over it again! I'd mumble something to the effect that I indeed got it, and go back and try to figure it out myself. Usually, I employed the tried-and-true Trial and Error method. This method is guaranteed to work – eventually. The disadvantage is that it takes a lot of time, but there is one subsidiary advantage. It is very productive when your supply of scrap paper is running low.

It was getting to the point that I was asking Brad about resolution once a month or so. His patience was wearing thin and my confidence was all shot to hell. I just couldn't get it. Now, there are very few things that I just can't get – most of which I don't give a rat's ass about anyway – such as why Ben Affleck and Sarah Jessica Parker are considered to be such hotties. But when Brad began to just nonchalantly mention to people that resolution was just something that I would never grasp I had to do something!

So, I got into my obsessive, Don't-Stop-Until-You-Understand-It mode. I gathered up a blank notebook, a handful of my favorite pencils (I hate mechanical pencils), and crammed a whole ream of paper down the craw of the printer. I was going to figure this out!

I spent a lot of time changing and recording parameters, rendering, printing, and drawing little diagrams. Slowly, I began to see a pattern, confusing at first, then suddenly the fog lifted. Ah ha! What we had here was a definitional problem. No wonder I was confused. Now the question was ... Why wasn't everyone else confused? The same term was being used to refer to at least three different phenomena. It's like using the word "cat" for cats, laundry and peanuts. I know what cats are, but when someone tells me to put the cats in the dryer, or to go toss some cats in the backyard for the squirrels, I get a little confused. Other people, apparently, have no problem with this.

What I figured out is that apparently the term "resolution" is used to refer to all of the following:

• DISPLAY SETTINGS – such as the number of pixels or dots per inch that a monitor displays or that a printer prints

• THE DENSITY OF AN IMAGE – the number of pixels or dots per inch of an image

AND

• THE DIMENSIONS OF THE IMAGE OR MONITOR – such as 600 x 400 pixels, which is also referred to as the "size" of the image or monitor. However, "size" also often refers to the amount of memory needed to store an image (e.g. 50 K).

You can see the problem. But, why aren't other people confused? At first, I thought that its context in a sentence distinguished one meaning from another, and I, being unaware of this, was lost in a "Who's on First?" world. So, when someone said, "What's the resolution of that image?" everyone (except me) just knew that they meant the dimensions of the image. Unfortunately, try as I might, there just doesn't seem to be anything in the context of that sentence that would give clues as to its real meaning. Perhaps there is some subliminal message that I never received. Maybe it's because I didn't watch *Star Trek.* Ahhh! I'm back at the beginning! I just don't get it!

BRAD AND ANDREA CARVEY HAVE BEEN DOING COMPUTER ANIMATIONS FOR A LONG TIME. IN 1969 BRAD USED AN ANALOG COMPUTER, WHICH WAS THE SIZE OF A CAR, TO PRODUCE HIS FIRST COMPUTER ANIMATION. ANDREA, AN ARCHEOLOGIST, PREFERS TO DO SCIENTIFIC ANIMATIONS; HER CREDITS INCLUDE PROGRAMS LIKE DISCOVERY CHANNEL'S *UNDERSTANDING CARS.* BRAD IS AN ELECTRICAL ENGINEER AND AN EMMY AWARD-WINNING MEMBER OF THE VIDEO TOASTER DEVELOPMENT TEAM. HE PREFERS TO DO FEATURE FILM WORK. HIS CREDITS INCLUDE FILMS LIKE *MEN IN BLACK, STUART LITTLE, BLACK HAWK DOWN, KATE & LEOPOLD* AND *MASTER OF DISGUISE.*

3-Democracy.

never before have **so many**, had access to **so much**, for **so little**.

↓

buy XSI today:

$495

Get free training!

