



G N O M O N   S C H O O L   P R E S E N T S

# H I G H E N D

M A G A Z I N E

ISSUE NUMBER 3 • SUMMER 2003



## THE HULK

THE MAKING OF A CGI HYPER-HUMAN

**Creating Custom Glows**

**Production Methods**

USING PASSES TO SPEED UP YOUR WORK

**Using XSI to Extract**

NORMAL & DISPLACEMENT MAPS

**Character Design-Detail  
Management**

ISSN 1541-5864



03

0 74470 97212 9  
Issue #3 \$12.95 USD



# GNOMON

School of Visual Effects

Specialized training for careers in high-end  
computer graphics for the entertainment industries.

**323.466.6663**

**[www.gnomon3d.com](http://www.gnomon3d.com)**

- » Courses on the tools and techniques of the digital visual effects industry
- » Courses on fundamental skills such as production design and miniatures
- » 21-month certificate program, extension courses or custom training
- » 9-Week intensive Maya Fast Track Program
- » Over 15 individual hands-on Maya classes
- » Classes held on evenings and weekends
- » Maya DVDs, videos and tutorials from the Gnomon Workshop
- » Maya streaming-video training and forums at Gnomon Online

"I can't say enough good things about my experience at Gnomon. Not only did the Certificate Program help me cultivate the necessary skills to make that happen, it provided tremendous focus and perspective and gave me the confidence to set my goals high. The program is truly first-rate, as are the members of the Gnomon faculty who are well-connected, successful industry professionals who have a passion for inspired education."

Steeg Haskel, Associate Cinematic Artist  
Blizzard Entertainment



Gnomon Inc. School of Visual Effects  
1015 North Cahuenga Blvd.  
Hollywood, CA 90038  
323.466.6663  
[www.gnomon3d.com](http://www.gnomon3d.com)



Images © 2002 Gnomon Students: Ergin Kuke, Seth Hall, Paulo De Almada, Tyson Cross, Dave Pasciuto, Barry Gelber.



## CONTRIBUTORS

---



ED HARRISS is a CG artist living in the USA. Currently, he runs a CG-oriented website dealing mostly with mental ray, SOFTIMAGE | XSI, SOFTIMAGE |

3D, some of his personal work (<http://www.EdHarriss.com>). He works for a company called Alternate Route Studios. (<http://www.altroutestudios.com>)



PAUL HOURMOUZIS IS A FREELANCE 3D ARTIST LIVING IN MELBOURNE, AUSTRALIA. HE'S BEEN INVOLVED WITH COMPUTERS AND CG SINCE THE

AGE OF 15 WHEN HE STARTED OUT RUNNING DELUXE PAINT AND IMAGINE ON AN AMIGA SYSTEM. AFTER COMPLETING A DEGREE IN COMPUTER SCIENCE, HE ENJOYED A SUCCESSFUL CAREER AS A BUSINESS DEVELOPMENT MANAGER SELLING ENTERPRISE INFORMATION TECHNOLOGY. DURING THAT TIME, HIS LIFE-LONG PASSION AND INTEREST AS A 3D ARTIST TRANSFORMED INTO A LABOR OF LOVE, WHERE HIS PORTFOLIO OF WORK NOW INCLUDES TELEVISION COMMERCIALS AND CORPORATE ARTWORK.

PAUL CAN BE CONTACTED VIA EMAIL AT [PHOURMOUZIS@HOTMAIL.COM](mailto:PHOURMOUZIS@HOTMAIL.COM)



BERNARD LEBEL IS A 3D ARTIST LIVING IN MONTREAL, CANADA. HE CURRENTLY WORKS AT CINÉGROUPE, AS A LIGHTING SUPERVISOR. AUTHOR OF

XSI - PRODUCTION SOLUTIONS AND TIPS, 13 HOURS OF VIDEO LESSONS FOR XSI USERS, YOU CAN FIND MORE ABOUT HIM AT [WWW.BERNARBLEBEL.COM](http://WWW.BERNARBLEBEL.COM).



FRANCOIS LORD IS A CG SUPERVISOR/ANIMATOR AT BUZZ IMAGE GROUP ([WWW.BUZZIMAGE.COM](http://WWW.BUZZIMAGE.COM)) IN MONTREAL, CANADA. SINCE HIS

GRADUATION FROM THE NAD CENTER ([WWW.NAD.QC.CA](http://WWW.NAD.QC.CA)) IN 1996, HE HAS BEEN WORKING IN 3D IN VARIOUS COMPANIES IN MONTREAL. HE WORKED MOSTLY ON COMMERCIALS, BUT ALSO ON SOME MOVIES. HE HAS ALSO TAUGHT CAMERA TRACKING (MATCH MOVING) AT NAD SINCE 2000.

**Writers** — Ed Harriss, Bernard Lebel, Paul Hourmouzis, Francois Lord, Andrea Maiolo, Bobby Milly, Olivier Ozoux

### CALL FOR WRITERS

If you think you are qualified to write for Highend Magazine, e-mail us at: [writers@highendmag.com](mailto:writers@highendmag.com)



GNOMON SCHOOL PRESENTS

**HIGHEND**  
MAGAZINE

ISSUE NUMBER 3 • SUMMER 2003

### Publishers

CHARLES A. EDGIN [charles@dmgpublishing.com](mailto:charles@dmgpublishing.com)

ALICE M. EDGIN [alice@dmgpublishing.com](mailto:alice@dmgpublishing.com)

### Editors

WILL MCCULLOUGH [will@highendmag.com](mailto:will@highendmag.com)

ALEX ALVAREZ [alex@highendmag.com](mailto:alex@highendmag.com)

DARRIN KRUMWEIDE [darrin@highendmag.com](mailto:darrin@highendmag.com)

### Managing Editor

AMBER GODDARD [amberg@dmgpublishing.com](mailto:amberg@dmgpublishing.com)

### Technical Editors

ALEX ALVAREZ [alex@gnomon3d.com](mailto:alex@gnomon3d.com)

DARRIN KRUMWEIDE [darrin@gnomon3d.com](mailto:darrin@gnomon3d.com)

ED HARRISS [www.edharriss.com](http://www.edharriss.com)

BERNARD LEBEL [www.bernardlebel.com](http://www.bernardlebel.com)

### Art Director

CHRISTINA WILKINSON [chris@sabredesign.net](mailto:chris@sabredesign.net)

### Design/Layout

Sabre Design [artists@sabredesign.net](mailto:artists@sabredesign.net)

### Highend Logo Design

DMITRY SAVINOFF [dmitry@savinoff.com](mailto:dmitry@savinoff.com)

---

**HIGHEND MAGAZINE** (ISSN 1541-5864) is published quarterly by DMG Publishing, 2756 North Green Valley Parkway #261, Henderson, Nevada 89014

ADVERTISING SALES

[sales@highendmag.com](mailto:sales@highendmag.com)

CIRCULATION & SUBSCRIPTIONS

[subscriptions@highendmag.com](mailto:subscriptions@highendmag.com)

Application to mail at periodicals postage rates is pending at Henderson, Nevada, and additional mailing offices.

### POSTMASTER:

Send address changes to [subscriptions@highend.com](mailto:subscriptions@highend.com) or to Subscriptions, DMG Publishing, 2756 N Green Valley Pkwy #261, Henderson NV 89014



# SIGGRAPH and where it leads us.



**Will McCullough**  
Editor — Highend Magazine

Maybe you went, maybe you wanted to go, but surely you've been to one at some point. Over the past few years SIGGRAPH has changed. After attending this year in lovely downtown San Diego (Gaslamp district has some cool restaurants and clubs; Alex recommends the Strip Joint Steak restaurant; I recommend Café Sevilla for salsa dancing and great Spanish food) and getting a chance to meet many of you face to face it has become clear amongst industry professionals exactly what SIGGRAPH means now. SIGGRAPH is now our social event, our chance to gather in one place and meet, greet, and make contacts. It is our industry mixer event. With news posted on dozens of websites minutes after the release of products and with video demos downloadable from every software company's web site there was very little talk at SIGGRAPH about what was cool or what was new. The topic of discussion was more of what is going on in the industry, who is doing what, and where we are going in our respective industries. Sure there were a few eye catching new products, but nothing we hadn't already heard about. It is about connecting with each other in person as artists: in work, and in play.

But the demographics of SIGGRAPH have also changed. Now you see thousands of students bussed in from colleges and high schools all with the interest of getting into the "3d animation field." Throughout the crowds the industry veterans are easily recognized and you stop to quickly give a run down of your next six months with your old pals. As our industry becomes more mainstream and as more of everyone picks up a copy of Maya and Shake we need to find ways to proactively grow the professional user communities without seeming like elitists.

The best way to do this is to create, to innovate, to distinguish the differences and lead our industries with new paradigms of creativity for a whole new generation of visual effects. With so many competitors there will be very few more Tron's and less and less Matrix's to wow each other with. But pushing the details of reaching the reality of whatever job is handed to you will be the ultimate testament of growth and creative accomplishment. Sharing with each other, demonstrating our methods, teaching. These are the areas that SIGGRAPH has become valuable. This is what Highend Magazine stands to help deliver to your doors every few months.

The result will be simultaneously inspiring, educating, and entertaining. Opening the doors to epiphanies of shots and previz scenarios playing out like stories in your imagination while you watch and read. Preparing for your next job or just keeping a backup trick in mind. It is to each other that we owe the wisdom that makes us successful professional digital artists. Thanks for the inspiration!

A handwritten signature in black ink, reading "Will McCullough".





## FEATURE

### THE HULK

It's not easy being green! Modeling and creating a CGI hyper-human.

BY ANDREA MAIOLO

4

## SHAKE

### CREATING CUSTOM GLOWS WITH SHAKE

Using macros in Shake make it possible to create custom glows with many parameters.

BY FRANCOIS LORD

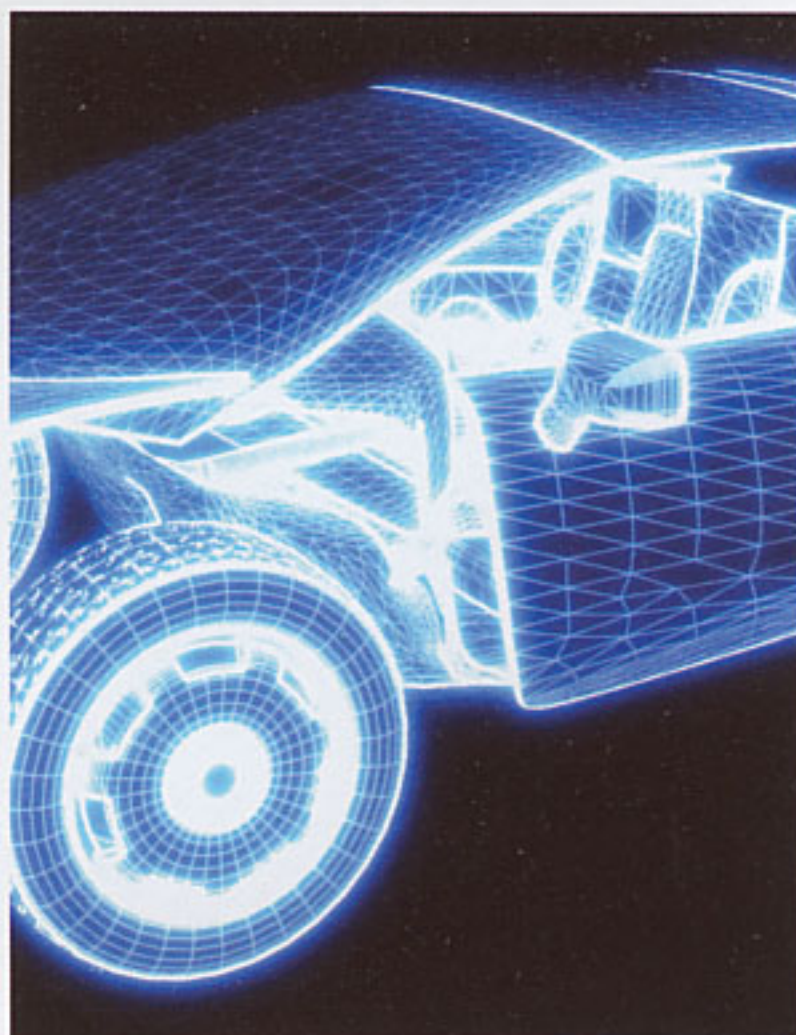
34

## ARTIST SPOTLIGHT

### VINCENT THOMAS

52

PAGE 34



PAGE 16



PAGE 4

## TUTORIALS

### REAL-TIME REFLECTIONS

Some of the process that the video game industry has created to achieve a clear visual representation.

BY BOBBY MILLY

12

### PRODUCTION METHODS

Using passes to speed up your work.

BY ED HARRISS AND BERNARD LEBEL

16

### USING XSI TO EXTRACT NORMAL & DISPLACEMENT MAPS

BY OLIVIER OZOUX

26

### CHARACTER DESIGN-DETAIL MANAGEMENT

A review of the tools and techniques for preparing high-resolution digital characters

BY PAUL HOURMOUZIS

44

## GALLERY

38

COVER ART:

Provided by

Industrial Light & Magic





# The Hulk

BY ANDREA MAIOLO



The Hulk was supposed to be a really big muscular guy. We had to find a way to represent the intricate motion of nerves, muscles, veins and tendons moving underneath the skin.





A year and a half ago we were asked to build a big green man in CG. We thought: "How hard can it be?" CG humans or semi-humans have been on the screen for a while now. The real challenge would be making him better, more realistic, more integrated with the scene, more sensitive and real. The goal was to make it the best CG actor appearing on the big screen to date. Eighteen months later, looking back, we realized how difficult it was to reach our goal. When we started on this project we knew that we had to try to overcome traditionally hard problems to solve in CG.

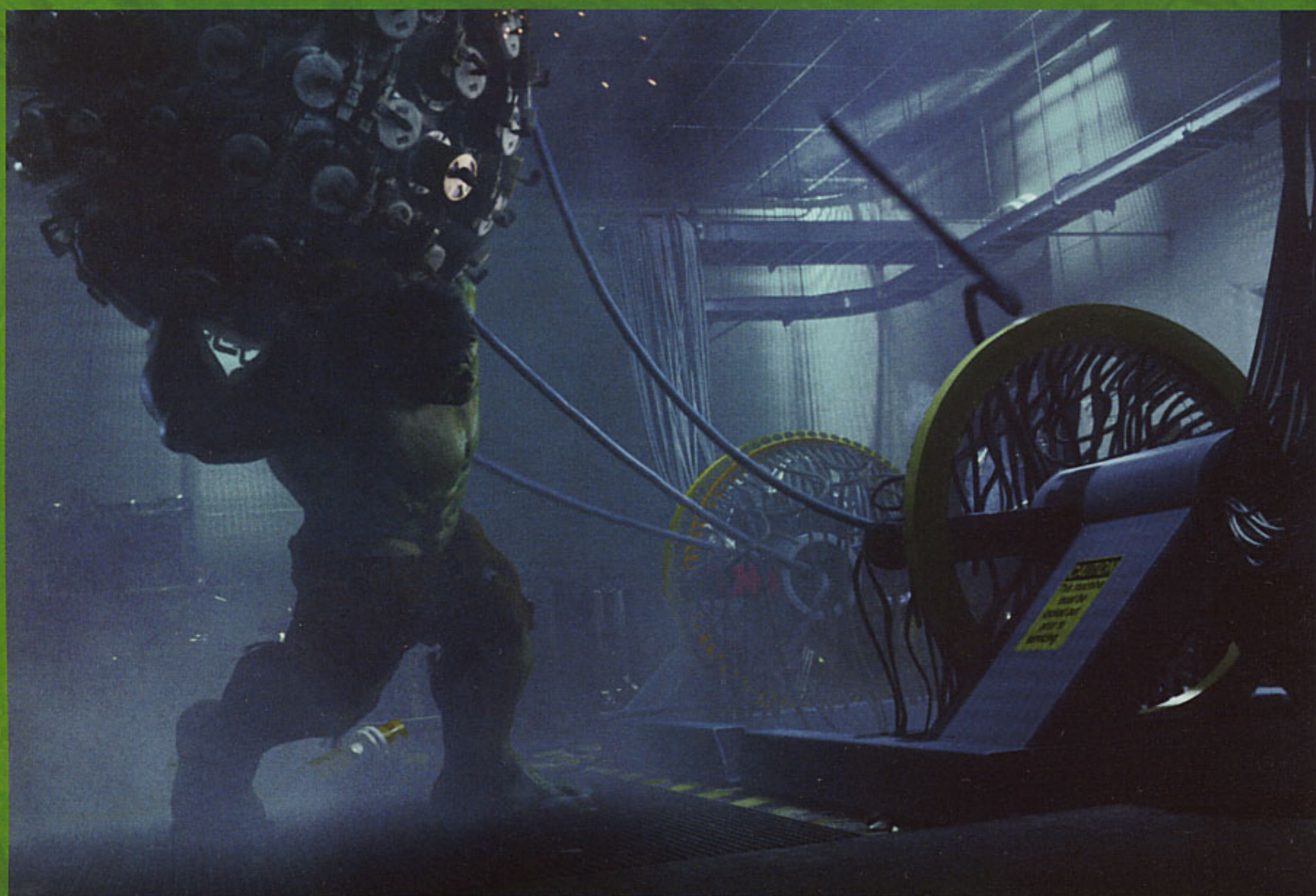
The Hulk was supposed to be a really big muscular guy. We had to find a way to represent the intricate motion of nerves, muscles, veins and tendons moving underneath the skin. He was going to be basically naked for the entire length of his appearance on the screen, so his skin had to display all the properties that we recognize so easily and are so familiar with. The skin had to be able to stretch, compress and slide over the underneath layers, plus have pores and display that typical translucency of organic tissues. The hair had to react to the motion of his head and his emotion had

The goal was to make it the best CG actor appearing on the big screen to date.



PHOTOS BY INDUSTRIAL LIGHT & MAGIC





THE MODELING DEPARTMENT DID AN AMAZING JOB CREATING ROUGHLY 300 SHAPES TO CONTROL HIS BODY.

to be revealed through his mute synthetic face. He had to be able to interact with objects in the set and transform to and from his human alter ego. Also, all of this had to look natural while cutting back and forth between him and real actors. But let's start from the beginning and go one step at a time.

The Hulk is a mix of really seasoned and really new techniques. His rigs existed in different variations for different packages. The animation rig existed in SOFTIM-

AGE|3D and Maya. Big attention was paid by the experienced ILM rigger to create a stable and solid setup. The animators could then choose their preferred animation package based on their preferences and experiences.

The skeleton was driving a polygonal proxy mesh that was carrying extra information. Among them there were transformations in

charge of dynamically driven motion. When the animation was completed and approved, a character TD was in charge of running the file through a software originally written for "Jurassic Park 3." This was performing a "ballistic simulation" on the locators that were giving broad, dynamically correct, secondary motion to certain areas. These were then baked out and eventually modified to accommodate directorial decisions. The jiggle nulls, as they were called, were attached to the final mesh through a secondary envelope. This gave the Character TD the ability to decide exactly the influence and the decay of each area for each jiggling component.

Instead of adopting a more anatomically accurate muscle system like that one used in EP2, the ILM crew decided to go for a more flexible and completely con-





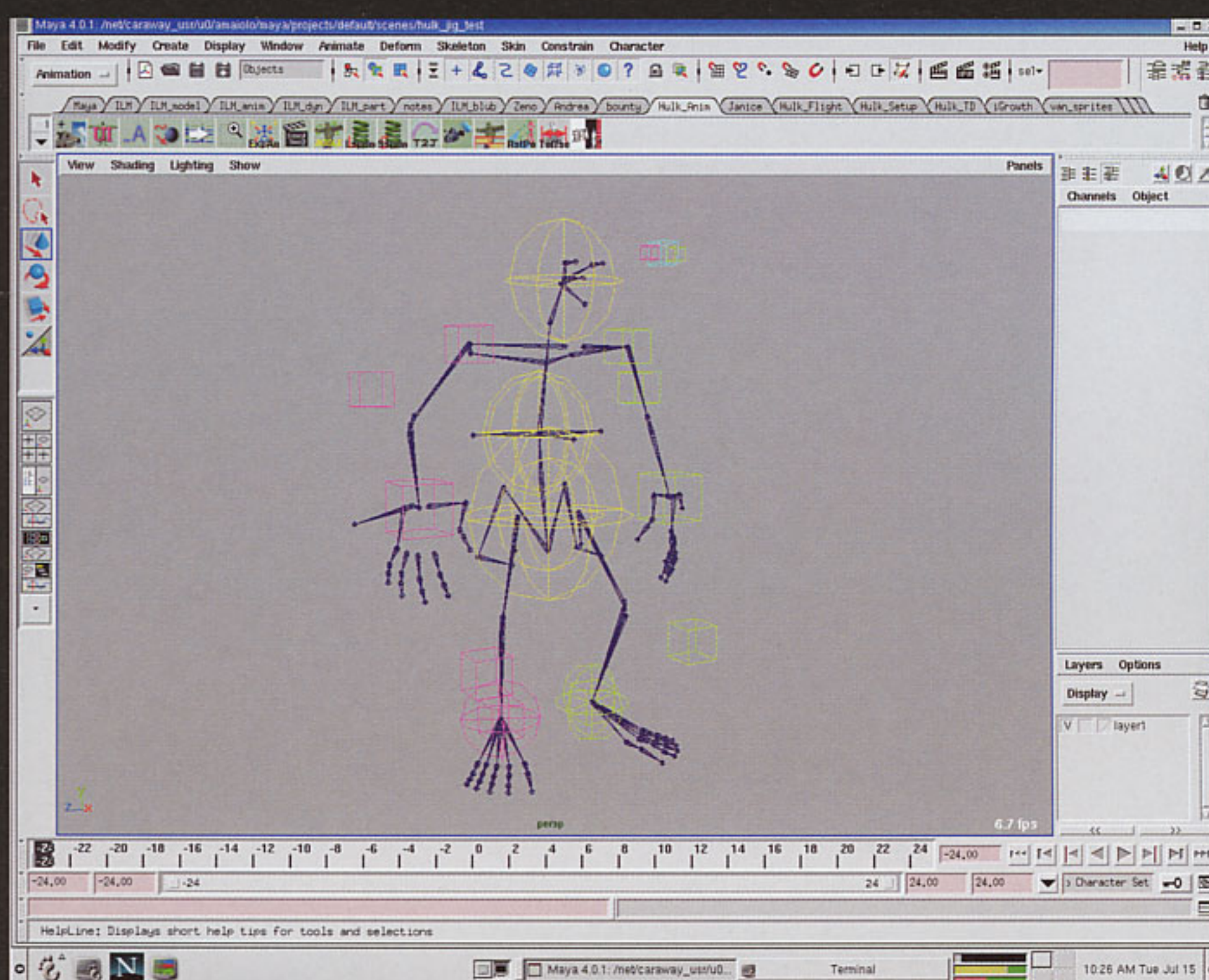
trollable pose approach. This method consists of determining a number of key poses for each articulation and then sculpting the way each one of them looks. Cari, one of our proprietary tools, was then modified to be able to properly interpolate and blend the different shapes while the Hulk was moving. On top of that, each muscle was modeled in its flex and relaxed states so that the animator or the Character TD could decide where and when the Hulk would flex his muscles. The modeling department did an amazing job creating roughly 300 shapes to control his body. The Hulk body was modeled in BSplines exploiting a proprietary ILM tool that was getting fed by a point cloud obtained from the scan of a Marvel approved maquette. The facial expressions were done through a series of morph targets in our proprietary animation software. It had the advantage of a really fast feedback and allowed us to animate the face without the need of proxy objects. On top of that, it had a custom tool specially designed to control stuff like the position and rotation of the eyes. The envelope was done with proprietary volumetric enveloping tools.

Another duty of the Character TD was taking care of the sliding of the skin on the tissues underneath. Using proprietary software, when the animation was done and the motion of the muscles approved, a flesh simulation was run to relax, compress and slide the skin around. This gave an extra layer of realism, especially on mid and close shots.

The hair was a joint effort between Character TD and the TD in charge of rendering the shot. The model of the

Hulk got created with a bunch of NURBS planes coming out of his scalp. These were converted in cloth strips and simulated accordingly, adding wind when needed. At render time, the NURBS patches were used to grow curves from and eventually interpolate position between them. Each curve was rendered in Renderman as a RCurve and a special anisotropic shader was assigned to them to simulate the hair reaction to light.

A completely different story was taking place when the Hulk had to perform some particularly dynamic driven stunt action. These are notoriously difficult shots to animate because of the difficulty in keyframing linked events driven by quick changing forces. In this case, a completely different rig was created. Using the dynamic section of a proprietary software called Zeno, each section of the Hulk was assigned his own mass and rough shape. Each joint



THE SKELETON WAS DRIVING A POLYGONAL PROXY MESH THAT WAS CARRYING EXTRA INFORMATION. AMONG THEM THERE WERE TRANSFORMATIONS IN CHARGE OF DYNAMICALLY DRIVEN MOTION.





WHEN BANNER WAS TRANSFORMING INTO THE HULK AND HIS BODY GROWING, THE SPRING TENSION WAS INCREASING, EVENTUALLY "BREAKING" THEM AND REVEALING THE CLOTH TEAR



was replaced with dynamic constraints and springs. Forces were applied to the model and the resulting animation was eventually baked out and sent back to the animation department for final tweaking and clean up.

The morphing sequences were handled by another custom tool. It was capable of establishing a relation between characters and was giving the Character TD the ability to paint the areas that had to be morphed at each given time. The director could then decide that the right arm, for instance, should start to morph at frame 36 moving from top to bottom, while the leg should start from frame 72 following a radial pattern.



MARVEL WAS HEAVILY INVOLVED IN THE EFFORT TO KEEP OUR CHARACTER TRUTHFUL TO HIS PERSONALITY.

The cloth team had the daunting task of taking care of the ripping cloth. The internal cloth simulator had to be modified to support this kind of event. The clothes were pre-cut by modelers or Character TDs to follow a predetermined pattern, then sewed back with "springs." The springs had a breaking tension point where they were disabling themselves. When Banner was transforming into the Hulk and his body growing, the spring tension was increasing, eventually "breaking" them and revealing the cloth tear.

A lot of time was spent on the look development. Several months went by to create the facial library for the Hulk. Everybody has an idea on how the Hulk looks when mad, but how about when he is sad? Or thoughtful, or maybe even happy?! The modeling team did a lot of research about it, and Marvel was heavily involved in the effort to keep our character truthful to his personality.

At the same time, TDs were working on the look of his skin. A really ingenious method was devised to create the pores. A Maya plug-in was written to generate particles on the surface of his

body. The plug-in was then giving each particle some kind of repulsion field that was pushing them around until they were equidistant and in balance to each other. Then, the particle position was baked in to a texture map and the map used to generate bump. In the end it was the texture the painters could edit and

modify, but because it was particle-driven, it didn't have any stretching artifact typical of projection mapping. An existing ILM skin shader was enhanced with the ability of calculating subsurface scattering. This took care of those ugly CG shadows and gave the skin a bright and organic feeling. The

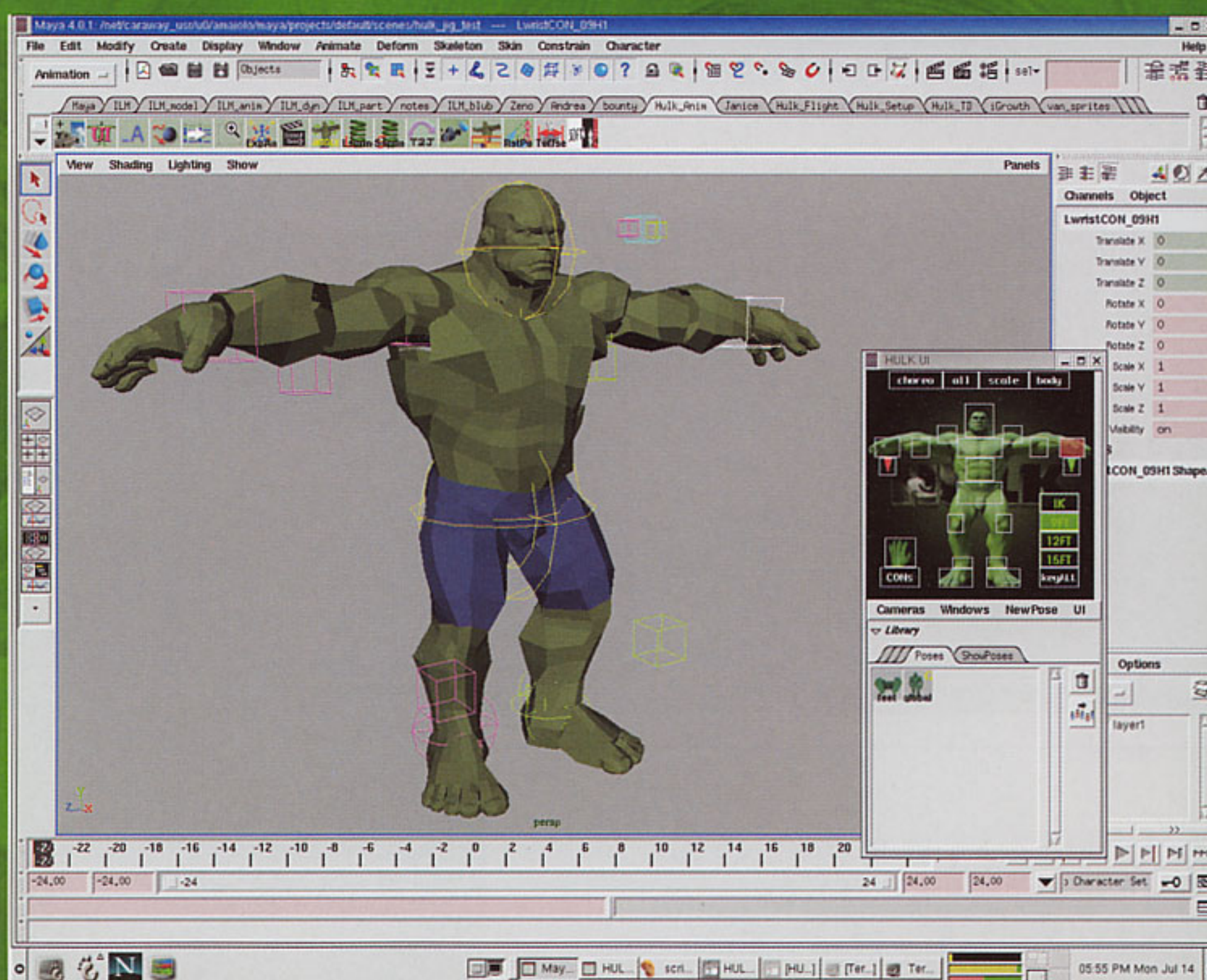


A MAYA PLUG-IN WAS WRITTEN TO GENERATE PARTICLES ON THE SURFACE OF HIS BODY.





SEVERAL MONTHS WENT BY TO CREATE THE FACIAL LIBRARY FOR THE HULK. EVERYBODY HAS AN IDEA ON HOW THE HULK LOOKS WHEN MAD, BUT HOW ABOUT WHEN HE IS SAD?



“...each muscle was modeled in its flex and relaxed states so that the animator or the Character TD could decide where and when the Hulk would flex his muscles.”



rendering was done through Renderman exploiting the now famous technique of ambient occlusion calculated through mental ray.

Many other objects and models had to be created for the show, such as three Hulk dogs that shared a lot of the techniques illustrated above for the Hulk and a bunch of Abrams tanks that went through the Zeno treatment for the dynamic sequences of interaction with the Hulk rage. A couple of helicopters followed the same path of the tanks. Composed of BSplines patches, they were textured with our proprietary painting system and modeled from references and blueprints in Alias Studio. Other elements included a jellyfish and a character that can morph into everything he touches. When he was something inorganic, his look was really driven by shaders, texture mapping and particles. When he transformed into rocks, Zeno dynamics was called again to the task of

keeping everything together, while at the same time preserving physics laws and animation.

Relying heavily on Maya dynamics and on a plethora of custom made fields and forces, TDs spent a lot of time making sure that the environment was reacting to what was going on in the set. Every time a 2-ton Hulk landed after a three-mile jump, they created the parti-

cle simulation that was moving the dust around him and was making small rocks and debris shake. The same sort of effect was created for the helicopter rotors, the tanks etc.

The frames were rendered mainly on our huge pool of Linux machines. All the elements were finally put together with our proprietary compositing tool and filmed out in our own scanners. 🟡







# Real-Time Reflections

BY BOBBY MILLY

In today's changing video game industry, technology is forever evolving. New exporters are written everyday, and existing engines are upgraded daily with new tools and fixed patches. And, technology will continue to progress. This is evident with the quality of gaming, as it is closing the gap to film production.

Many of today's gaming companies are often faced with problems and issues with their exporter, trying to get their assets across multiple platforms effectively; and many have been met with disappointing results. This often occurs when there is no strong programming team to supply the game artists with front-end software support, such as a tools programmer. Basic features such as transparency alpha, UV animation, partitioning of levels (sectorization), or even sorting between geometry surfaces, are simple items (tools) that may never be excisable to the game artists. So what do we do as artists? Well, this article will explain some of the process that the video game industry has

created to achieve a clear visual representation. See image above.

The one thing you must know is that all reflections are not real. If a game console engine had to reflect objects in the world in real-time, the console would just shut down completely. If surfaces are reflecting, a standard route would be using a texture map in the environment tab. A more robust reflection would be shown here.

Figure A is using duplication of geometry in -1 in Y reverse to give an illusion of objects in the world that are reflecting back (items such as light that glows on the floor, walls or character). All are simple tricks achieved by applying Vertex

Paint Tool in the color tab under the edit polygon tab. By Vertex Painting (lighting), you can create a number of lighting schemes to give a sense of mood or to enrich it.

See images: *figure A* , *figure B1 & B2*

Most exporters do not support lights from a 3D package unless you are using a Level Editor of some sort (such as Doom, Quake or Unreal). For your models or assets to display with an exporter that is not a Level Editor, you would have to vertex paint your lights into your model (almost like baking them into your model/BSP). Through an exporter, not all engines support true lights from Maya or any other 3D packages.



Figure A



Figure B1

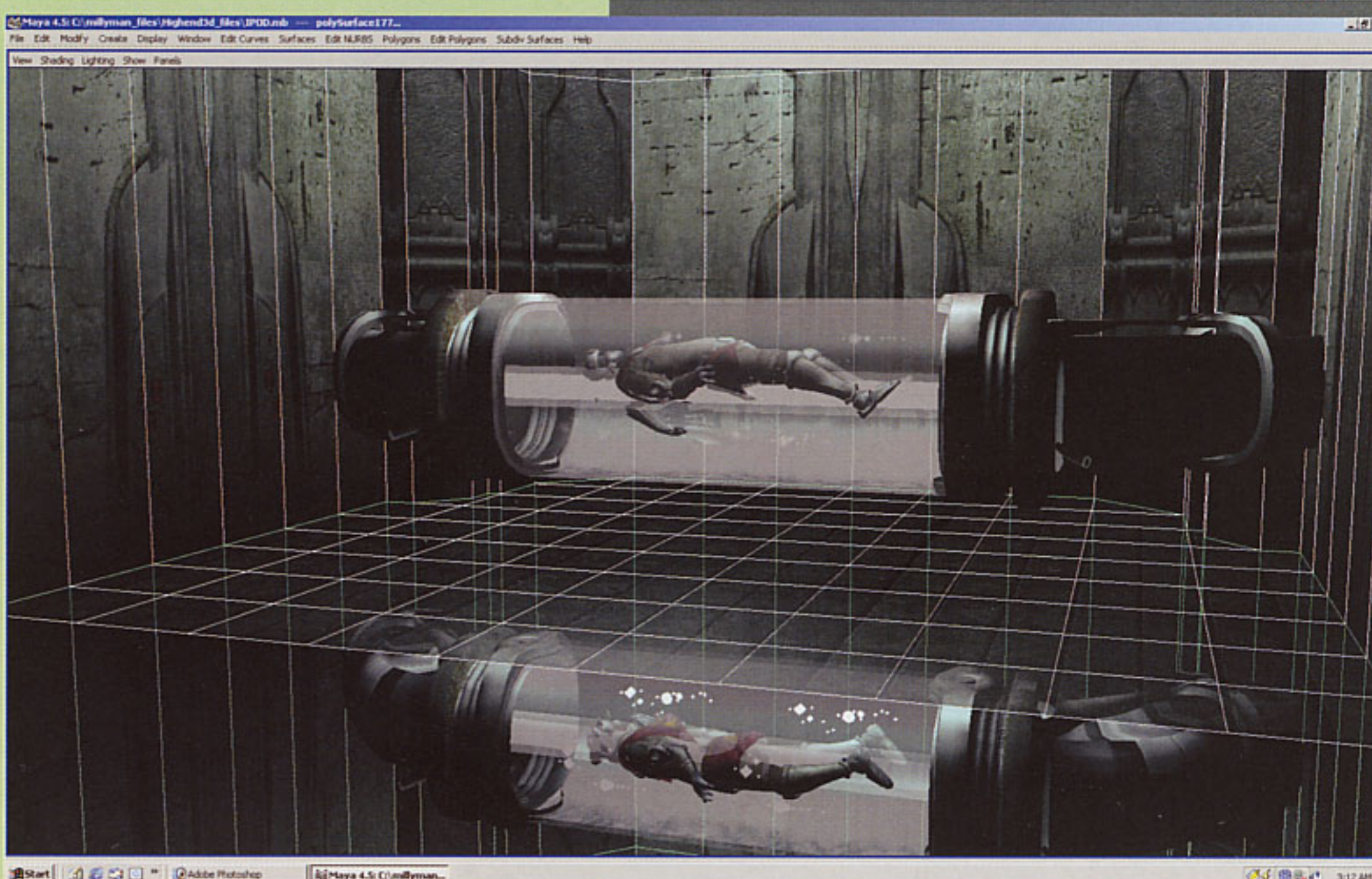


Figure B2

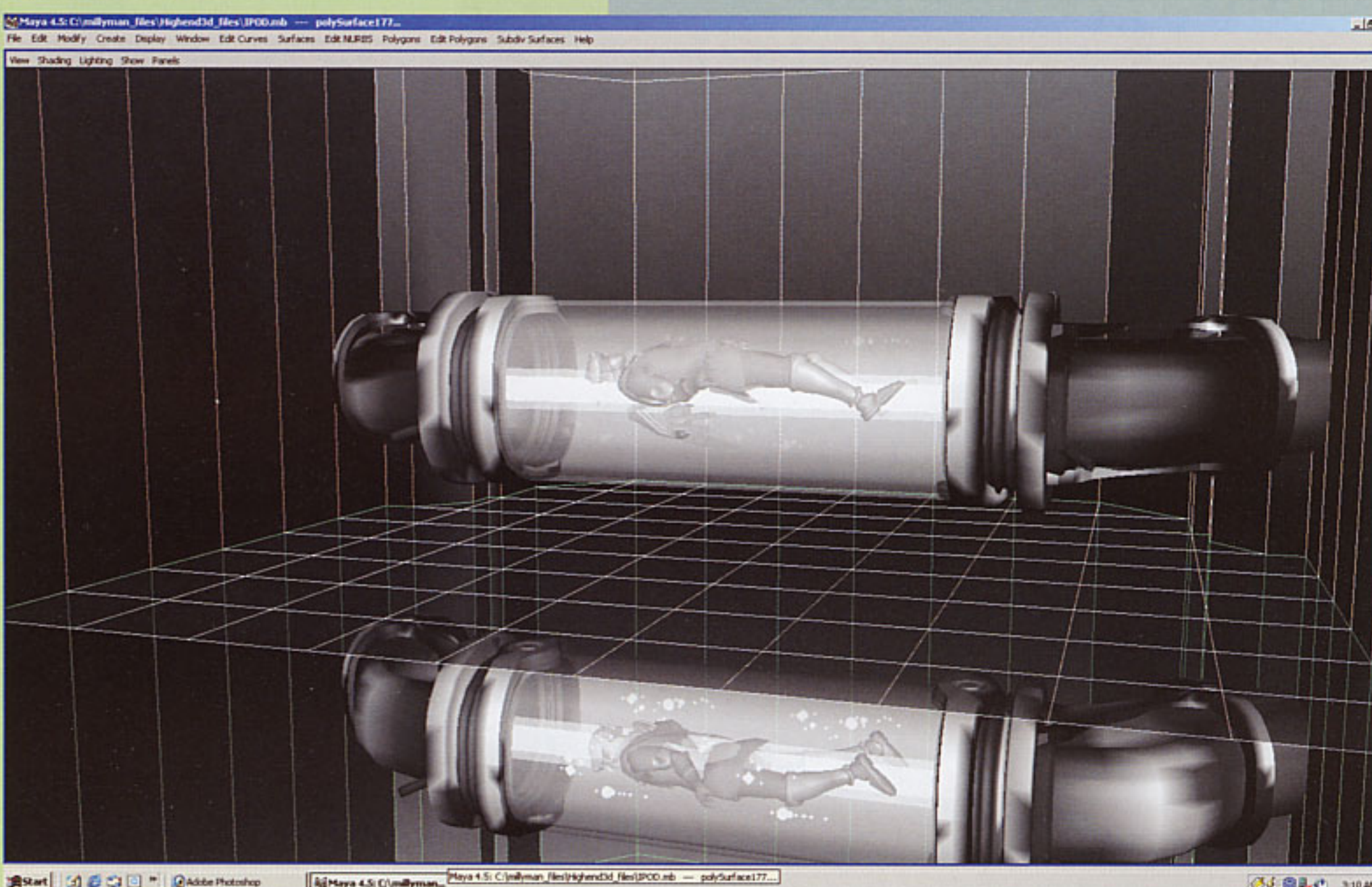






Figure C1

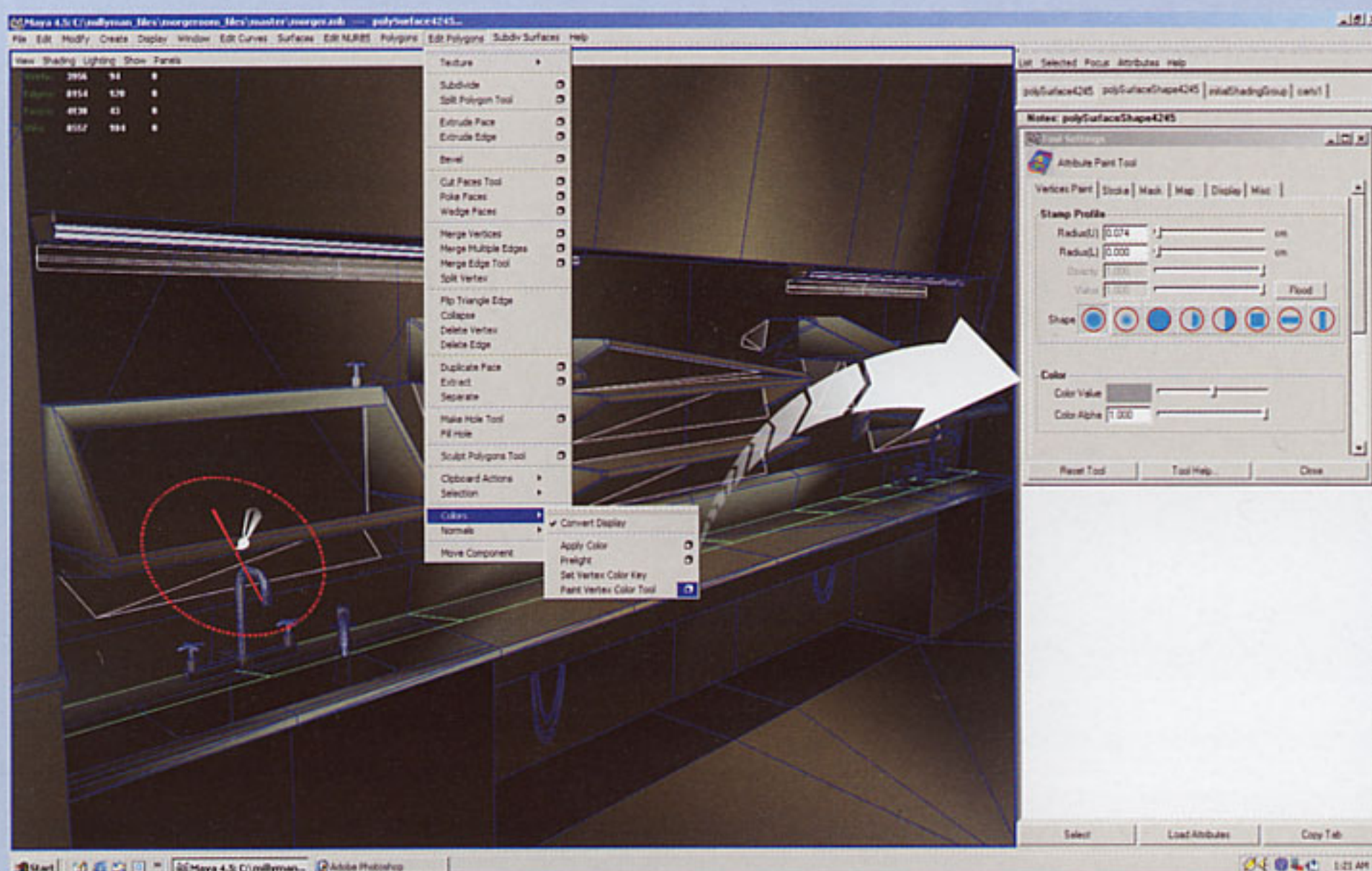


Figure C2

See images: Figure C1 & C2

The images shown here in Figure C1 & C2 are good examples of geometry usage for representation. The fluorescent light on the sides has a reflection below it which is built with a geometry surface. Using geometry instead of texture maps may increase game performance. Please keep in mind that you are limited in terms of texture space, especially on the Playstation 2, which has only 4 MB of space assigned just for textures. So maybe the texture solution may prove to be costly more than helpful. For a majority of the time, you may use simple mirrored geometry. For a richer effect, you could have LODs (levels of detail) work for you.

See image: Figure D

In Figure D, we can see that everything is reflecting back (items such as the walls and the encased character). Vertex painting was also used for the light beam passing through the character model and the bubbles within the encasement, all to provide the illusion of being encased in liquid. Also, the floor is using an alpha value of 0.6 with an RGB color value of 0.8 per vertex point.



"...the floor is using  
an alpha value of 0.6  
with an RGB color value of  
0.8 per vertex point."