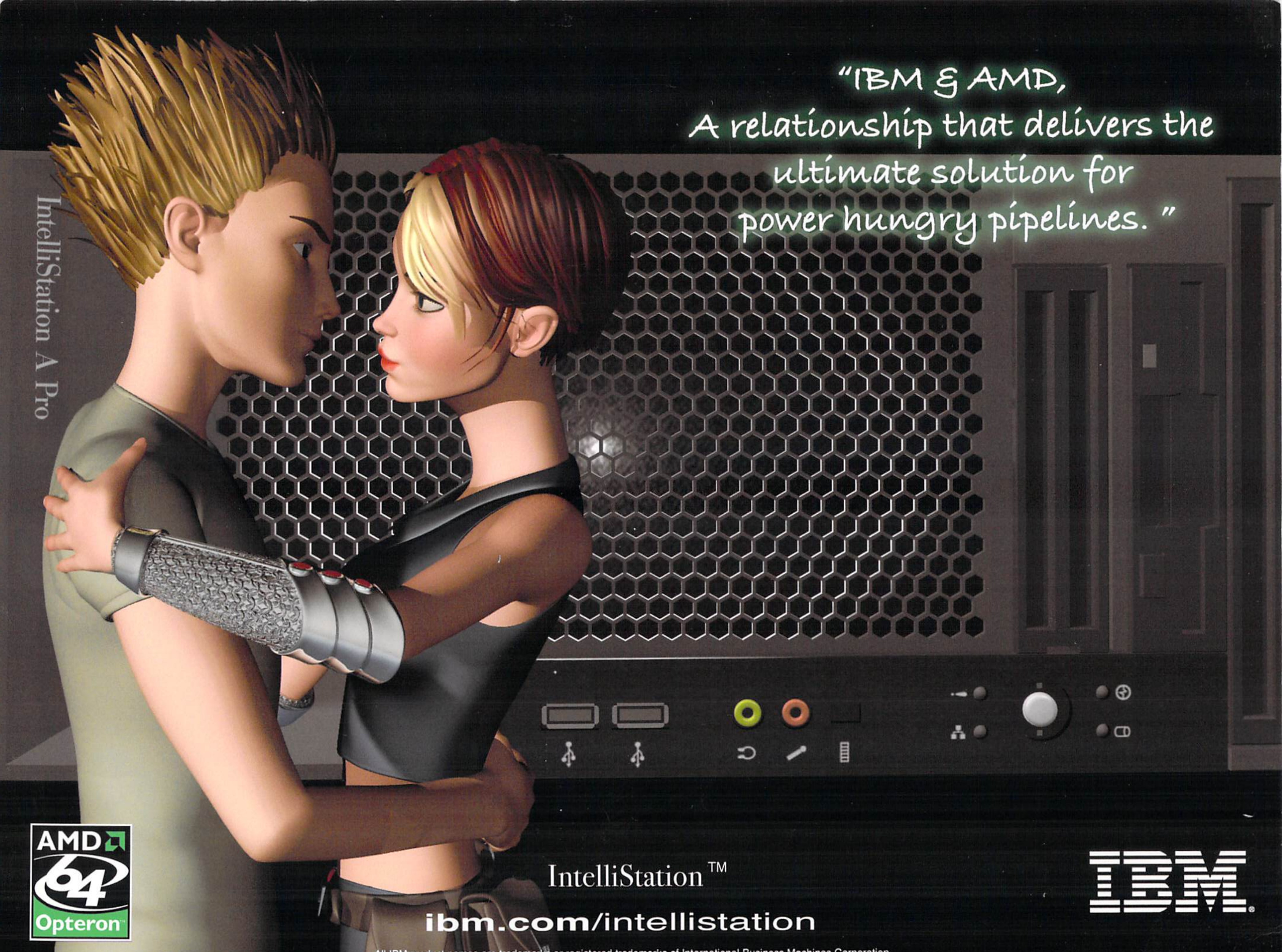
!

try it free for

30 days

softimage.com/XSI

SOFTIMAGE|XSI v.4.0 Foundation **$495**


"IBM & AMD,
A relationship that delivers the
ultimate solution for
power hungry pipelines."
IntelliStation A Pro
AMD 64 Opteron
IntelliStation™
IBM
ibm.com/intellistation