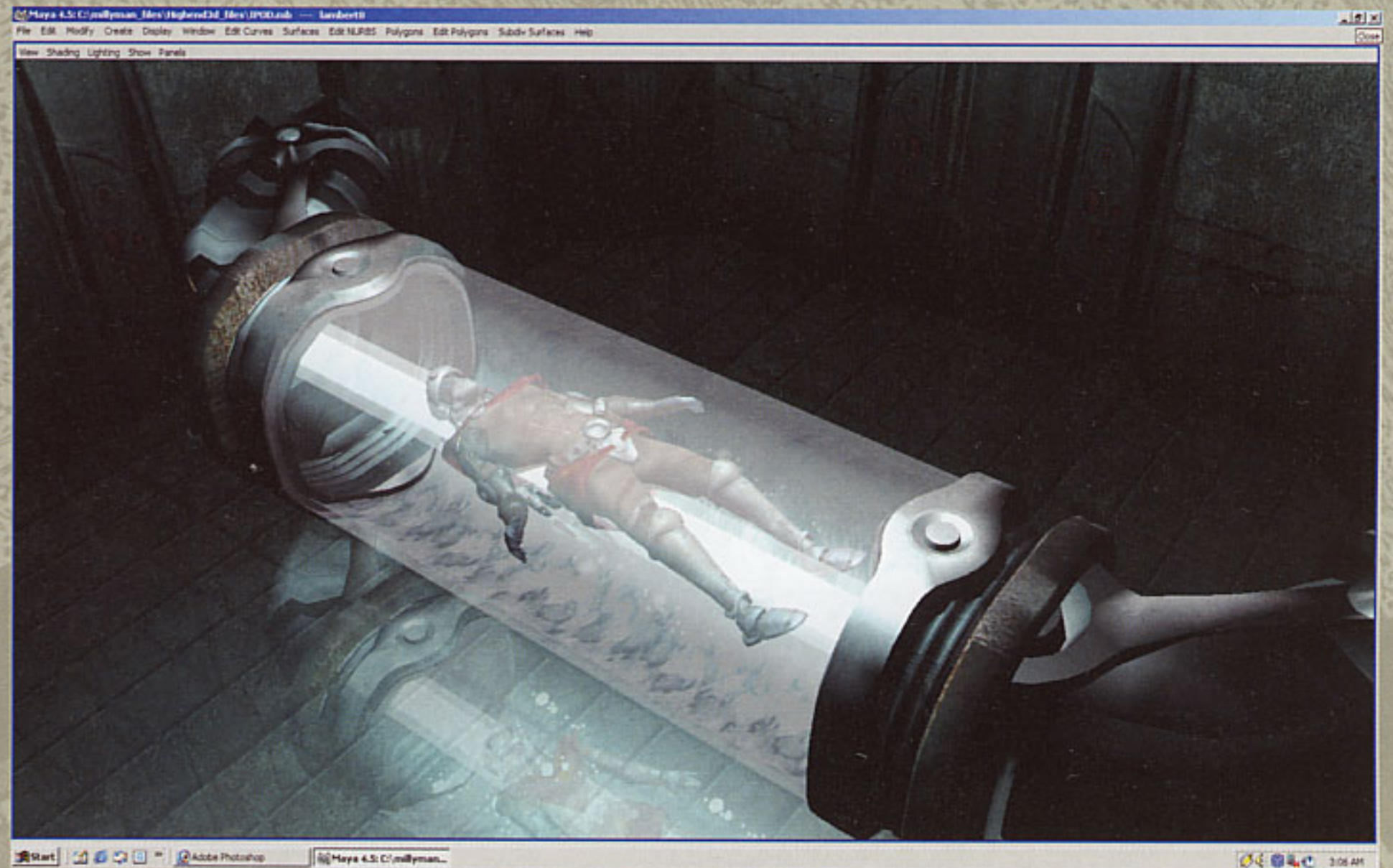


Figure D

All assets were exported out using a 3rd party exporter. There's not an industry standard, yet, though there are many different engines out on the market. If you've got the cash you can buy an existing engine, find someone to code it, and depending on how advanced the engine is already, you could always improve on it. The basic out of the box exporter/engine is just that — basic. Many companies still have to code on top of existing engine code. I'm placing my bets on an

engine that comes supplied with a level editor and event scripting capability.

And just like the movie industry, certain shots are altered in order to present a more clear vision of what the director had in mind. Cheating shots, rather it be film or video games, is always the better route in terms of time and man power. In the future, there won't be a gap between film and video games. In the present, certain effects will always be cheated to create a

more epic vision.

See image: Figure E

Figure E shows the Paint Vertex Tool menu, which reflects that of an Artisan menu. So the next time your gaming solution is waiting to hear from a programmer, try vertex painting your lights, shadows and reflections. 🟡

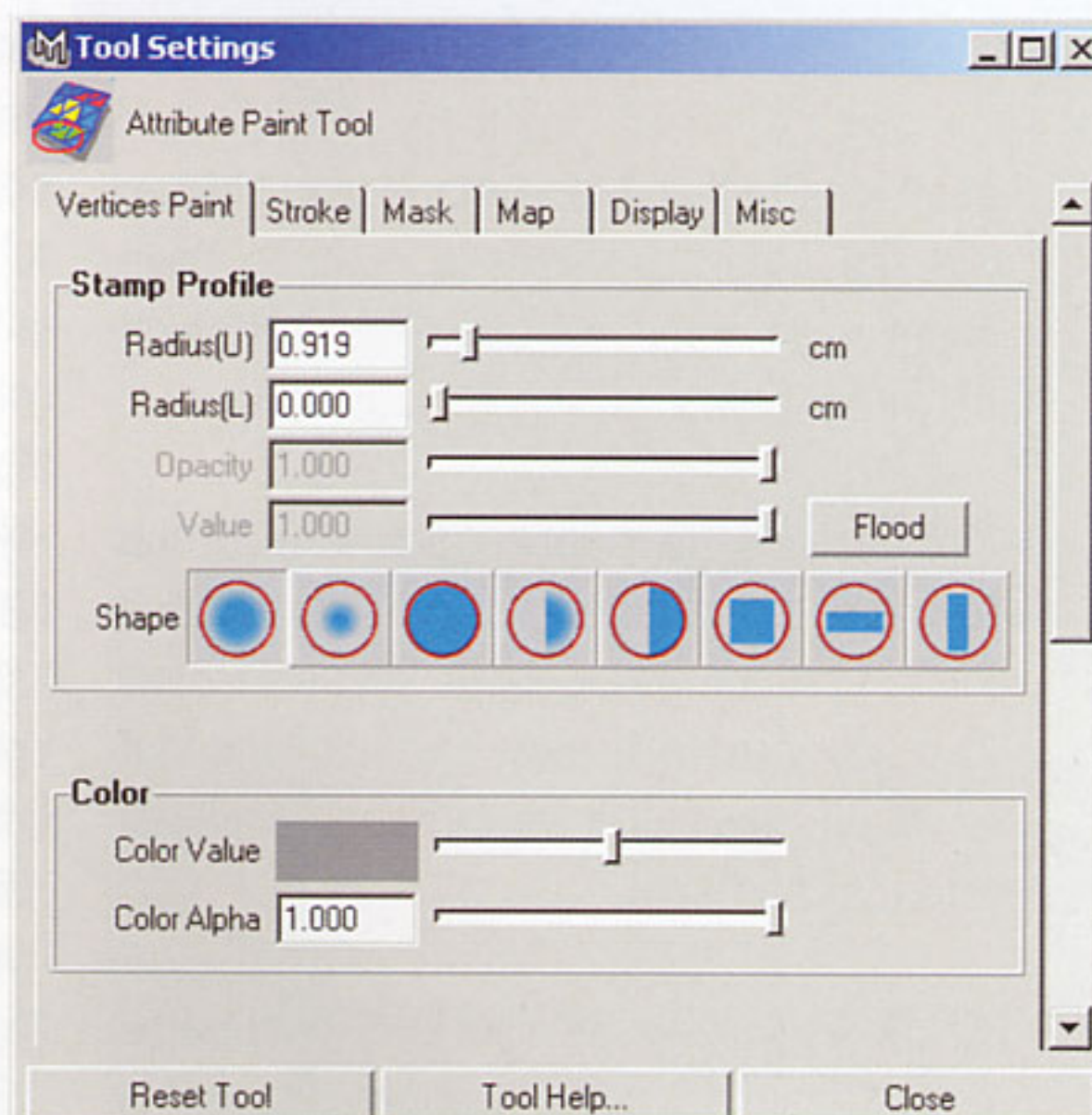


Figure E



# Production Methods

## Using Passes to speed up your work

BY ED HARRISS AND BERNARD LABEL

Most people think that passes are just for splitting your scene into individual rendered elements, (Shadows, Highlights, Ambient, Diffuse, Reflections, Mattes, etc...) but they can be used for so much more. For example: Say you are working on a scene that has a lot of geometry in it. Playback and interaction is extremely slow because the scene is so heavy. In the past, animators may have been given a separate scene, with bounding boxes representing the geometry. This works fine, but in the end all the animation will have to be translated back into the original scene. Not to mention, animation can be difficult when you can't see the objects in your environment. Wouldn't it be easier if the animator could work on the original scene? How is this possible? It can be done with the pass system. It's actually very simple and once you've done it a few times you'll be amazed at how easy it is and wonder how you ever got along without this technique.

1. Create two spheres. Leave one of them with the default settings. Change both the U and V subdivisions on the other sphere to 100. Now you have one sphere that is 64 faces and another that is 10,000 faces.

2. Create a new pass by choosing Pass>Edit>New Pass>Empty. (SEE *edit newpass empty*) Change the name of that pass to Low-res. Create another empty pass and call it High-res.

You now have three passes listed in your pass dropdown. (*pass dropdown*) Default\_Pass, High-res and Low-res. It is a

good idea to leave the Default\_Pass alone. But the other two passes need to have some work done on them, as switching between them currently does nothing.

3. Switch to the High-res pass. Select the low-res sphere and choose Pass>Partition>New. (*edit partition new*) This will create a new partition and put anything that you have currently selected into that partition.

The property page for your new partition will pop up. (*view visibility*) Change its view visibility and render visibility to Hide

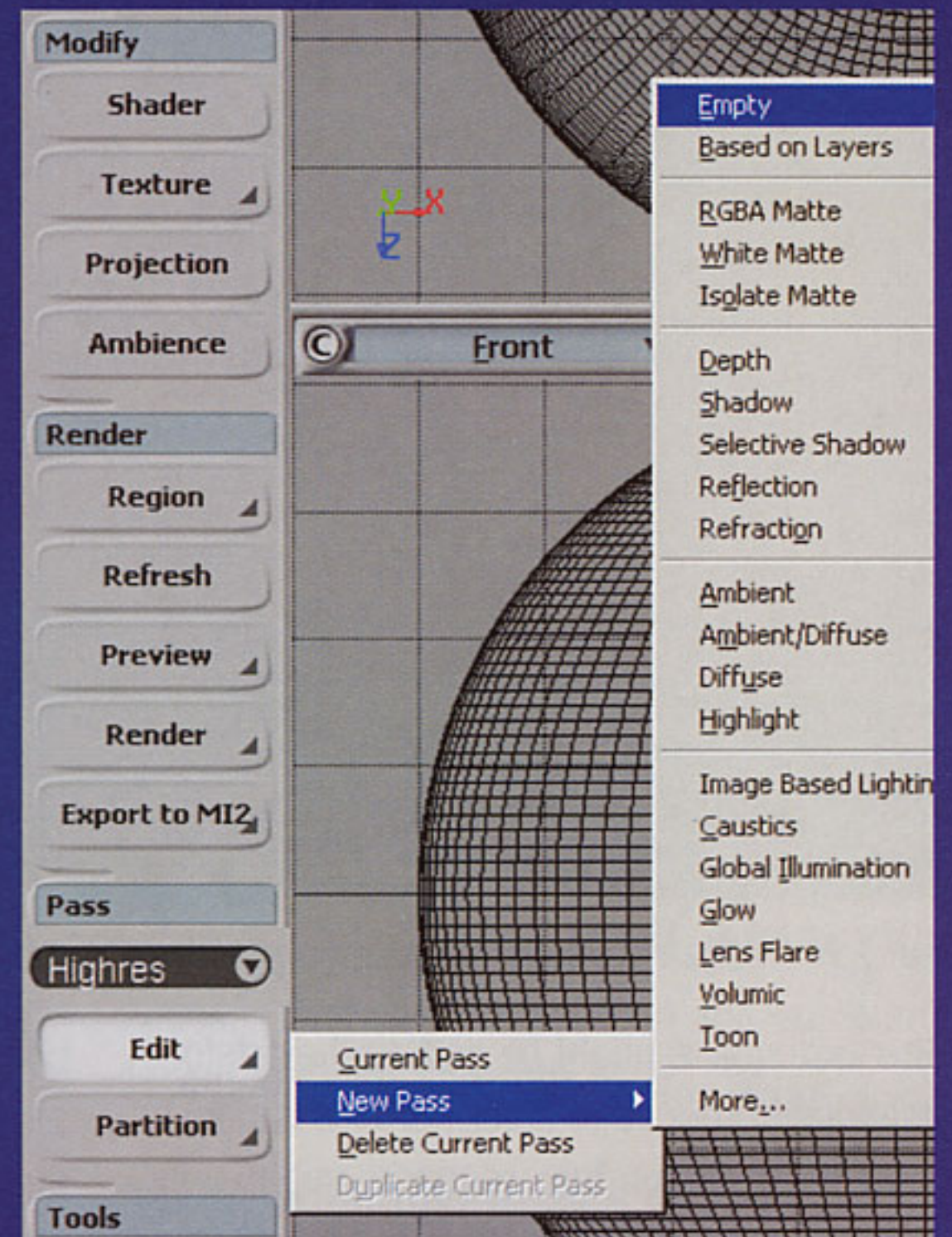
Members. This will force anything in that partition to hide in both the viewports and rendered images whenever the High-res pass is active.

4. Switch to the Low-res pass. Select the High-res sphere and choose Pass>Partition>New. Change its view visibility and render visibility to Hide Members.

**Tip:** Say you have a partition that contains hundreds of objects in one pass. You want a partition just like it in another pass, but don't want to have to set up the partition manually. Simply dragging the partition from one pass to another will create a duplicate with the same attributes.

Try switching between the High-res and the Low-res pass. Notice how the High-res sphere only appears when the High-res pass is active and the Low-res sphere only appears when the Low-res pass is active. This may not seem like much of an achievement when your scene only has two objects in it. But imagine if you had a scene with thousands of objects in it. For example: A city scene with hundreds of buildings, cars and people. In a perfect world, large scenes like this are





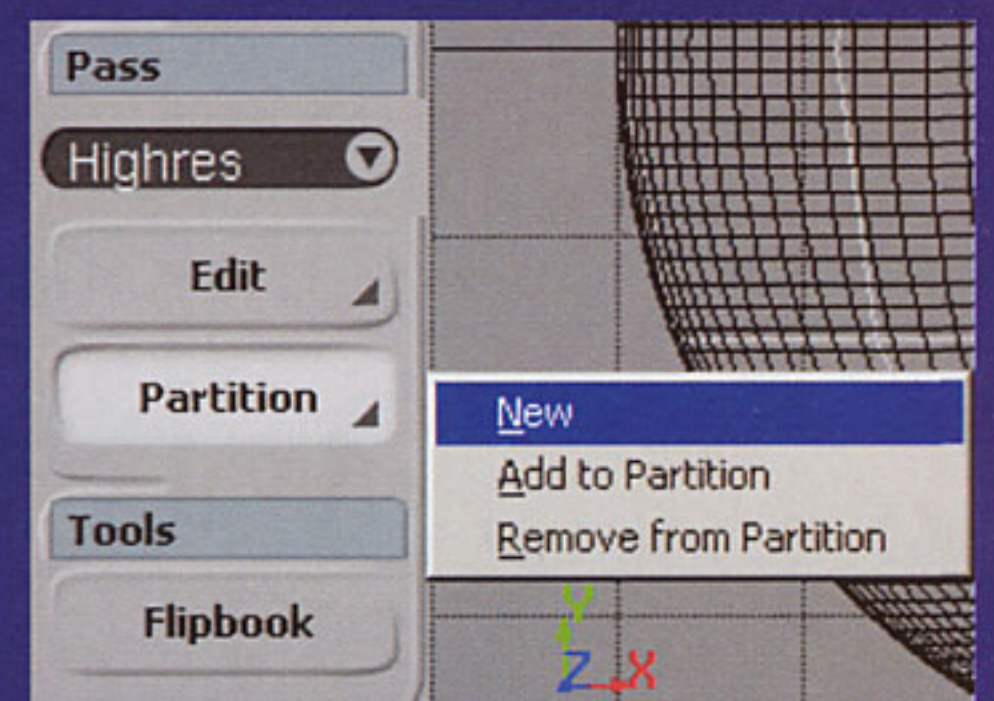
edit newpass empty

made up of many different pieces and are assembled into a final scene or composited together into a single image. But there are times when huge scenes are unavoidable. In scenes like that it would be very difficult for an animator to perform simple tasks due to the slow screen redraw rates. Things like animating the camera, moving geometry from one place to another or correcting the placement of textures would be painfully slow. Even quick test renders can take ages to finish.

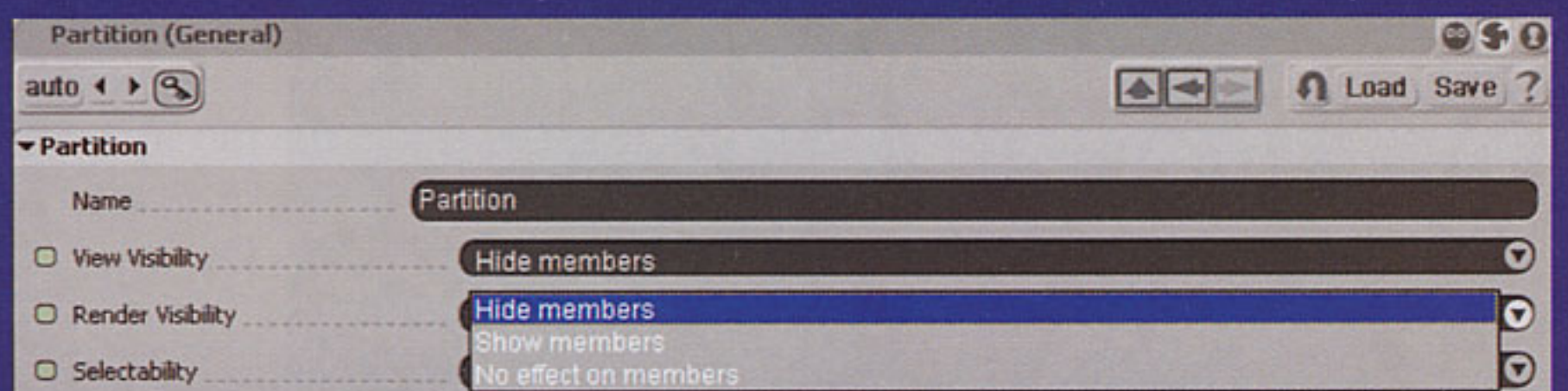
Have a look at *city render tilted* or *city render top* OR *city render side*, *citywire-frame*. Imagine that you have to animate these buildings shaking back and forth for an earthquake scene. At the same time, the camera needs to come flying out of the sky. It would eventually make its way to street level, where it will focus on a specific building. (*city render*) It will have to fly through the intricately detailed buildings on a very specific path. Setting your scene to display in box mode, just won't do. (*city box mode*) Not only will it be difficult to see if your camera is flying through walls, but it will be hard to distinguish between objects because of all the overlapping geometry. In this example, the buildings that are being "shaken" by



pass dropdown

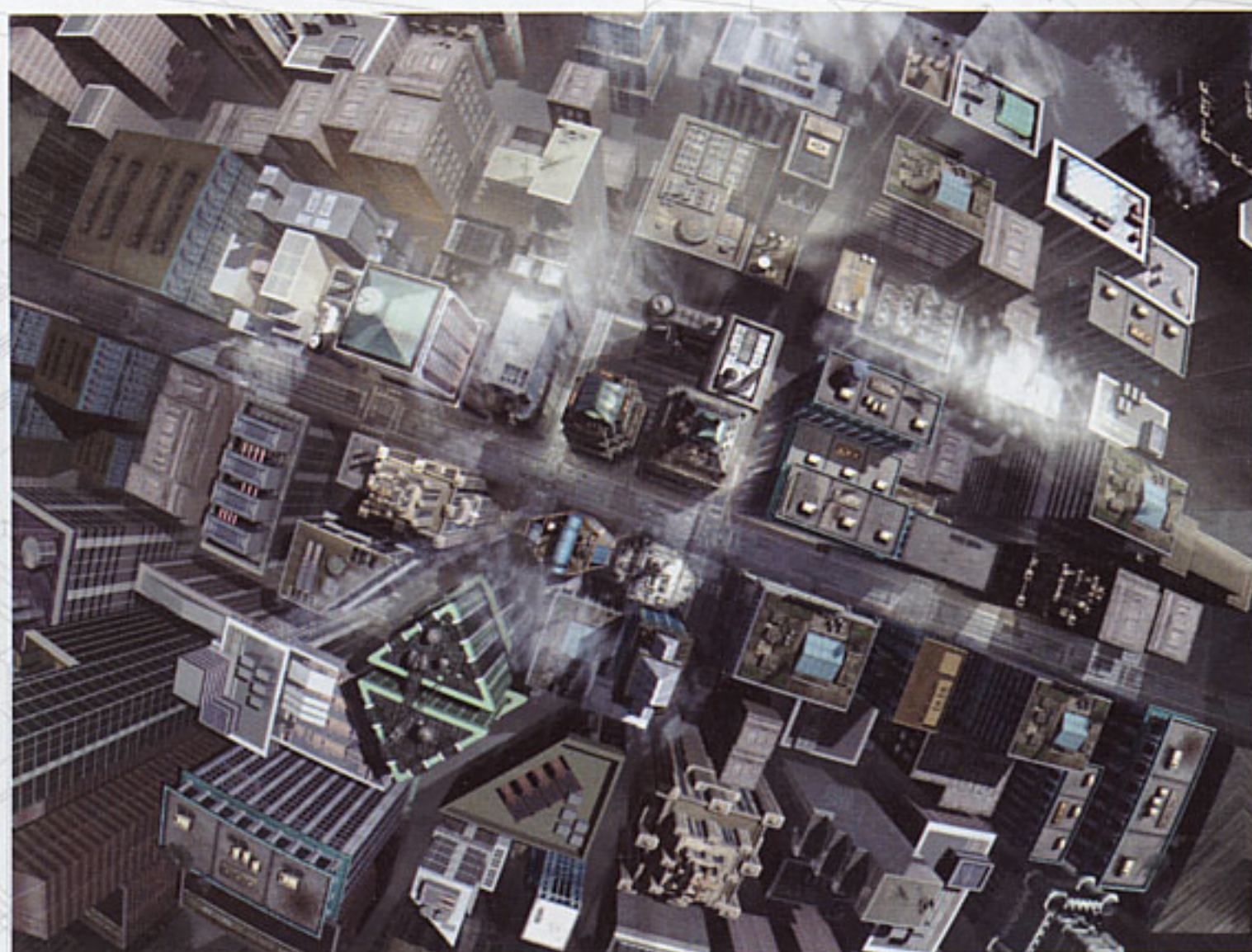


edit partition new



view visibility





city render tilted

the earthquake might be getting their deformations from a cage deformer or a lattice. A low-res pass for this scenario might only have the deformer visible, while all the buildings are hidden. There will be another low-res pass for the camera animator. This one hides all the buildings except those that the camera will be flying through. (*city street box mode*) As the project progresses, other passes performing various other functions might be set up. Setting up these low-res passes may take a little time up front, but on a scene this size they will save you a lot of time. When you want to switch between very specific display types, all you have to do is

change passes. Before this sort of system existed, simple tasks like this might require cumbersome scripts or tedious changes to each model's display mode.

## This opens up other options as well.

Imagine working on a camera animation and getting it exactly the way the client wants it. At the last moment, they ask you to change it. You suspect that the change is a bad idea, but you move forward with it anyway. You can duplicate your pass, the camera rig and its animation. Then put the old camera into a partition that hides it. This allows you to change the animation



city render top

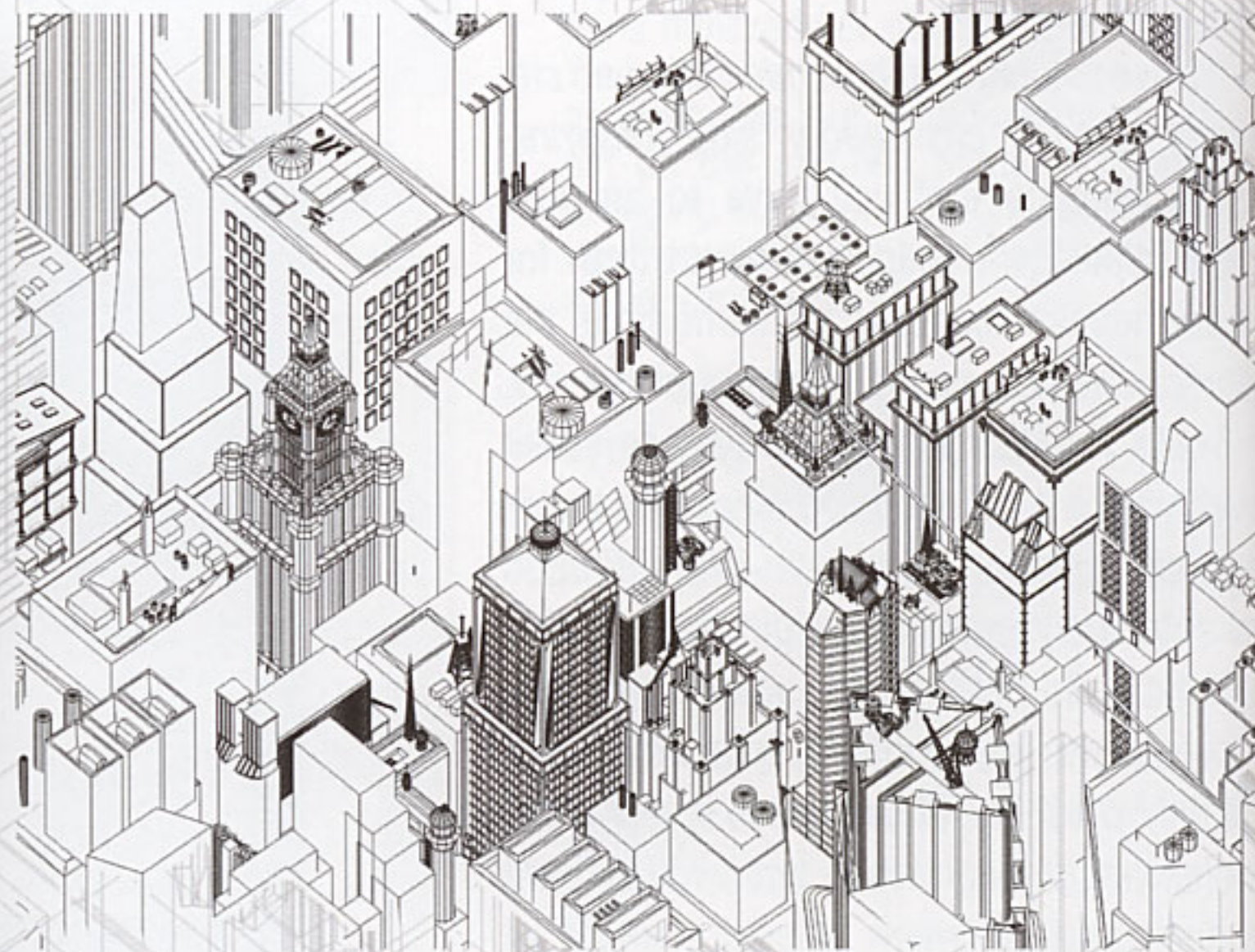
on your new camera without affecting the old one. This set up also makes it easy to compare the two camera animations simply by flipping back and forth between the two passes. This can be a real lifesaver when you've got the client sitting right behind you. This method makes it very easy for them to see just how bad their new camera animation idea is. Once they've realized this, you simply switch back to the old one and continue working.

## Using passes to show a "work-in-progress"

Not only are passes good for organizing your



city render side



city wireframe



scene, but they make a powerful material management tool as well. Passes can be used to preserve your work in different stages without the need to save multiple versions of your scene. If you apply textures and materials to your objects via partitions, you can easily and non-destructively alter those materials.

### Here is how it works.

1. Create a sphere and create a new pass by choosing Pass>Edit>New Pass>Empty.
2. With your sphere still selected, click Pass>Partition>New. This will create a partition that contains your sphere.
3. Open up an explorer and set its scope to Passes. Select the partition that you've just created and assign a Phong material to it. (Get>Material>Phong) Make it purple, just so that it will stand out.
4. Your sphere will need a texture, but before you assign one to it you'll need to give it a texture support. To select the sphere, choose Get>Property>Texture Projection>Create New Projection. Select UV from the Projection Type drop-down and Spherical from the Default Projection drop-down. Type in a unique name into the Projection Name section of the dialog box. Giving your projection a name is not necessary, but can save you time in the long run.
5. Select the partition and assign it a texture by clicking Get>Texture>Image. From the drop-down list, choose your texture projection.

You've just set up a basic partition-controlled material. Any object you drop into this partition will inherit the material and the textures that you assigned to the partition. This makes managing the same material across many different objects very easy. As you drag objects from one partition to another, their materials will change depending on what has been assigned to that partition. But if you put them back into the background partition or a partition that

does not have a material assigned to it, they will revert to their normal state as if nothing had ever happened to them, even if their normal state included many different textures and materials. You can do so many cool things with a system like this — but this is just the beginning of what passes can do for you.

As we all know, when doing work for a client, they always want to change things and “experiment” with whatever you are working on. They often go back and forth between hundreds of ideas before settling on the one you showed them first. How many times have we all heard the client say, “Can we go back to the way it was two hours ago?” Because of this, many people end up saving countless versions of their scene as they work with the client. This way they can “go back to the way it was two hours ago” without hitting the undo key 50 thousand times. This means that previewing and adjusting older versions requires finding and loading up new scenes, not to mention trying to keep them all sorted in your head so that you don't get lost in your own project. How would you like it if all those scenes you saved were all contained in the same scene? You could pick them from a list and be able to instantly render an example. With passes and a properly set up scene, this is entirely possible.

1. Select your pass and duplicate it by pressing Ctrl-D.
2. In your new pass, select the partition that you assigned the material to. Click Modify>Shader. Adjust the materials/textures on that partition until you reach a point where you would have saved another version of the scene had you been working with a client.
3. Draw a render region and switch back and forth between the two passes. Notice that the original pass never changed, while the new pass

has all the adjustments that you've made.

4. Select the new pass and duplicate it by pressing Ctrl-D. Return to step 2 and tweak until your client is satisfied that they have wasted enough of your time.

Using this method, you can flip back and forth between the passes to see “before” “in-between” and “after” versions. It is like having all of your scenes and revisions loaded at the same time. Not only does this save you a tremendous amount of time, but it can improve your work, as you can take the aspects of the passes that you like the best and combine them into one final look.

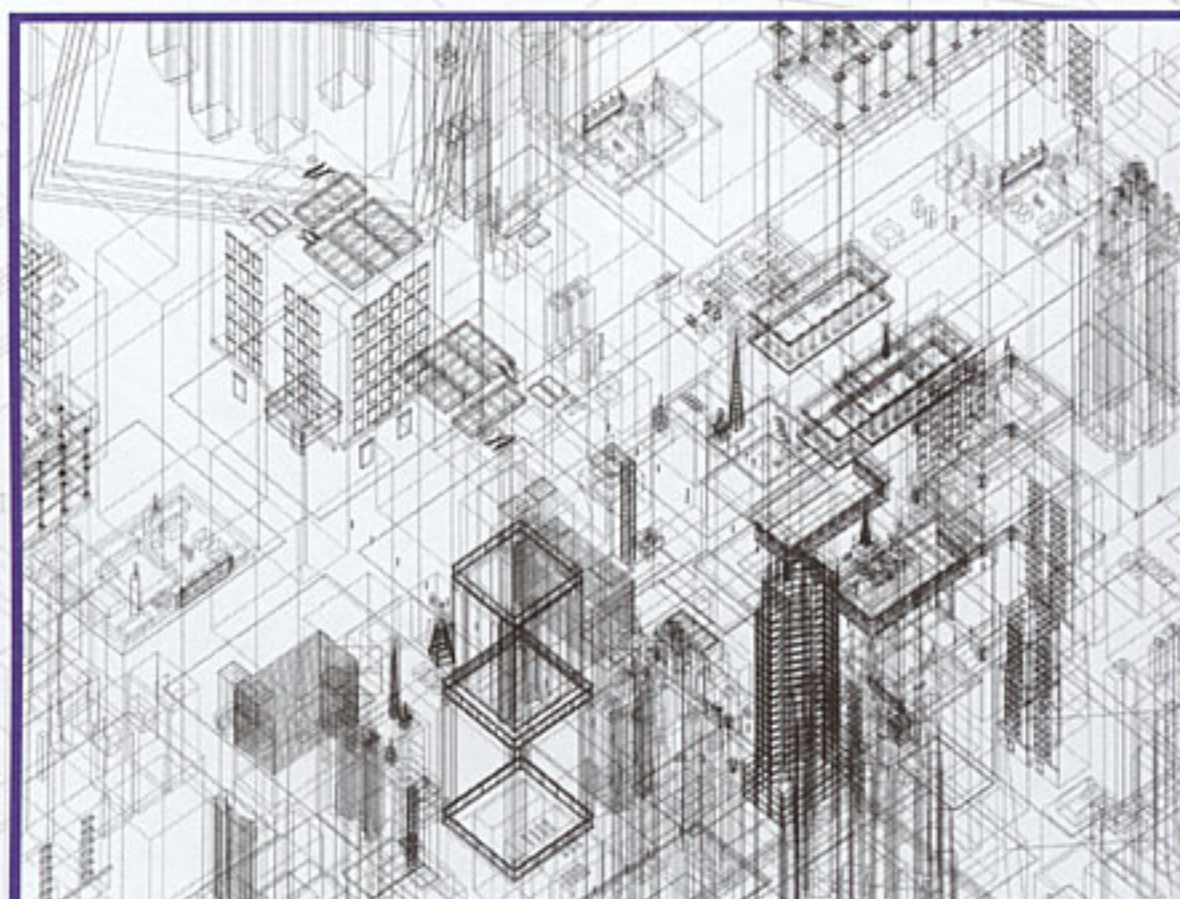
### Naming conventions

When working on any production, it is always a good habit to have rigid standards. On medium to large productions, it becomes absolutely vital. These standards are meant to keep all aspects of the production uniform. Rarely will you be the only person to work on a job. In fact, most of the time you will be working with a team of people. Without standards, working on scenes that other people have created would be very time consuming, as it would be impossible to find anything quickly. Also, when it comes time to work on an older scene, proper standards will make your life easier.

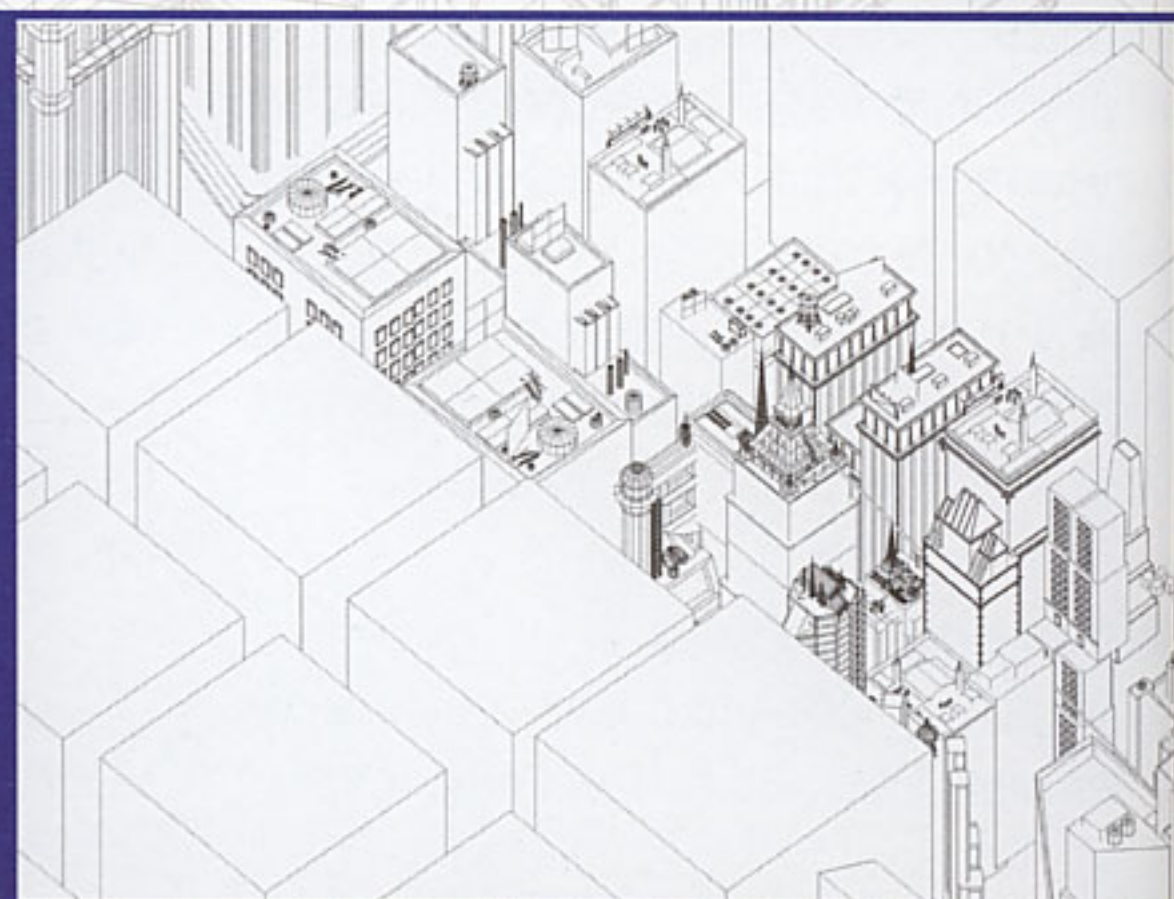
city render







<city box mode  
city street box mode>



There is another major benefit of having rigid standards: It makes the developers (scripting gurus) life easier. When conventions are carefully observed, scripts meant to work on explicit items in a scene will run well. (Even though a good scriptwriter will write his scripts to be as non-explicit as possible). Plus, it will add to your value as an employee.

Among these standards are naming conventions. It is vital that you have a consistent naming convention for everything, including:

- Client/Projects
- Elements
- Shots
- Project folders
- Scenes
- Models/reference models

- Hierarchies
- Groups
- Passes/partitions
- Objects
- Lights
- Materials
- Maps
- Shaders
- Rendering frames
- Footage

The important thing about naming conventions is that they have to be consistent, logical, and they must be defined early in the production. Ideally, you would define them even before creating anything for the project. If the naming conventions are followed closely, then you should be able to work on any scene at any time even if it was not created by you.

## Clients/Projects

From now on I will use the term ID. What I call ID is not an identity card, but codes that you can use to name your files and the elements present in those files. IDs are basically made of two things: the client/project identification and the project identification.

**Client:** Assign a code made of two or three characters (letters or numbers) to each of your clients. When you are dealing with many clients on the same project, then it is better to assign an ID only for the project. This ID will be used to separate each client or project on your drives.

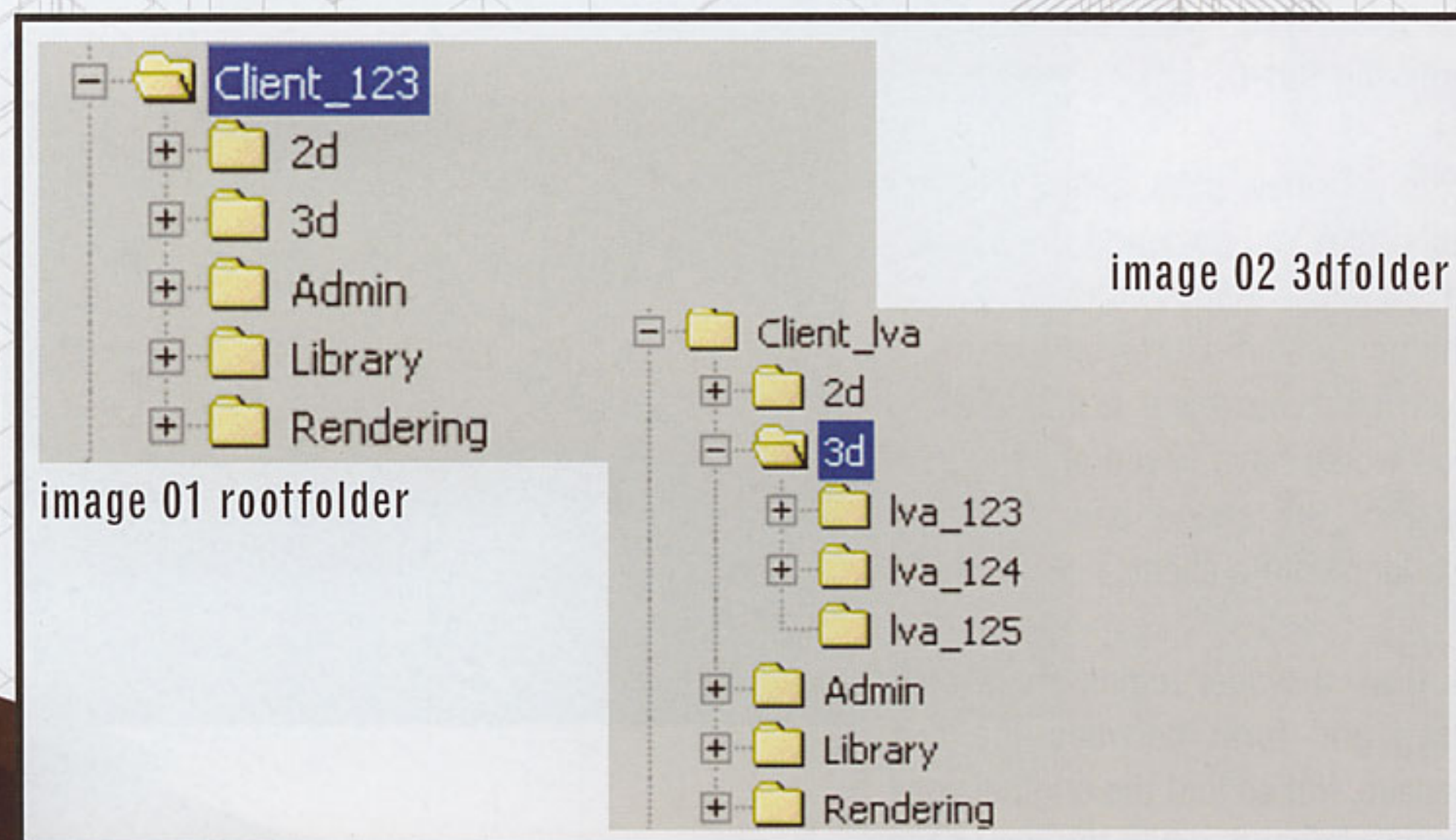
**Project:** This is the actual project you're working on. It may be put into a clients folder, or be part of a bigger project. For example, if you're working on a TV show, the entire show is the big project, while each show is one project.

An ID for a project would look like this: Iva\_123\_, where "Iva" is the client ID and "123" is the project ID.

The client and project IDs will be used throughout the project. For example: project folder names, scene names, rendering folders names, rendering frames names and so on.

## Element IDs

Element IDs work a bit like the project IDs. While project IDs identify the client and the project itself, element IDs identify elements that will be used throughout the entire project but are not necessarily bound to a specific part of the project. For instance,





characters, props and environments are things that will use an element ID.

An element ID is generally more descriptive than a project ID. While a project ID is a combination of two sets of characters, an element ID will consist of many parts: the type of element, followed by its code number, followed by its version, followed by its name. Here are examples:

env0034\_01\_bobcar  
cha0012\_01\_nancy  
cha0012\_02\_nancyswimsuit  
pr0149\_01\_bobsunglasses

### Shot IDs

Each shot of a given project should receive its own ID. This ID consists of the client ID, followed by the project ID, followed by "sc" and the shot number. For instance, shot 24 of project 123 of client Iva would be named Iva\_123\_sc024. Note that you should always use leading zeros, like in the previous example.

### Project folders

Ideally you'd have a dedicated folder for the entire project. In larger studios, you should try to have the entire project in a single drive on the network. (Or at least set up in a way that it appears to the end user that it is located on a single drive.) If it is a shared folder on a drive, make sure the same drive letter is assigned on every network mem-

ber's machine. Having everything at the same place, under the same letter, will allow anyone on the network to access the data without any special consideration.

On this root folder, you should have dedicated folders for: the 2D concepts and storyboards, administration files, rendering output (frames), a centralized library that contains all the textures, environments, characters, props, shaders and materials and so on. Finally, have a dedicated "3D" folder, where the work files will be stored.

(image 01 rootfolder)

In the 3D folder, you should have a folder for each spot (if it's advertising), show (if it's a TV series), or shot (if it's a feature film). (image 02 3dfolder)

Now, in each folder for spot/show/shot, you should have three folders: pre-production (modeling, texturing, pre-lighting, rigging and so on...), production (layout, animation, lighting and so on), and post-production (compositing, fx and so on).

(image 03 showfolder, image 04 stagefolders)

Finally, you should get something like this: image 05 fulltree.

Basically each shot in the production folder will consist of an XSI project folder,

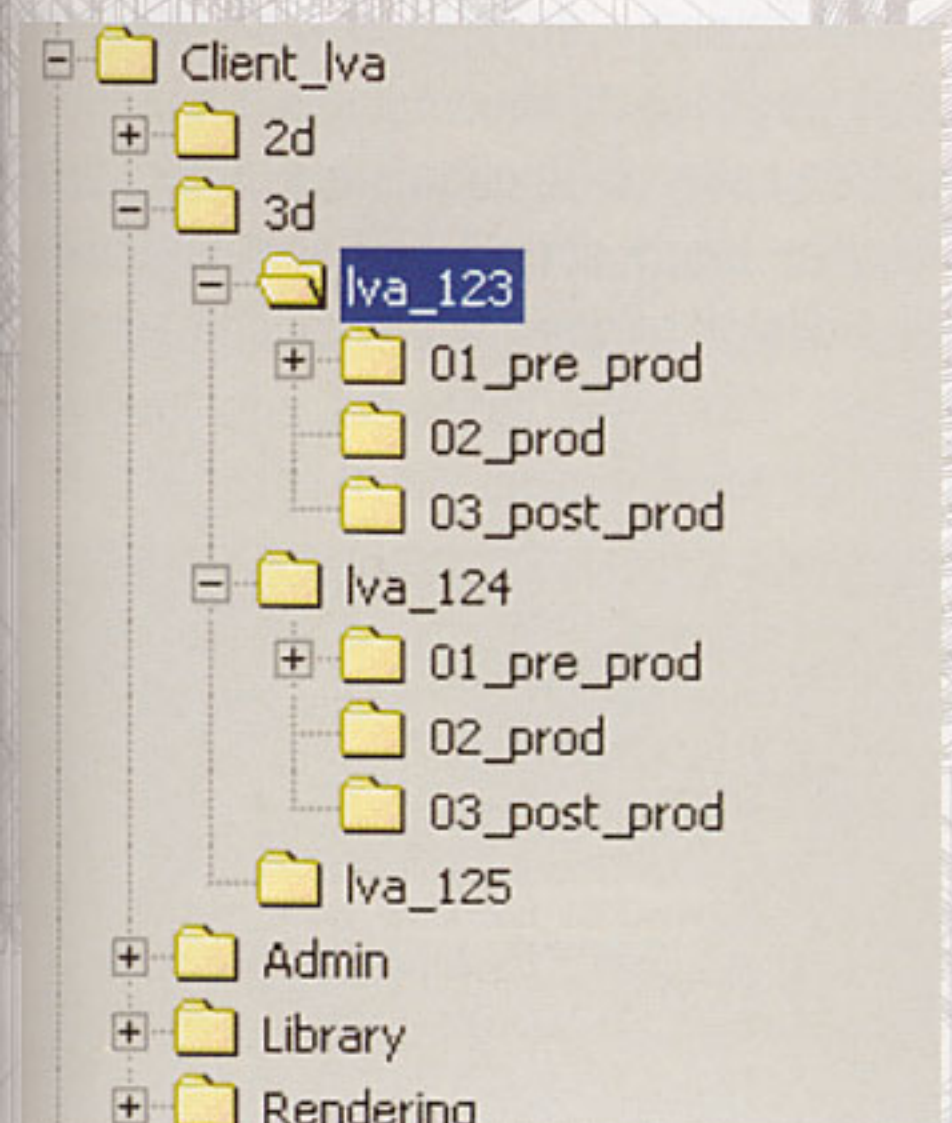


image 03 showfolder

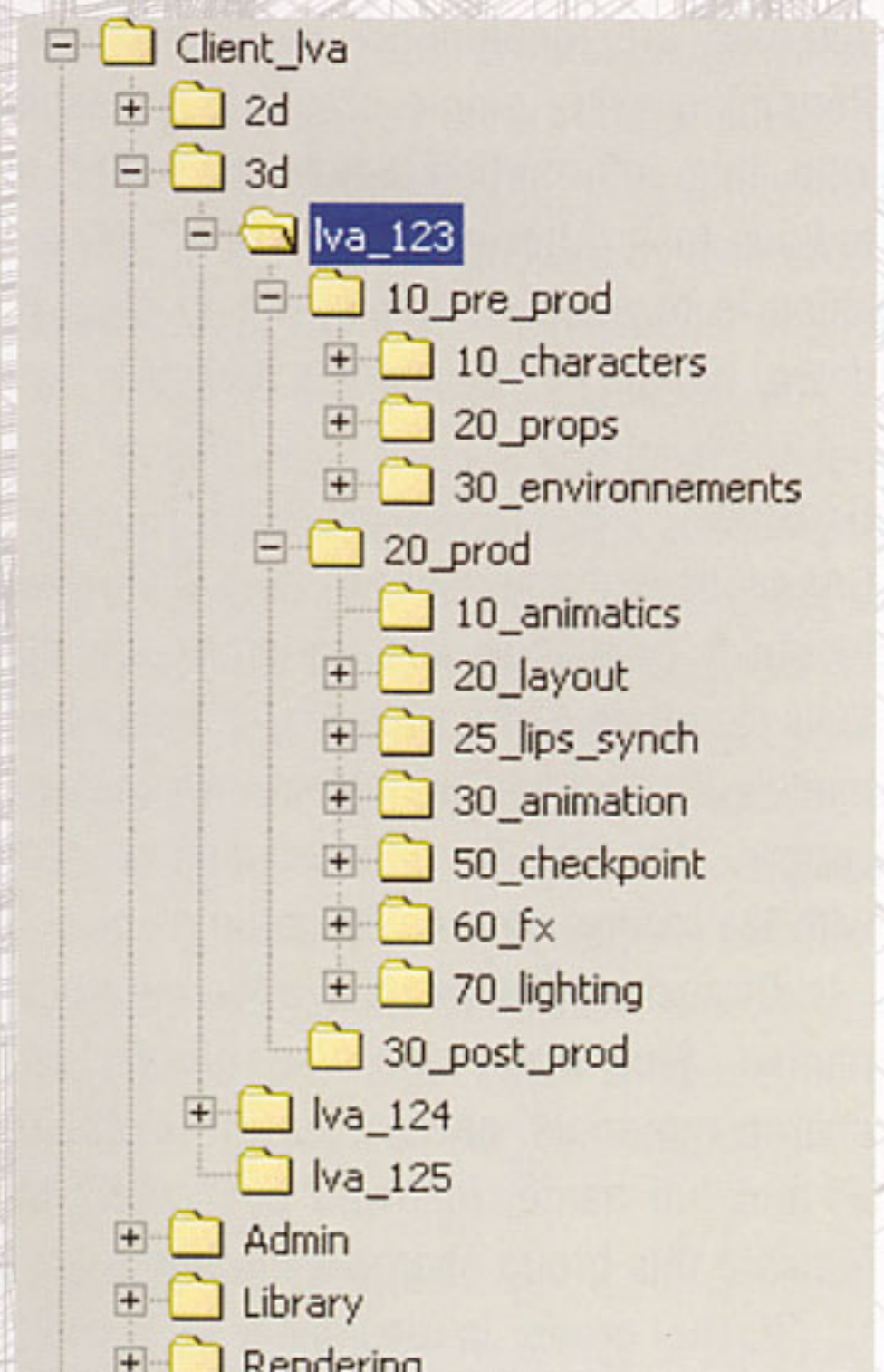


image 04 stagefolders

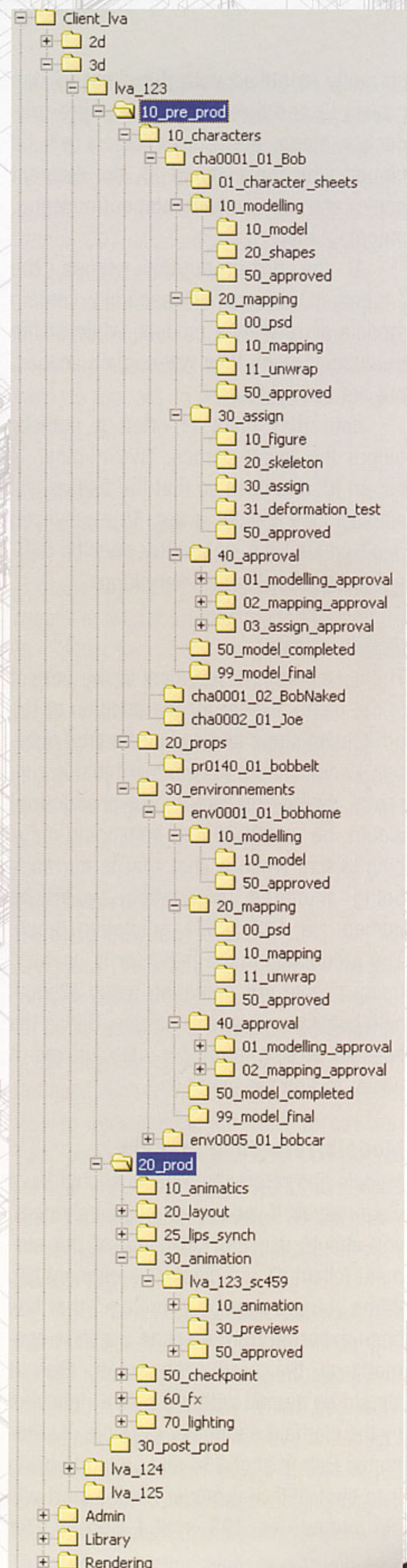


image 05 fulltree



correctly identified with the client ID, the project ID and the shot ID. In the pre-production stage, the project folders will be named with the element ID, the element name and the current stage (modeling, mapping, etc).

At the pre-production stages, the “Approved” folders will generally contain models or reference models, while at the production stage they will contain scenes, but not project folders.

Note that there is often a number before the folder names. This number is not an ID. It is used so that the folders will display in a logical order. This way you don’t waste time looking for specific folders once the list grows very large.

## Scene name

The scene name is identical to the project folder name, except for the addition of the production stage and an incremental number. A scene for an environment that is currently being textured (pre-production) would be `env0026_01_bobshack_mapping03.scn`, while a shot that is currently being animated (production) would be named `lva_123_sc571_animation13.scn`. The production stage indicator is useful if many people are rendering many scenes, as it helps to manage rendering. Giving the scenes version numbers also makes it easier to find the latest one.

## Models/reference models

Models and reference models should have clear names. If you are naming an element, you should use the full name of the element (client ID + project ID + element ID). When you are naming something that has been exported from a scene which is currently at the production stage, then it should be named with the shot ID followed by the element name. For example, if I animated Bob in shot 354 and I want to bring it to shot 359, the model I will export will be named `lva_123_sc354_Bob`. If, for

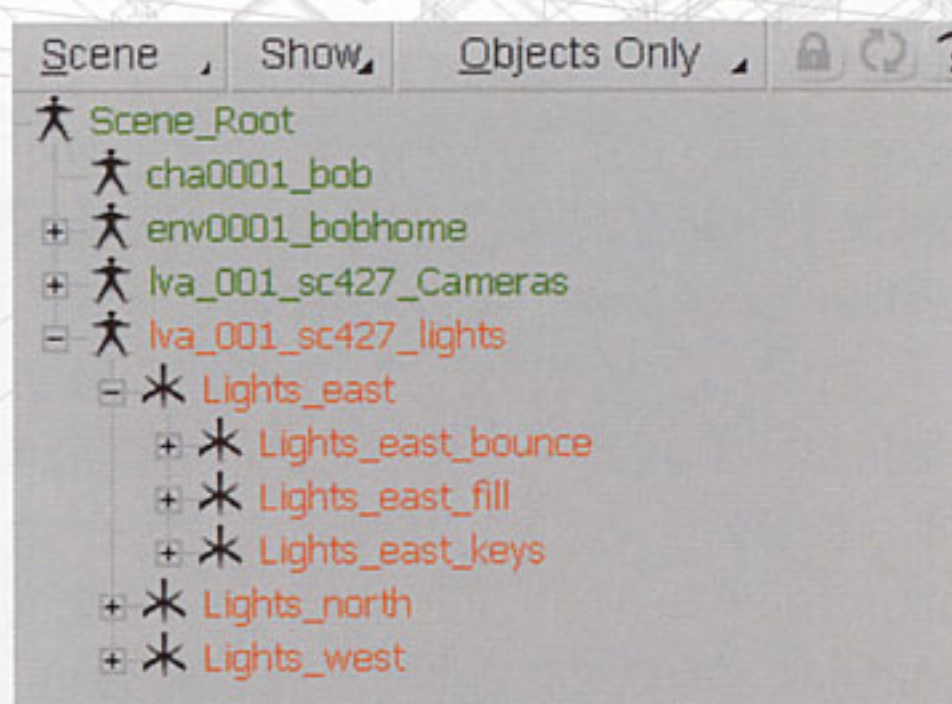


image 07 hierarchies names

instance, I want to export the lighting set of shot 290 to shot 378, then I will name the model `lva_123_sc290_lights`.

When I work in a scene, I put everything under models, which I then name very carefully. I don’t feel it is necessary to rename each model with the entire IDs and names, but it might help if you plan on exporting some models from it. (*image 06 modelnames*)

## Hierarchies

Breaking your scene down in to smaller chunks using hierarchies will make your life much easier. However, it is not necessary to name each child of the models with the IDs and names, as long as the names you use are meaningful and consistent. Renaming every single object in a scene with this information would be quite a tedious task. After all, the point of hierarchies is to ease workflow, not to slow it down. (*image 07 hierarchies names*)

## Groups

Groups have many uses, as they allow you to share properties and materials among their members. Store your groups under the models that contain the objects which they control. This way the groups are exported with the models. (*image 08 group names*)

Of course, groups have to be carefully named, too. When you use groups for shared materials, always put the element ID and full name, followed by “\_mat” to indicate this group shares a material.

If the group is designed to control properties, like motion blur, geometry

approximation or visibility, then rename the group with the property type, followed by the extension “\_pty.”

## Passes

Passes are another matter. If you want to do things right, you should rename your passes with the shot ID and the name of the pass. Leave the Default\_Pass alone. Renaming it might cause you problems later on. (*image 09 pass names*)

You may wonder why we don’t use the entire name. This is because passes contain everything in the scene. Even if you want a pass to isolate an element, the entire content of the scene will be present — in invisible partitions.

Partition naming is pretty much up to you, provided names used are meaningful. If a partition hides its members, it should be clear just by its name that it hides its members. My favorite name for these is Invisible Objects/InvisibleLights. I also always leave the Background partitions alone.

## Objects

Unfortunately, objects have to be as carefully named as the models. Use the entire element ID and name, plus an incremental number if there are copies of the object. For example: `pr0561_01_bobhometable_toppart01`. This is a must if anyone needs to retrieve an object from your scene. For props and environments you can skip the element name (in the previous example, that would be “bobhometable”), but this is not an option with characters, where there may be many scripts and scripted operators design to work with explicit names.

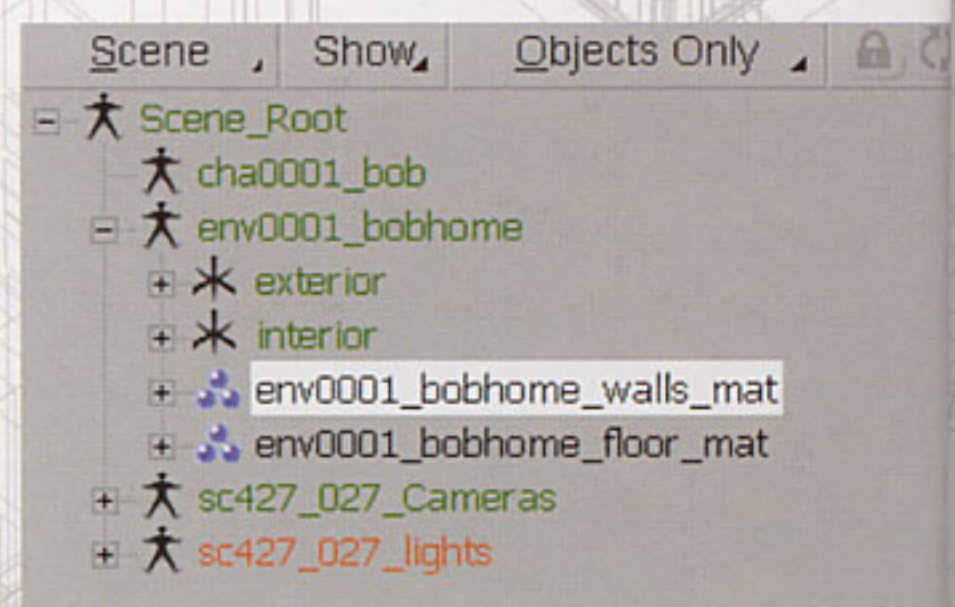


image 08 group names



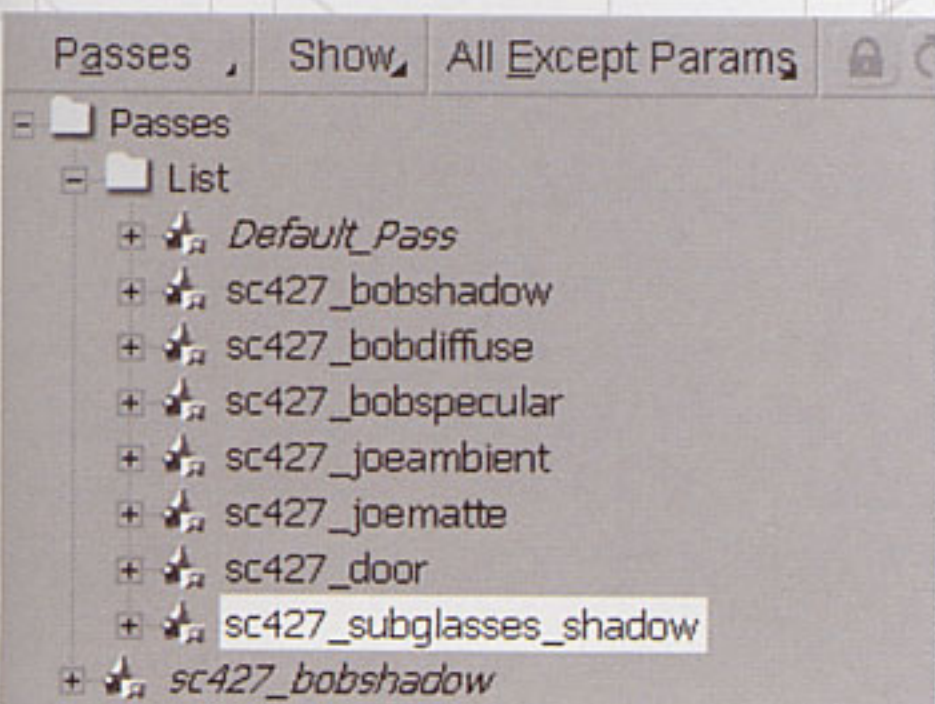


image 09 pass names

## Lights

Lights are a special case. First, avoid putting lights under the model of an environment. Why? This will make your life much easier when you have to export your light set to another scene. You won't have anything to take care of beyond pushing the export button. Put the lights under their own model and name that model with the shot ID followed by "\_lights."

How you organize your light model is pretty much up to you. I use lots of nulls to create separate hierarchies. In case of small environments, I have hierarchies for bounce lights, key lights, character lights and so on. On larger scenes, I have a hierarchy for each part of the environment. Lights used only for characters have their own hierarchy. Of course, if you do that, you must rename the parents and the lights very carefully. Names must clearly say what they do. For example, a light used as a fill light for a character could be named "cha\_Bob\_BlueFill\_point01."

Big scenes with hundreds of lights pose an organizational challenge. When I name my lights (and their parents), I first start with "lv" (level), followed by a geographic reference based on the top view, like "nrth," "sth," "w" or "e," then by the description of the light and its number. I end up with names like lv2\_w\_TopFillWhite\_spot02. I always rename spot roots to match the name used for the child spot, with "\_rt" at the end. Interests get "\_in" at the end of their name.

## Materials

Materials pose another challenge. You have to make a choice: do you put a name

that describes the type of material, or do you put a name specific to the object on which it is applied to. This is even trickier when the material is shared.

Ideally, in large-scale productions, where materials will be reused often, each object should have its own material, even though the material is used on many objects. (Unless the scene contains so many common objects that it is just not practical, as in a crowd scene or a forest scene.) The reason is because a unique material is easier to manage when exported to a .preset file or to another scene. Because of that, each material should be considered an element. This means its name will be the element ID that it is attached to, followed by the name of the material. So a material name might look like: cha0001\_02\_bob\_tshirt.

## Texture maps

Texture maps are named exactly like materials: the element ID and name, plus a description of the texture. For example, the color and bump textures for Bob's shirt would be named cha0001\_02\_bob\_tshirt\_col and cha0001\_02\_bob\_tshirt\_bump.

## Shaders

Output shaders, global environment shaders, lens shaders and so on are more related to shots than to elements. They should have the shot ID in their name, followed by their proper name. For example, a lens shader in shot 256 could be named lva\_123\_sc256\_dof.

On the other hand, if you have shaders that you use frequently in many scenes, you will probably import them via .preset files or merges. Such shaders should be named like elements. If a shader is specific to an environment (let's say you have a nice glow shader for each shot taking place on a given street), you could name it with the environment name followed by its own name. In this example, you might use a name like env0034\_03\_mainstreet\_yellowglow.

## Rendering frames

Rendering frames follow a very simple rule: the shot ID followed by the pass name. The

pass names should always be the same. For example, a depth pass should always be the shot ID + "\_depth" + the frame number. For instance, the depth pass frames of shot 549 might be named lva\_123\_sc549\_depth\_001.pic, lva\_123\_sc549\_depth\_002.pic, and so on.

In addition, each pass should be put in a separate folder. Such folders need not be renamed according to the shot, but it may help to do so, in case something goes wrong.

## Footage

If you work in special effects, then you may very well have to work with live footage. Footage should be placed in its own folder, ideally in the root folder. A good name for it is "Sources." Each source should be named with the shot ID, followed by a small word describing the state of the footage. For example, an original might be named lva\_123\_sc019\_ori, while footage with one keyed character could be named lva\_123\_sc019\_bob\_keyed.

## Final word about naming conventions

The ideas I've given are meant for medium to large-scale productions. If you're working on small projects (low budget advertisings, shots for your demo reel and so on), you may not need such a rigid system. (When you are free to decide how you want to structure your projects, then you can choose any naming convention you want.) On larger scale productions, however, a TD will probably set up the naming conventions, so you'll have to live with it!

Also, every production company has different needs. Some of the ideas presented here might not be compatible with the production company where you work. There are many different ways to set up a "naming convention" pipeline. This is just one of them. Nonetheless, having consistent and relevant naming conventions is a valuable production skill, and it will save you lots of headaches. 🟡



# GNOMON ONLINE

Maya Training and Community  
*modeling.texturing.lighting.rendering.animation.character.effects*

- » Learn Maya through streaming video via Quicktime from your desktop
- » Over 20 hours of online training, tutorials and helpful tips
- » 22 new Gnomon Online Forums and Discussions
- » \$49 per month allows unlimited access to tutorials and Forums
- » 30-hour Texture Mapping Course online
- » 110 minutes of free content!



"My thanks to Darrin for the great 'Jewels and Gemstones' tutorial. The whole texturing course was awesome! I ran into some problems with shading networks in Maya (I previously used 3dsMAX), but now it's so much easier....I especially liked the displacement explanations."

Antonio Lelas

www.gnomononline.com

Images © A. Alvarez and D. Krumweide, ZUFUHR Images © Gnomon Inc.



Get Your Copy Today!



GNOMON SCHOOL PRESENTS

HIGH END

MAGAZINE



INDUSTRY NEWS -- MAYA T-PLUGINS -- MAYA  
SOFTWARE -- SOFTWARE LIST SERVER  
MAYA LIST SERVER -- TIPS / TUTORIALS  
RENDERMAN FILES -- ROSETTA STONE  
ALIAS STUDIO SYSTEMS -- CONTESTS /  
INDUSTRY TOLLS -- MAYA SHADERS  
ANIMATOR TOOLS -- MEL SCRIPTS

# HIGHEND3D.COM



YOUR HIGH-END 3D RESOURCE AND COMMUNITY WEB SITE



## HIGHEND CAREERS

World's leading digital career placement service

### Attention Artists!

- Upload your resume
- Upload your portfolio
- Be seen by major studios worldwide

### Attention Employers!

- Search through thousands of qualified artist's resume and portfolio using our proprietary artist management system
- Turnkey solution for all processes of hiring digital artists for your next project

[www.highendcareers.com](http://www.highendcareers.com)



# Using XSI to extract Normal & Displacement maps

BY OLIVIER OZOUX

## Introduction

**N**ormal maps (also known as bump maps) and displacement map textures are a great way to add visual complexity to models while keeping the geometry light enough for animation and rendering. For film and TV applications, these maps are usually hand-painted in a 2D or 3D paint program to add small high-frequency details to the final geometry, but more recently have also been used to preserve information from high-resolution scanned data. With the latest generation graphics hardware, games like the upcoming Doom III or Half-Life II are also starting to use real-time bump mapping to make objects appear more complex than they really are.

This tutorial shows how you can use the RenderMap tool in SOFTIMAGE|XSI to extract the differences between a high-resolution object and its simplified version, and store them as normal map and displacement maps.

### Step 1: Create the low-resolution mesh

The high-resolution object used in this example is the Stanford Bunny, which is taken from the Stanford 3D Scanning Repository. The original polygon mesh object based on the scanned data contains 69451 triangles. After importing in XSI, I created a low-resolution version with 626 triangles to use in this example.

To model the low-resolution mesh, I loaded the original scanned bunny in XSI, and used the snapping tools in surface mode to outline the main flow lines of the object with splines. These were then used as a guide to create the object using the Add/Edit Polygon tool, snapping the resulting vertices to the high-resolution mesh to line them up. In some cases, I manually pulled the vertices above or below the high-resolution surface to better approximate the curvature with the low-resolution object.

The next step is to create a non-overlapping UV space to hold the texture. This can be done by using the Unique UV Texture Projection or by creating a non-overlapping texture manually, using sub-projections in best fit mode, which is what I did, and then packing the different islands as close as possible to each other to maximize the space used.







- Start by creating a base projection covering the entire object. Select the object, and use Property -> Texture Projection -> XY to define the base UV property.
- Open the texture editor, and select the UV. I generally use a black image clip so that the default noicon pattern doesn't distract me.
- Use the Polygon Raycasting or the Polygon Paint tool to select an area, like the ear of the bunny.
- In the Texture Editor, create a cylindrical sub-projection, using the best fit mode. Click on the cylindrical sub-projection icon in the editor's toolbar and pick the Best Fit mode.
- Make sure that the resulting UV don't overlap, and translate the resulting UVs outside the image boundary for now.
- Repeat with all the other areas, using the cylindrical or planar sub-projection where applicable.
- Use the island selection tool (ISL icon) to select and manipulate each of the areas. Scaling them so that they occupy roughly the same area proportioned to the object itself.
- Rotate and Translate the pieces so they occupy a densely packed but roughly square area. Don't try to make them fit the (0,1) UV boundaries just yet.
- Once you are satisfied with the packing, select the entire UV set, and scale and translate it to fit the boundary.



original clay bunny



high resolution scan

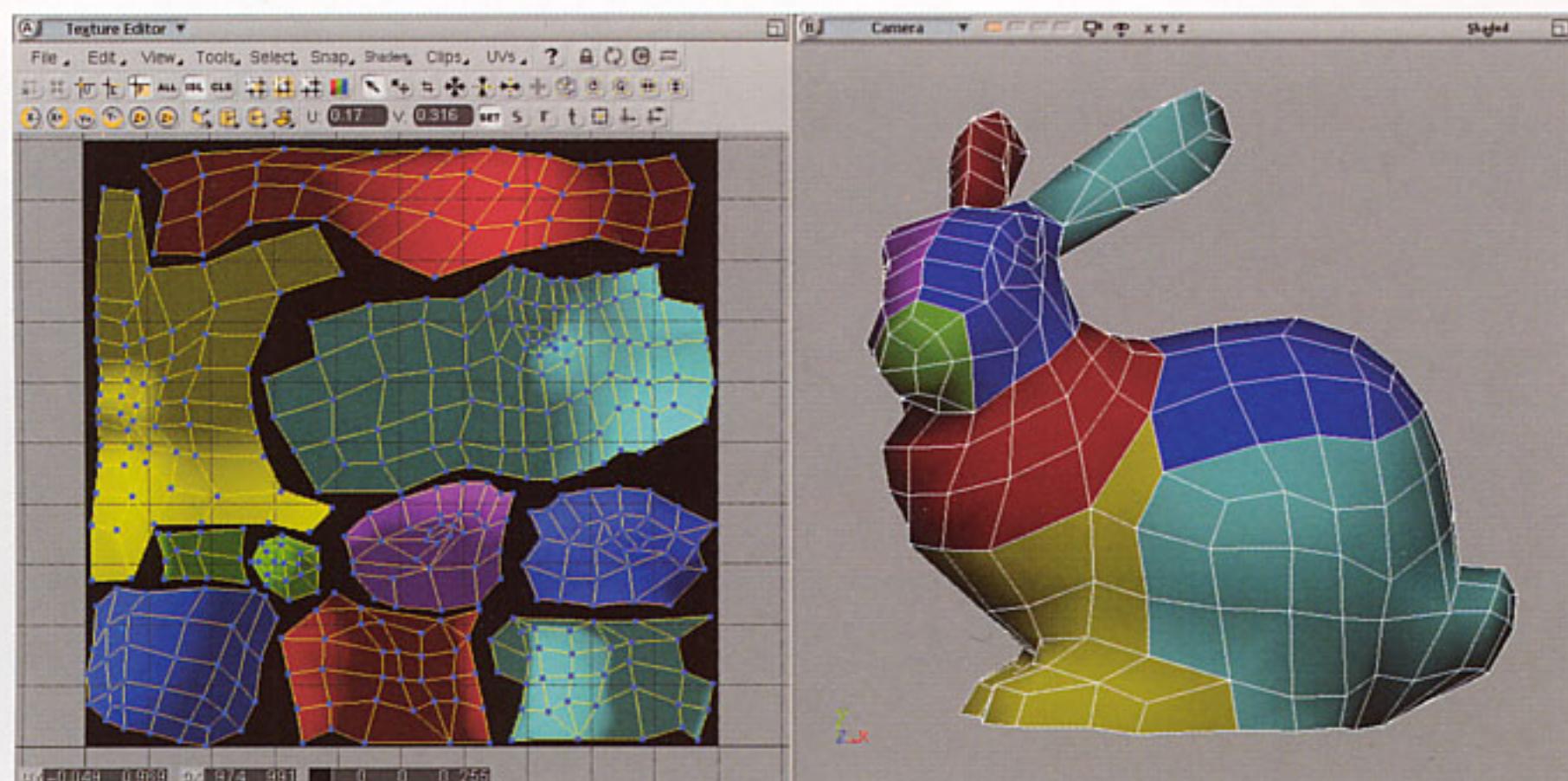


low resolution object



Result with normal and displacement maps





## Create a non-overlapping UV texture space

- Freeze the UV property to get rid of all the operators and projections.

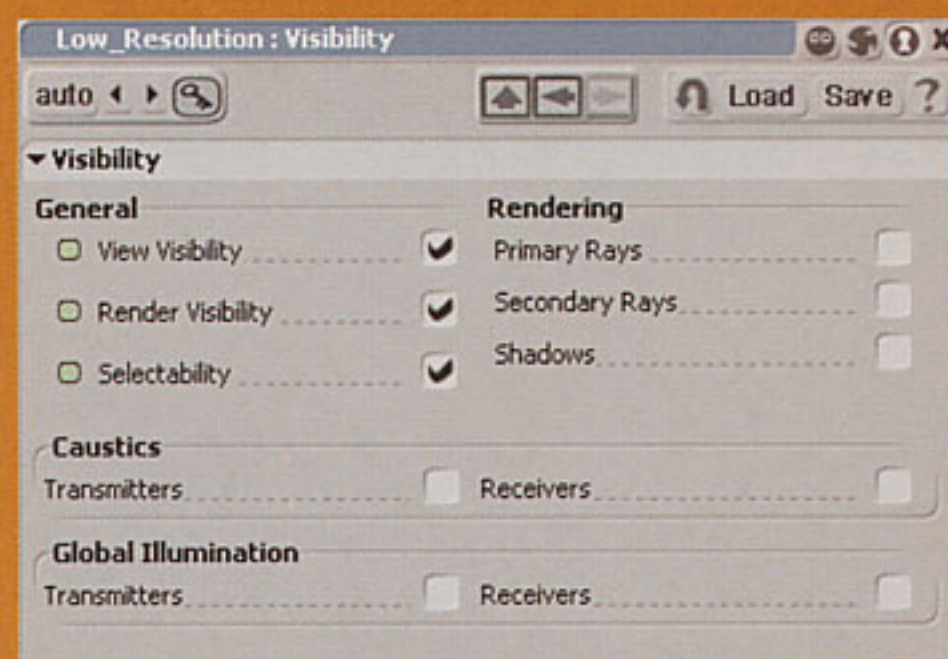
## Create a non-overlapping UV texture space

## Step 2: Preparing the stage for RenderMap

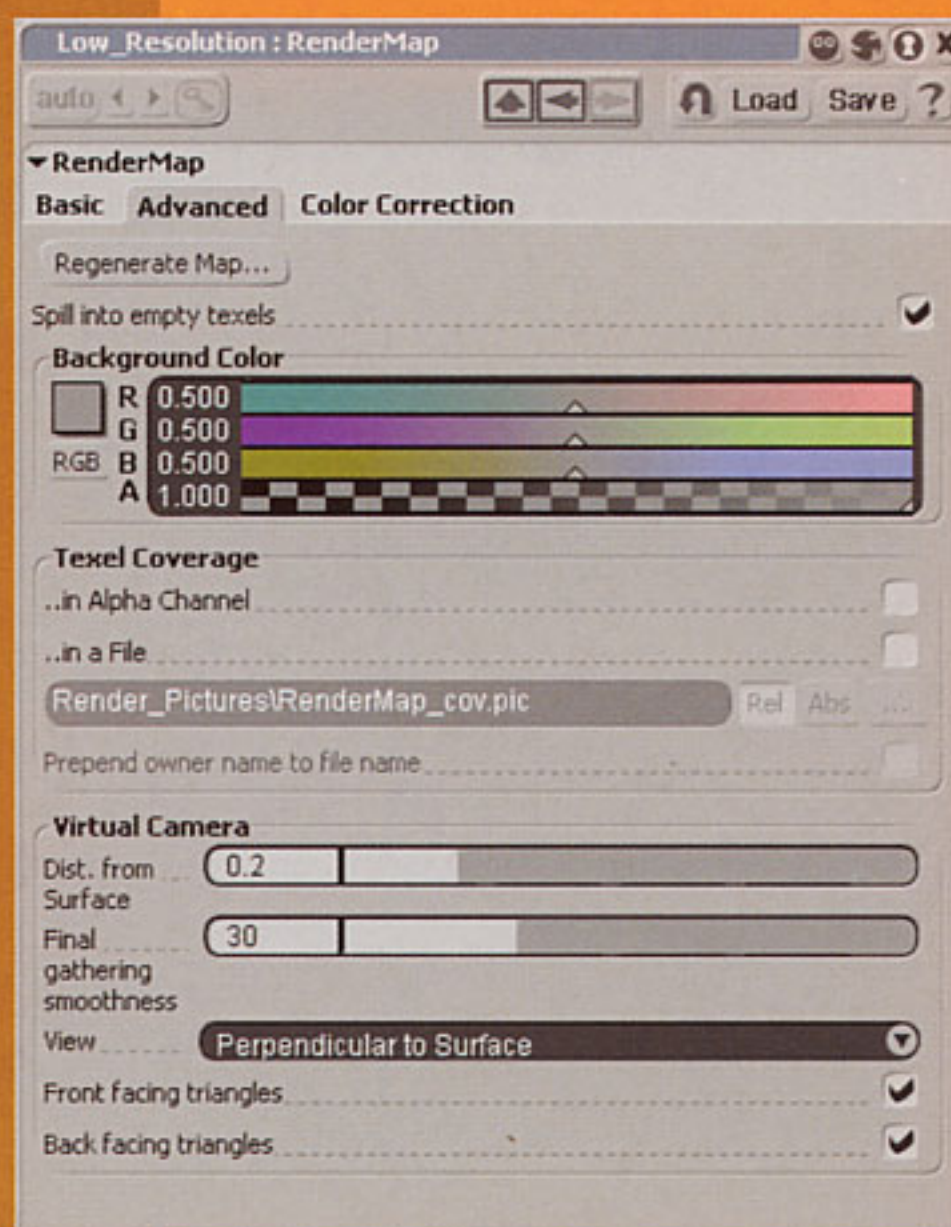
The RenderMap Tool in XSI is designed to extract the surface color of objects and bake it into an image map or a vertex color map. It uses the XSI rendering engine (mental ray) and creates a virtual camera that follows the surface of the object using the selected UV texture space and render resolution as a guide and transfers the resulting surface colors to an image. This image can then be used as a texture on the same object, potentially replacing a complex shading operation with a simple texture map.

Since we want to extract information from the high-resolution mesh and use it on the low-resolution object, we will need to use a different setup.

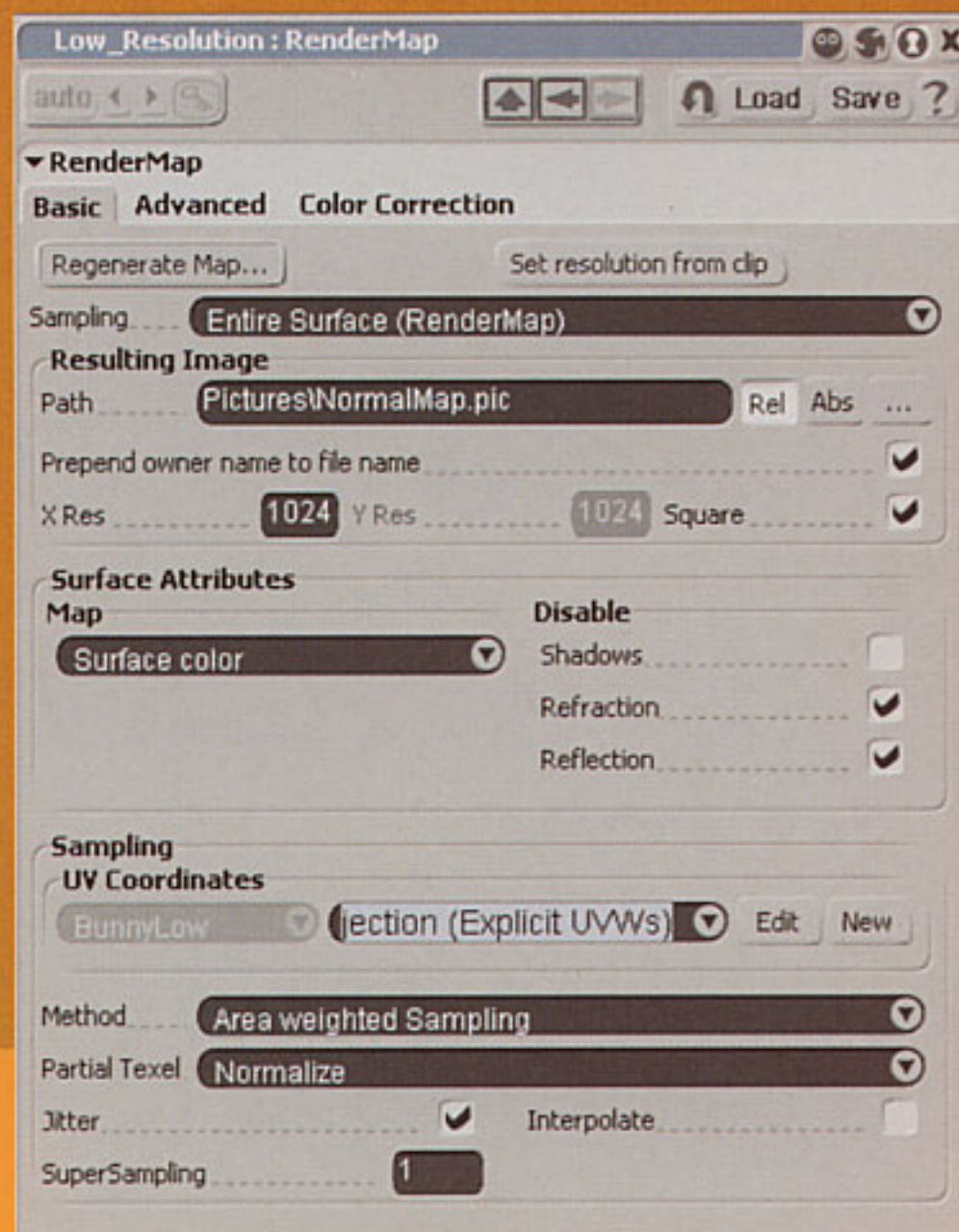
- Make sure that the low-resolution and the high-resolution object overlap one another.
- Open the Visibility Property of that object and change the rendering properties to make the object invisible to primary, secondary and shadow rays. Don't simply turn off the rendering visibility because RenderMap will ignore the object.



## Visibility property settings



## RenderMap Advanced settings



## RenderMap Basic settings

## Visibility property settings

- Select the low-resolution object, and create a RenderMap property. In the render toolbar choose: Property -> RenderMap



- Make sure the sampling is set to: Entire Surface (RenderMap)
- Set the resulting image to: Pictures/NormalMap.pic, the resolution to 1024 and check the Prepend owner name to file name option.
- Set the surface attributes to Surface color and make sure that shadows, refraction and reflections are disabled (checked).
- Select the correct UV property for the low-resolution object
- Make sure that partial texels are set to normalize; this will cause the sampled color to bleed at the edges, rather than reveal the background color.

## RenderMap Basic settings

- In the advanced tab of the rendermap ppg, check the box next to "Spill into empty texels" (See rendermap\_advanced.png or rendermap\_advanced.bmp).
- Set the background color to 50% grey (0.5, 0.5, 0.5) this will be the neutral color.
- Set the virtual camera. Make sure that the Distance to surface is greater or equal to the largest positive difference between the high-resolution object and the low-resolution one. The virtual camera will travel around the object at that distance, so if the distance is too small, it will miss the surface at some point.

## RenderMap Advanced settings

We can regenerate the rendermap at this point, which would create an image containing the color and lighting at the surface of the high-resolution object. Make sure there are no holes in the resulting map, which could indicate that the virtual camera is either too close or too far away from the low-resolution object, and there-

fore is missing the high-resolution surface. A good trick to use at this point is to use the FrontBack shader (In the rendertree: Nodes -> Switch -> Front-Back); you should only see the front color, never the back.

## ColorMap image

## Step 3: Extracting the Normals

Now that we have the basic rendermap setup transferring information from the high-resolution object to the low-resolution one, we need to setup the material shader on the high-resolution object to display the surface normal as a color. This can be done using the rendertree.

## Normals in World Coordinates

The basic way would be to use the vector\_state shader that can return the surface normal as a vector, and convert that to a color. Because vectors range from -1 to 1 and colors from 0 to 1, we can't use the normal vector2color conversion node. We'll extract the three components as scalar values, and convert those individually



color map

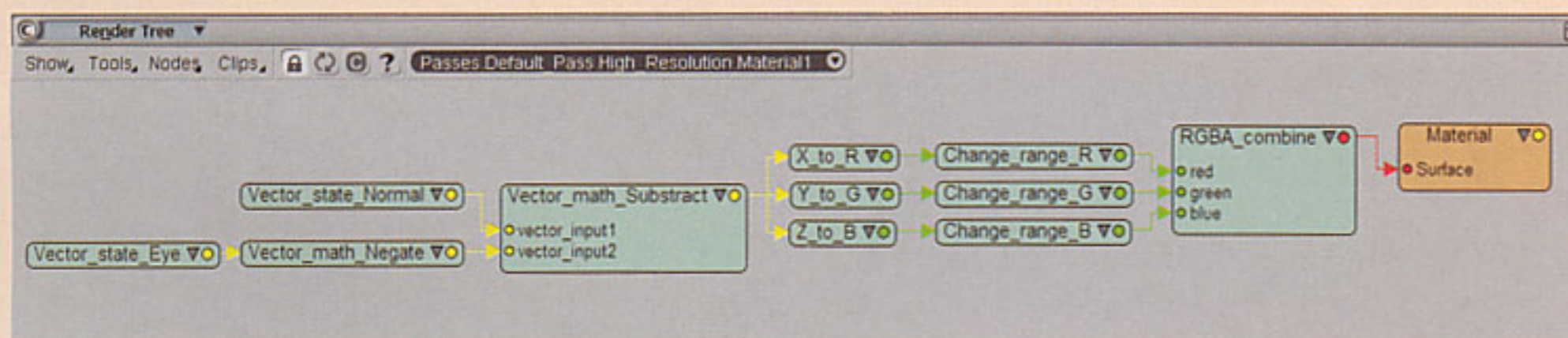


Normals in World Coordinates

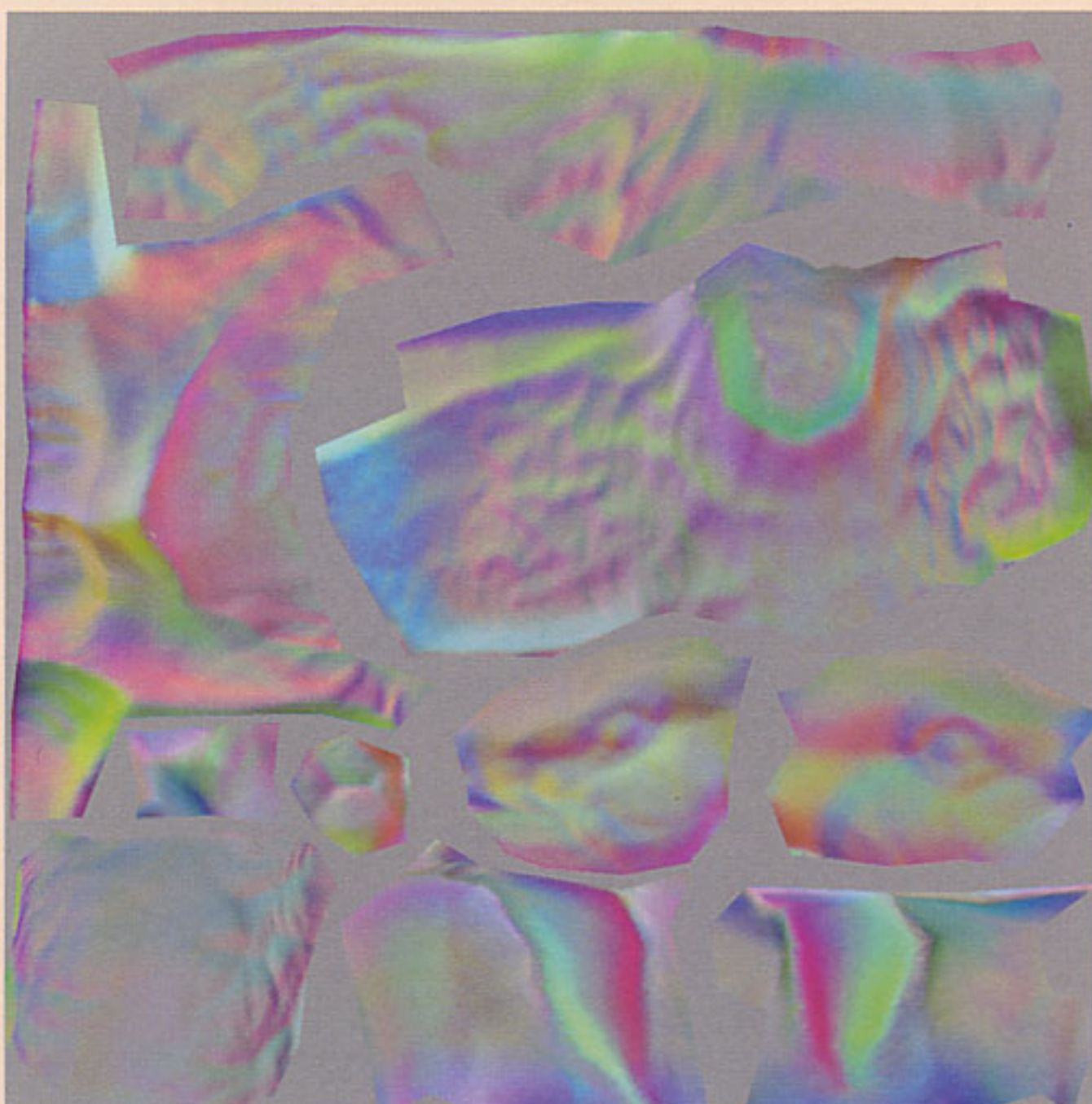


Rendertree to extract world coordinates normals





## Render tree to extract local coordinates normals



local coordinates normals



Low resolution with normal map

before combining them back into a color, which we'll apply to the surface of the object.

- Select the high-resolution surface and open the rendertree. Make sure the object has its own material and that you are not editing the global scene material.
- Get a Vector\_State shader (Nodes -> State -> Vector state) and select the Normal Vector from the popup list. Rename it to Vector\_state\_Normal
- Get three Vector to Scalar nodes (Nodes -> Conversion -> Vector to Scalar) and set them to extract X,Y and Z respectively. Rename them to X\_to\_R, Y\_to\_G and Z\_to\_B. Connect the Vector\_State\_Normal node to each of the inputs.
- Get three Change Range shaders and connect one to each of the vector to scalar nodes above. This is where we re-map the range of the vector into something that will fit into a color. For each node, set the old range to (-1,1) and the new range to (0,1). Rename the nodes Change\_Range\_R, Change\_Range\_G and Change\_Range\_B.

- Finally, get a RGBA Combine shader (Nodes -> Color Channels -> RGBA Combine) and connect each of the change range nodes to the RGB inputs. Connect the output to the surface port of the material.

## Render tree to extract world coordinates normals

If you were to render the high-resolution object at this point, you would get an image where every point on the surface is shaded based on the normal orientation in world coordinate. We can't really use that technique to transfer the normal map to the low-resolution object, because that map would only be valid and shade correctly if the low-resolution object keeps the same orientation as the high-resolution object at the time the map was transferred.

What is really needed is a way to encode the difference between the surface normal on the low-resolution object, and the corresponding surface normal on the high-resolution object. Because we are using RenderMap to extract the color, we know that the virtual camera is going to be facing down, perpendicular to the low-resolution surface, exactly in the direction of the normal, but pointing the other way. We can then use the eye vector (the camera direction) and subtract it from our normal vector to get a relative change. Also, because we can assume that our virtual camera will never hit a surface that is facing away from it, we can increase the dynamic range of the map by scaling the values by half.

- Start from the previous rendertree, and



disconnect the Vector\_state\_Normal shader.

- Get another vector state node, and select the Eye Ray Vector. Rename the node to Vector\_state\_Eye.

- Get a vector math node (Nodes -> Math -> Vector-Vector) and connect the Vector\_state\_eye to input1. Select Negate Vector input1) as the operation. Rename the node to Vector\_math\_Negate.

- Get another vector math node. Connect Vector\_state\_Normal to Input1 and Vector\_math\_Negate to input2. Set the operation to Vector Input1 - Vector Input2. Rename the shader to Vector\_math\_Subtract.

- Connect the output of Vector\_math\_Subtract to each of the X\_to\_R, Y\_to\_G and Z\_to\_B nodes.

- Edit each of the Change\_Range\_R, Change\_Range\_G and Change\_Range\_B shaders so that the input range is (-0.5,0.5) instead of (-1,1).

### Rendertree to extract local coordinates normals

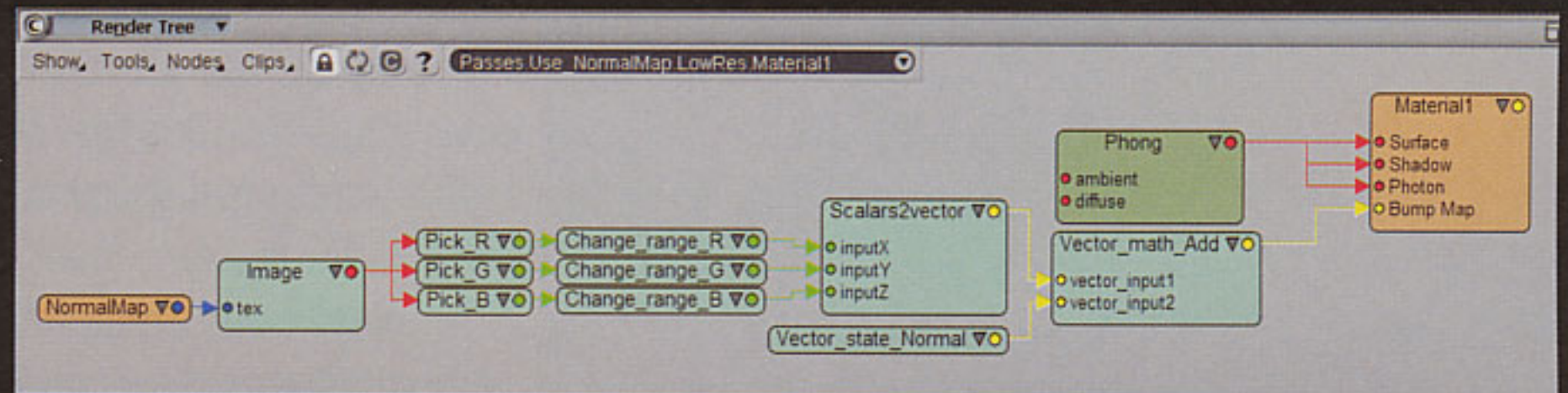
At this point you can use RenderMap to regenerate the normal map image, which would give you the following result:

The next step will be to create a rendertree to convert the normal map images back into a vector to modulate the surface normal of the low-resolution object.

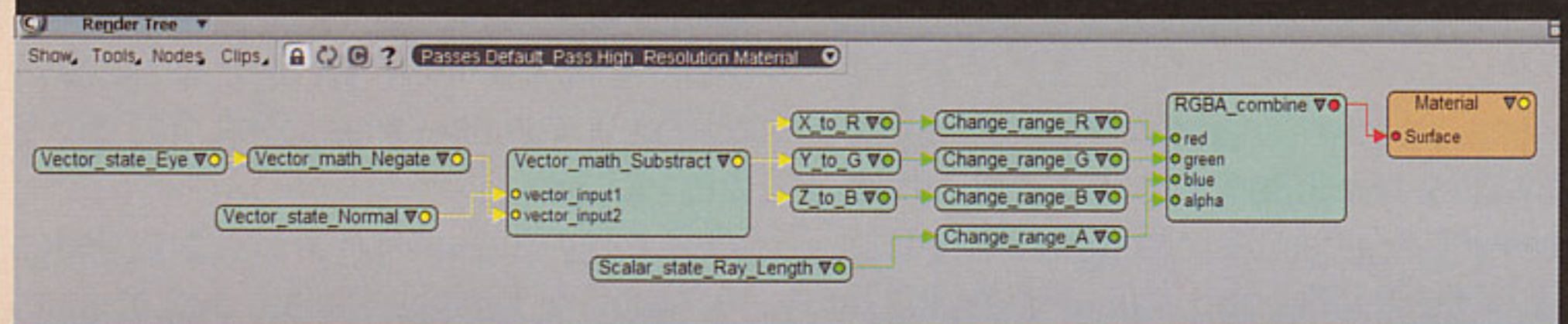
### Local coordinates normals

#### Step 4: Displaying the Normal Map

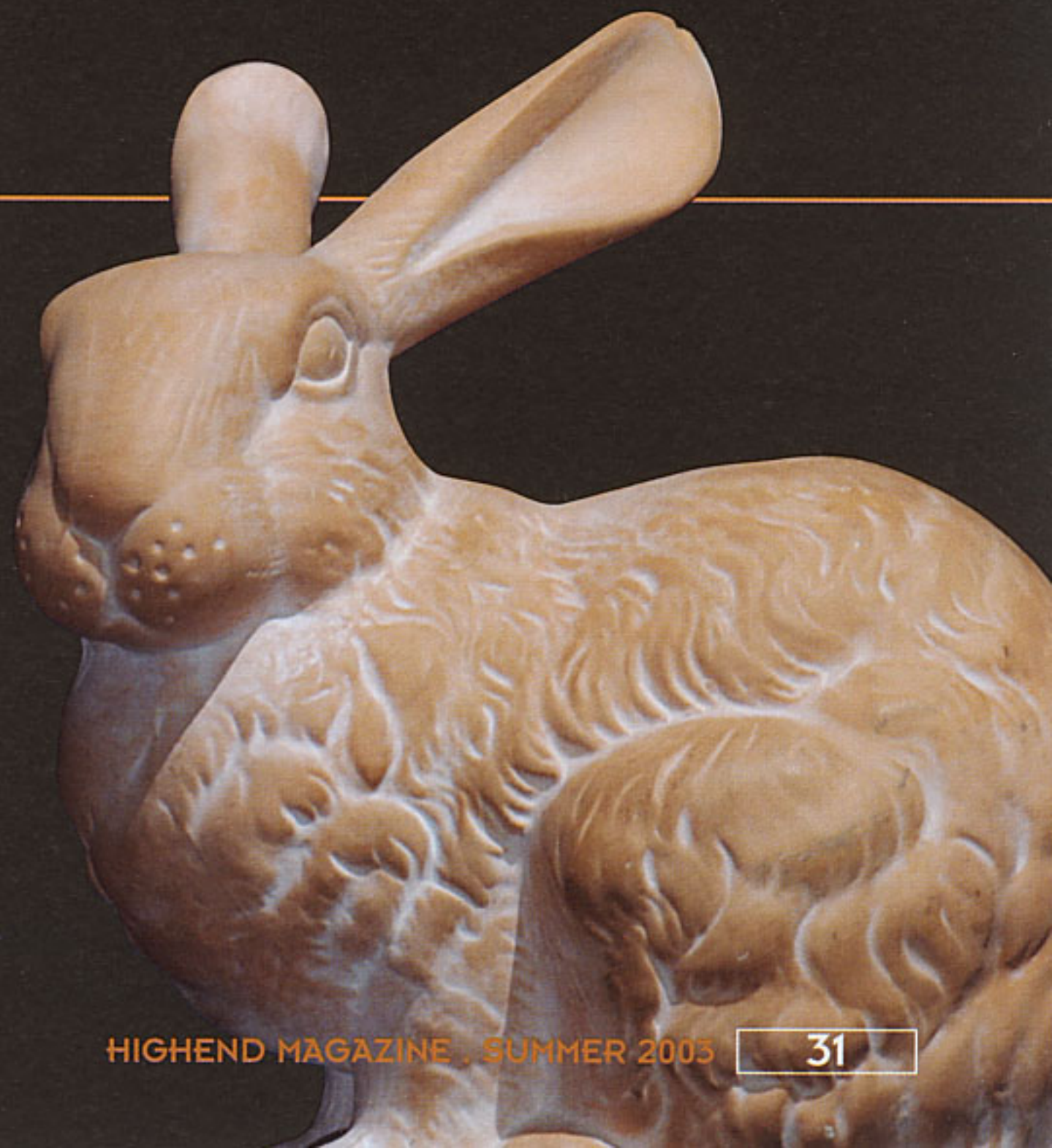
We now have an image that contains the difference between the normal on



Rendertree to display the local normals



Rendertree to extract the normal and displacement







Render tree to restore the normal and displacement

the low-resolution object and the high-resolution object encoded into the RGB, with the middle grey values meaning no difference, and the high and low values positive and negative differences between the two surfaces.

We now need to convert the color data from the image back into a vector that we can use to modulate the surface normal of the low-resolution object in order to make it react to the light in the same way as the high-resolution object would, therefore making it appear to have more visual complexity than it really has.

#### Low-resolution with normal map

- Select and hide the high-resolution object.
- Select the low-resolution object, and change its visibility property so that it now renders primary, secondary and shadow rays.
- Update the rendertree to edit the material on the low-resolution object.

- Get an image shader (Nodes -> Textures -> Image) and connect it to the image clip we just rendered. Make sure you select the correct UV texture space. Test the result in the render region by connecting the image node to the Ambient and Diffuse inputs of the Phong shader node connected to the material.

• Get three Color Picker shaders (Nodes -> Color Channels -> Picker) to extract the RGB components of the image map. Connect them to the Image node, and set them to their respective channels. Rename them Pick\_R, Pick\_G and Pick\_B.

- Get three Change Range shaders (Nodes -> Math -> Change Range) and connect one to each of the color picker nodes above. This is where we re-map the range of the color into a vector. For each node, set the old range to (0,1) and the new range to (-0.5,0.5). Rename the nodes Change\_Range\_R, Change\_Range\_G and Change\_Range\_B.

- Get a scalars to vector shader (Nodes -> Conversions -> Scalars to Vector) and connect the change range nodes to the InputX, InputY and InputZ respectively.

- Get a Vector\_State shader (Nodes -> State -> Vector state) and select the Normal Vector from the popup list. Rename it to Vector\_state\_Normal.

- Get a vector math node (Nodes -> Math -> Vector-Vector) and connect the Scalars2Vector to input1 and the Vector\_state\_Normal to Input2. Select Vector input1 + Vector input2) as the operation. Rename the node to Vector\_math\_Add.

- Connect the result of the Vector\_math\_Add node to the Bump Map port on the material. Draw a render region around the low-resolution object.

#### Rendertree to display the local normals

#### Step 5: Extracting the Displacement Map

Going back to the setup we did on the high-resolution object to extract the normal direction, we can modify it to also track the difference in the distance between the two surfaces. Since we have setup the RenderMap property so that the virtual camera samples the surface from a constant and known distance from the low-resolution surface, if we can measure the distance traveled by the ray before it hits the surface, we can store that information into the alpha channel of the normal map that we are generating.

However, because the displacement is an arbitrary distance, we are going to re-map the values from the maximum positive distance to the maximum negative distance between the two surfaces and normalize those values from 0 to 1 so that it can be stored in the image map. In the case of the bunny, the maximum positive displacement is 0.2, and for simplicity we will assume that the maximum negative displacement will be -0.2 from the low-resolution surface. This is important because the range will not be stored in the map, but is needed to reconstruct the displacement on the low-resolution surface afterward.

- Select the high-resolution object and refresh the rendertree.



- Get a scalar state shader (Nodes -> State -> Scalar State) and select the Ray Length as the state parameter. This will return the distance between the virtual camera and the high-resolution surface. Rename the node `Scalar_state_Ray_Length`.
- Get another change range node (Nodes -> Math -> Change Range) and connect the `Scalar_state_Ray_Length` to its input. Since the ray length is 0 at the origin of the camera, we need to re-map the value from (0,0.4) to (0,1). Rename the node • Connect the `Change_Range_A` node to the alpha channel input of the `RGBA_Combine` node.
- Regenerate the map.

### RenderTree to extract the normal and displacement

#### Step 6: Displaying the Displacement Map

Restoring the displacement map on the surface is very simple, but as with the extraction of the displacement values, the results are dependent on re-mapping the values to the correct range. Starting with the rendertree setup to restore the normal map on the low-resolution surface, we can extract the displacement value stored in the alpha channel of the image, re-map it to the proper range, and use that value to drive the displacement port on the material.

- Select the low-resolution object and refresh the rendertree to edit its material.
- Get an additional color picker shader (Nodes -> Color Channels -> Picker) and connect it to the texture node to extract the alpha channel component. Rename the shader `Pick_A`.

- Get also another change range shader (Nodes -> Math -> Change Range) and connect it to the `Pick_A` shader. Make the old range (0,1) and the new range (0.2, -0.2), since we stored the alpha channel with black as a positive displacement, grey as neutral, and white as a negative displacement. Rename the node `Change_Range_A`.
- Connect the change range node to the displacement port of the material. Make sure that you adjust the Geometry Approximation property of the low-resolution object to render the displacement correctly.

### RenderTree to restore the normal and displacement

#### Conclusion

For real world applications, I would invest the time to write a custom shader and script to extract and display normal and displacement maps, but one of the goals of this tutorial was also to show how you can prototype shaders using the rendertree and the built-in atomic shad-

er nodes to manipulate values, colors and vectors. In fact, this is what people at Softimage have done (see sidebar).

Also, the example in this tutorial doesn't track the tangent space of the surface, which means that in some cases, the normal map can appear inverted. This is because we are using only two vectors, the surface normal and an offset vector stored in the map, so the shader tree is acting like a direction constraint on the normal, and has the similar behavior of appearing to flip if the orientation goes past the hemisphere in which the effect was applied. Storing and using a tangent space vector (i.e. a vector that is perpendicular to the surface normal) and using it as an up-vector during the restore operation would solve that limitation.

RenderMap is a very powerful tool that can transfer any type of rendering attribute from a surface to another or even convert image maps from a specific UV domain to another, and this example is only one of the many effects you can achieve with RenderMap and the built-in shaders in `SOFTIMAGE|XSI`. 🟡

## New in `SOFTIMAGE|XSI` v3.5

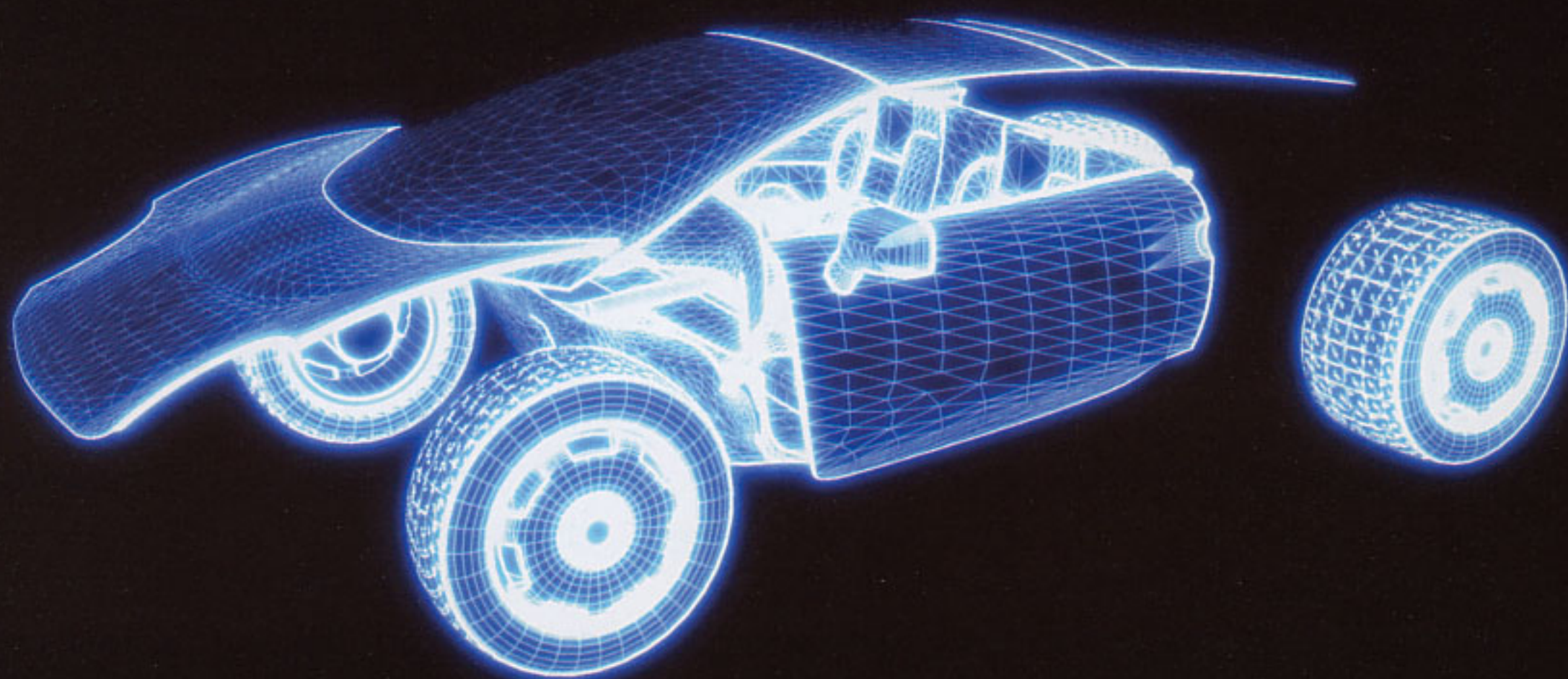
With version 3.5 the whole process of creating and using normal maps with real-time shaders has been simplified.

The local NetView database contains an add-on called GPU Surface FX, which installs dedicated scripts and custom shaders to automate the process of extracting surface normals and baking them to another object, along with real-time shaders to display the results using either DirectX or OpenGL.



# Creating Custom Glows

BY FRANCOIS LORD



One of the things I've always thought was lacking in compositing packages was a good glow. As a 3D animator myself, in the old days I was using SOFTIMAGE|Eddie to composite my shots. All there was in Eddie was that lame Diffuse Glow from the Painterly Effects. It created a soft white halo around bright areas of the image without much control. So I always ended up doing it by hand every time I needed one. When Shake arrived, with its macro possibilities, it didn't take too long before I figured it was possible to make my own glow with all the parameters I wanted and make a button out of it. It all happened on a commercial we were beginning for an electricity company. They wanted a 30-second commercial, full 3D with a cool luminescent look. I knew I needed a good glow that I would be using on all the shots. I took an afternoon and created FLGlow. Of course it continued evolving afterwards to FLGlow2 and later to FLGlow3. Speed improvement was very important in the development.



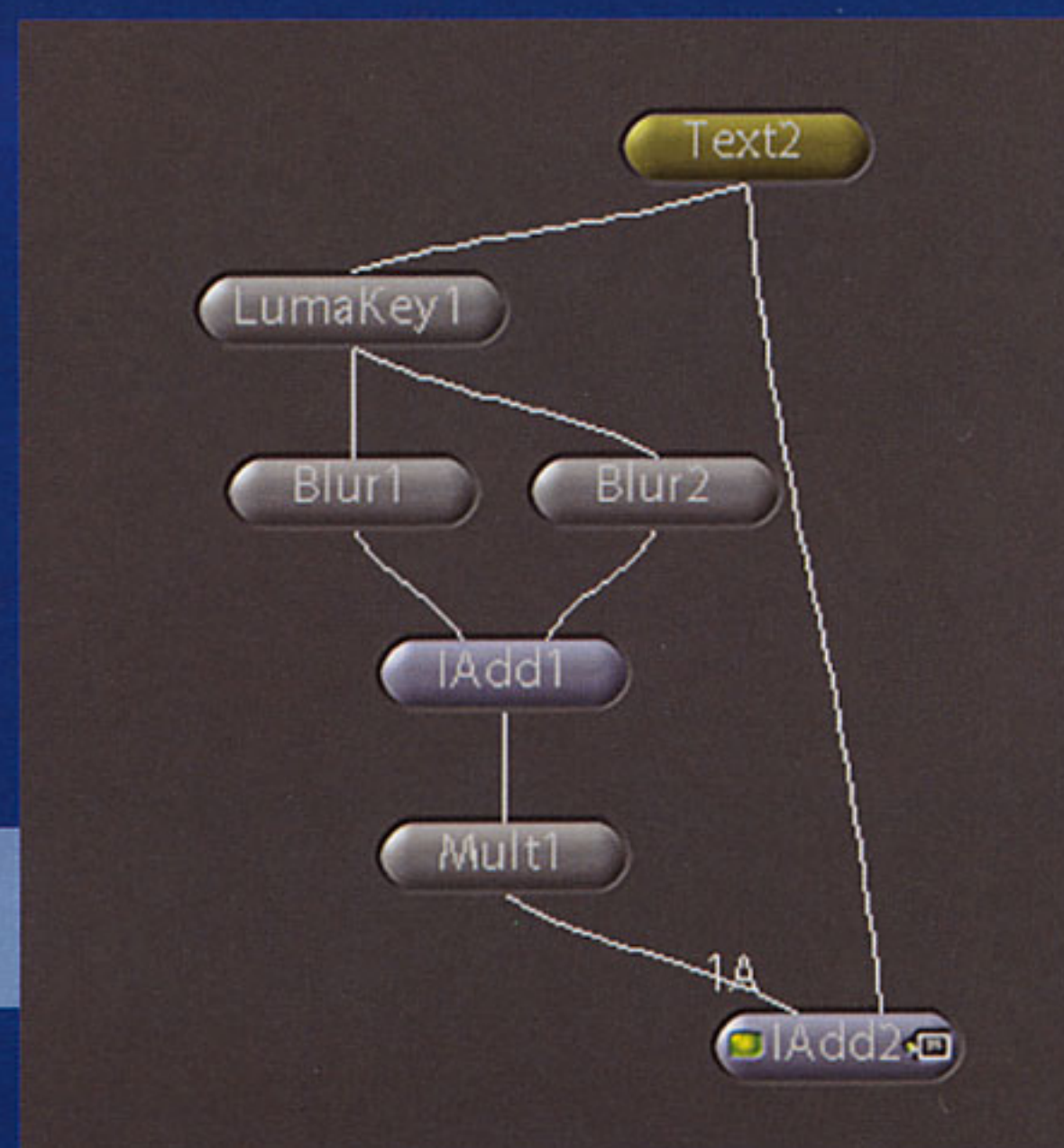


forest before



forest after

A glow is basically a blur composited on the original image. This gives a very soft and boring glow. To make it a little more interesting, we must have many different blurs added together with different sizes. In FLGlow3, only two blurs are used. One is small, and the other is three times larger. They are added together and then multiplied by a user-defined color to allow an overall tint. The result is added to the original image. All we need then is a way to make only certain parts of an image glow. This is achieved using a lumakey. We just



glow 1

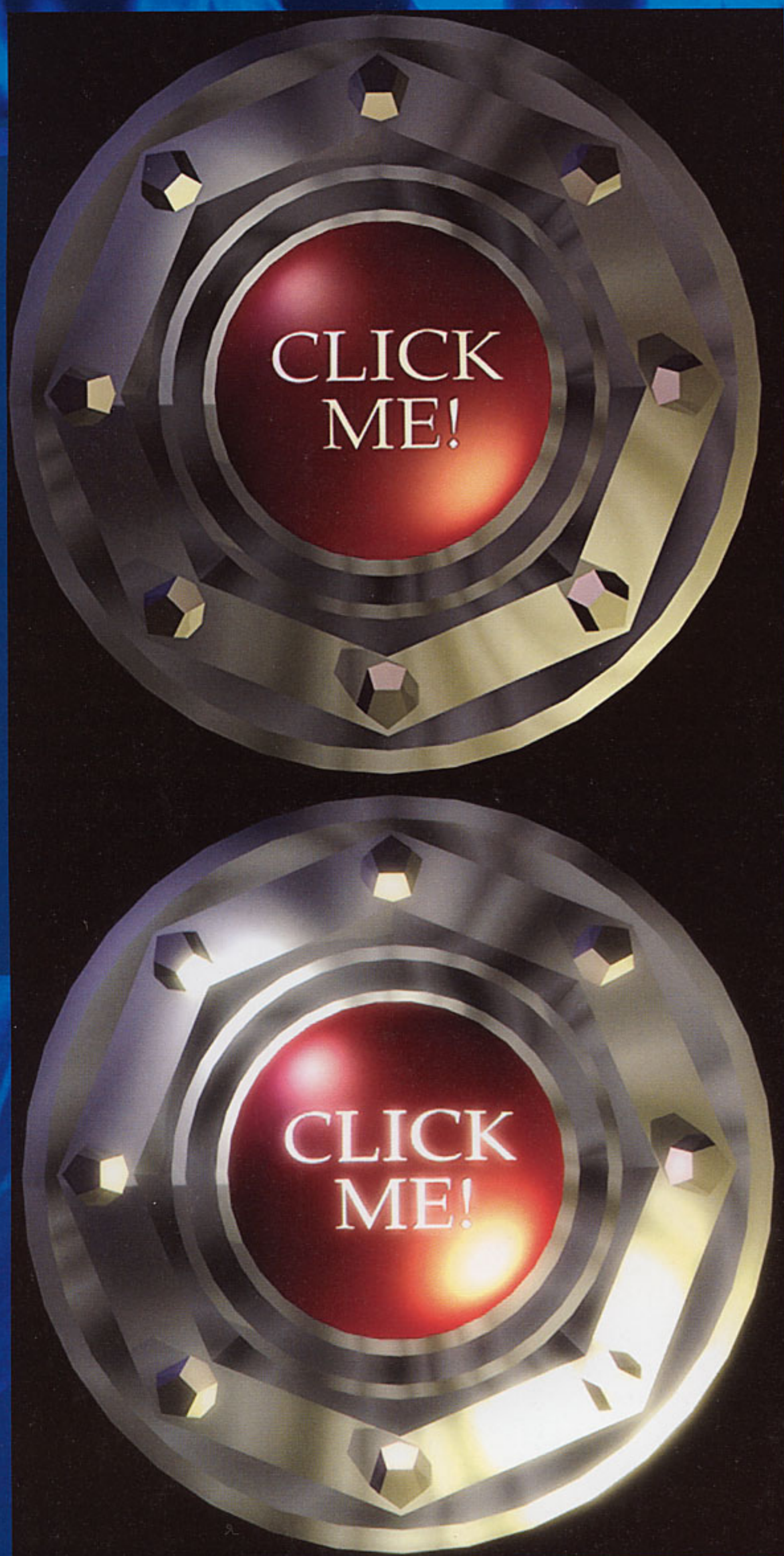


glow 1 before



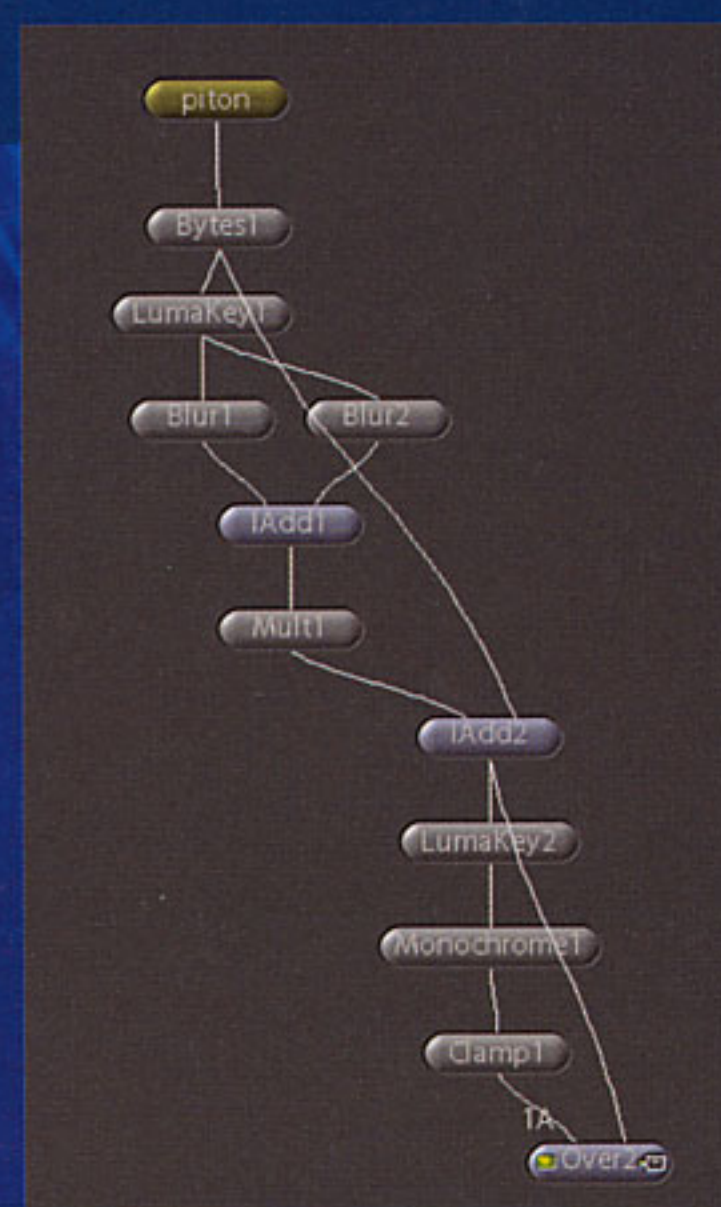
glow 1 result





hook it before the blurs. This will filter the dark areas of the image to keep only the bright ones to glow. This is all adjustable for each shot. That's it! We have a working glow!

But that wasn't satisfying for me. I also wanted to recreate the effect of film, where very bright colors become desaturated (burn effect). We add another lumaKey to select areas where pixels have values above 1.0. The Low and High settings are set to 0.8 and 1.2. This must be done in 32-bits per channel (float mode) from the top of the tree. The result is completely desaturated, clamped to zero to avoid negative values and composited (over) the glow. The effect is subtle, but it gives a little more



glow 2





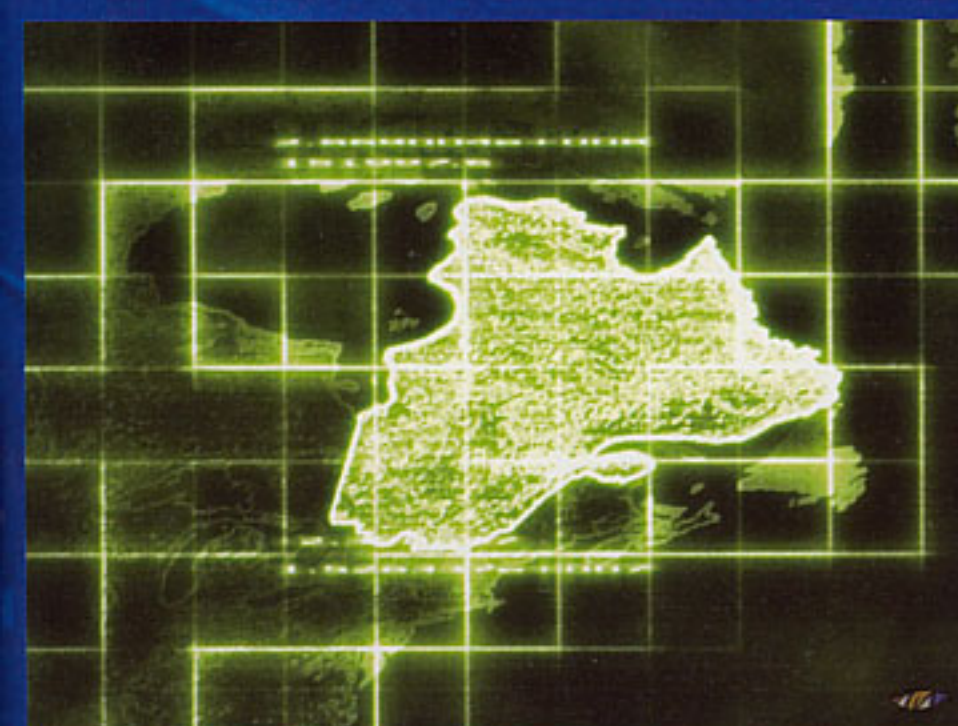
realism. The highlights don't have that yellow ring around them.

The final FLGlow3 has a switchmatte at the top to allow the input of an external matte. There is a select node to bypass the burn effect if desired. It also selects 16 or 32-bits automatically, depending on the use of the burn effect, but returns an image in the same bit-depth as the input image. What I like about this technique is that we can use it to create a bold glow effect and as a subtle diffuse lens filter.

Since I made this macro, everybody here in 3D department uses it regularly and finds it very useful. This is why I decided to post it on highend2D.com. 🟡

### *The exposed parameters are pretty self-explanatory*

- Spread is the size of the small blur (the large blur is three times this size).
- GlowFade is the amount of glow added to the original image
- Glow\_Lo and Glow\_Hi are the low and high values of the lumakey. By default, it will apply the glow only to the bright areas of the image. I often set both values to zero, to make sure everything is glowed...especially with motion graphics.
- Color is the overall tint.
- MatteChannel is used only with an external mask. I didn't find a way to make it active automatically.
- Burn will toggle the desaturation effect.



glow 2 results





# Gallery

To have your work considered please submit to  
[www.highend3d.com/artists/add.3d](http://www.highend3d.com/artists/add.3d)



This image was done for practice and portfolio filler. I wanted an image that revealed more and more the closer you got. Looking at the image as a whole...it's a bored looking drunk buy. Get closer and you can see the crust in his eyes...yummy!

## **Joel F. Thomas**

**Workplace:** Sony Computer Entertainment of America

**E-mail:** [screamcheese@stalecracker.com](mailto:screamcheese@stalecracker.com) **Highend Username:** stalecracker

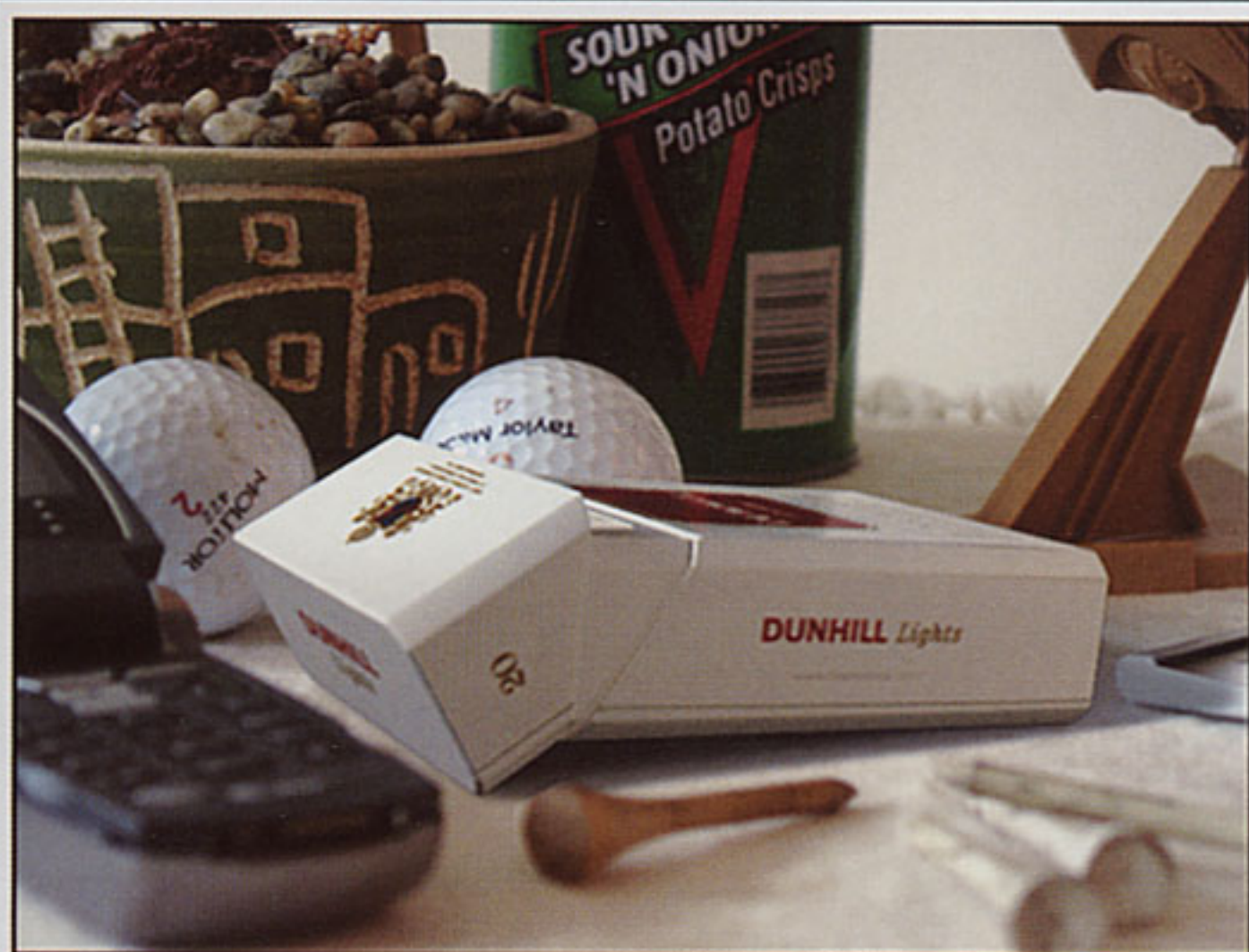
**Software used:** Photoshop 7

**Special techniques:** A few animal sacrifices...that's about it

**Hardware:** P4 1.8GHZ/1GIG RAM

**Render time:** No render time





## ***Kyungsu Kim***

**Where you work:** freelancer

**E-mail:** Kimble92@naver.com    **Highend Username:** Kimble92

**Software used:** Maya 4.0, Photoshop 6.0, After Effects 5.5

**Special techniques:** I am working on the lighting and compositing section in 3D animation; this scene was rendered by around 10 passes. I rendered several beauty, shadow, reflection passes and I composited those passes with After Effects.

**Hardware:** PC Pentium 4.. 1.8

**Render time:** 3 hours



## **Peter Pankras**

**Where you work:**  
Currently looking for a new job!!!

**E-mail:** peter@corvidae.net

**Highend Username:** pjpankras

**Software used:**  
Maya, Raydiffuse, Photoshop

**Hardware:**  
Dual Pentium III 800

**Special techniques:**  
Multiple tweakable layers rendered

**Render time:**  
Not sure, but 12 Hours per shoe  
(raydiffuse taking almost 70%)

**Client:** Nike ...doh







### ***Niklas Ström***

**Where you work:** visual art,  
Stockholm, Sweden

**E-mail:** niklasstrom@telia.com

**Highend Username:** musse

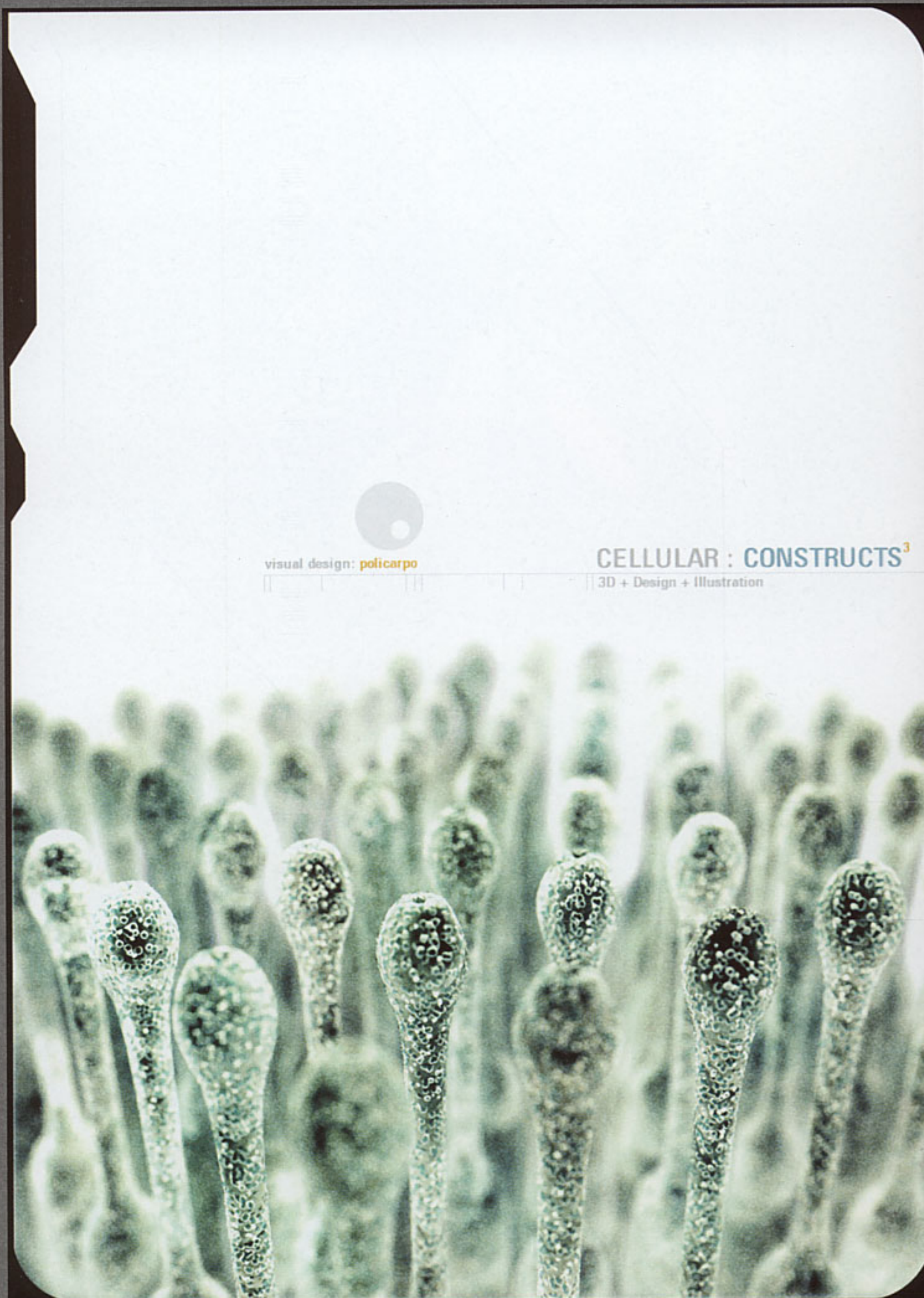
**Software used:** 3d Studio Max, Photoshop



### ***Niklas Ström***

**Software used:** 3d Studio Max, Photoshop





## ***policarpo***

### **URL:**

<http://www.policarpo.us>

### **Where you work:**

Visual Designer in Austin, Tx

### **E-mail:**

[visual.design@policarpo.us](mailto:visual.design@policarpo.us)

**Highend Username:** policarpo

**Software used:** LightWave 7.5 and Photoshop 7

### **Special techniques (the more the better):**

The geometry is relatively dense Subdivision Surfaces which I increased in density at render time. For both images I rendered out a separate depth pass so I could apply my depth of field technique on the fly in post.

Relies heavily on a procedural texture displacement mapping to create the *gnarly* bits on the geometry. Some mild color balance tweaking was also applied make them pop a bit more.

**Hardware:** PIII 1Ghz, 512mb, 64mb GeForce3 Ti4200

**Render time:** approximately 2 minutes





## *White Room*

***Neil Blevins***

**Where you work:** Soulburn Studios

**E-mail:** neil@soulburn3d.com

**Highend Username:** soulburn3d

**Software used:** MAX 3.1, Ghost Renderer (now Brazil), Digital Fusion

**Hardware:** P3-555 with 2 GIGs RAM



TUTORIAL

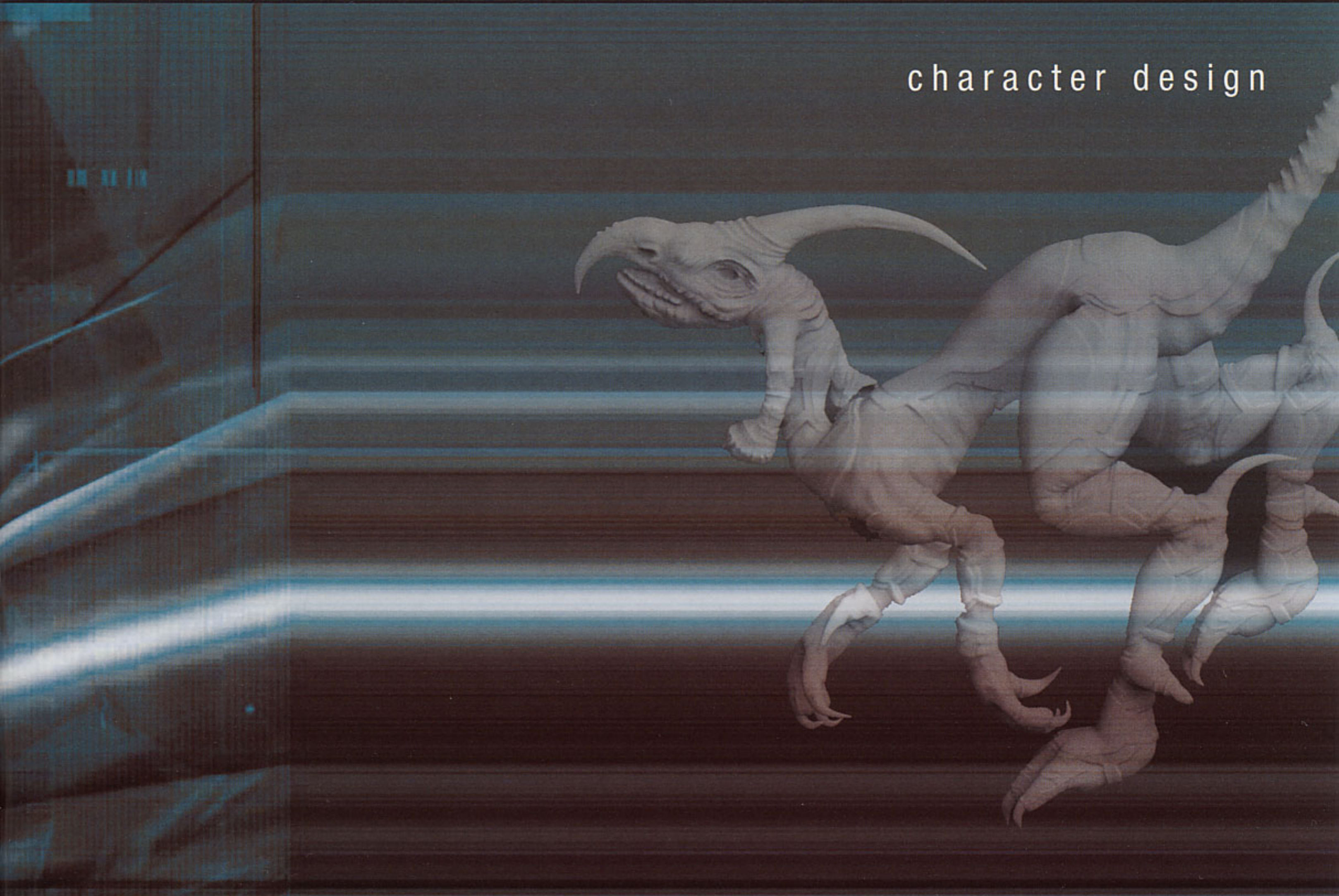
# *Character Design-Detail Management*

A review of the tools and techniques for preparing high-resolution digital characters

BY PAUL HOURMOUZIS







It's a well-known fact that any character animation work requires a geometry resolution that will provide the interactive speed and animation playback within your 3D application. Give an animator a mesh a few hundred thousand vertices to skin and animate, and more than likely, he'll throw it back at you.

Less well-known are some of the high-end tools and techniques available to artists that help speed up the process, from design and animation phases through to final output.

This article will review some of these techniques, as well as discuss the tools that are available to tackle a given problem.

But first, a recap of the usual techniques for modeling your digital character:

1. High-resolution laser scans, either from clay sculptures, miniatures or real-life actors: These scans typically produce very high-resolution geometry in the order of several hundred thousand, or possibly millions of vertices, depending on your subject. This approach is typically used for creating digital doubles (see article in issue 2), where sending your real life actor out of a plane without a parachute or off the edge of a plank into shark infested waters wouldn't be taken too favorably.

2. Digital sculpting and detailing, building up a character from scratch, using reference images as a guide, with NURBS, polygons or subdivision surfaces: A well thought out plan allows for a low-resolution, animator-friendly version of your mesh, which is subdivided at render time. For designs requiring finer details, you can then import your low-resolution model into a sculpting and detailing application such as Pixologic's ZBrush, and with a tablet, produce a highly detailed resolution mesh. The resulting character is beautifully detailed, but the process of detailing involves continual subdivision.





figure 1

Each of these two techniques results in geometry with high density that is typically a nightmare to skin and very slow to animate.

So, how do we get these highly detailed characters into a low-resolution form that is animation-friendly?

And what about when we go to render — how can we be sure our animated character will be represented in its glorious detail as intended, considering it was animated using a low-resolution version?

To answer these questions, we'll look at two alternative approaches.

**The first approach is based upon a commercially available product, CySlice, available from a company called headus, based in Australia.**

In a nutshell, CySlice allows you to convert high-resolution laser scan geometry, such as scans of real life actors and clay maquettes, into lower resolution industry-standard formats such as NURBS surfaces, polygons and subdivision surfaces based upon the Catmull-Clark variety. But its true value is in the way it captures details - CySlice conveniently allows you to capture the color and displacement information in your

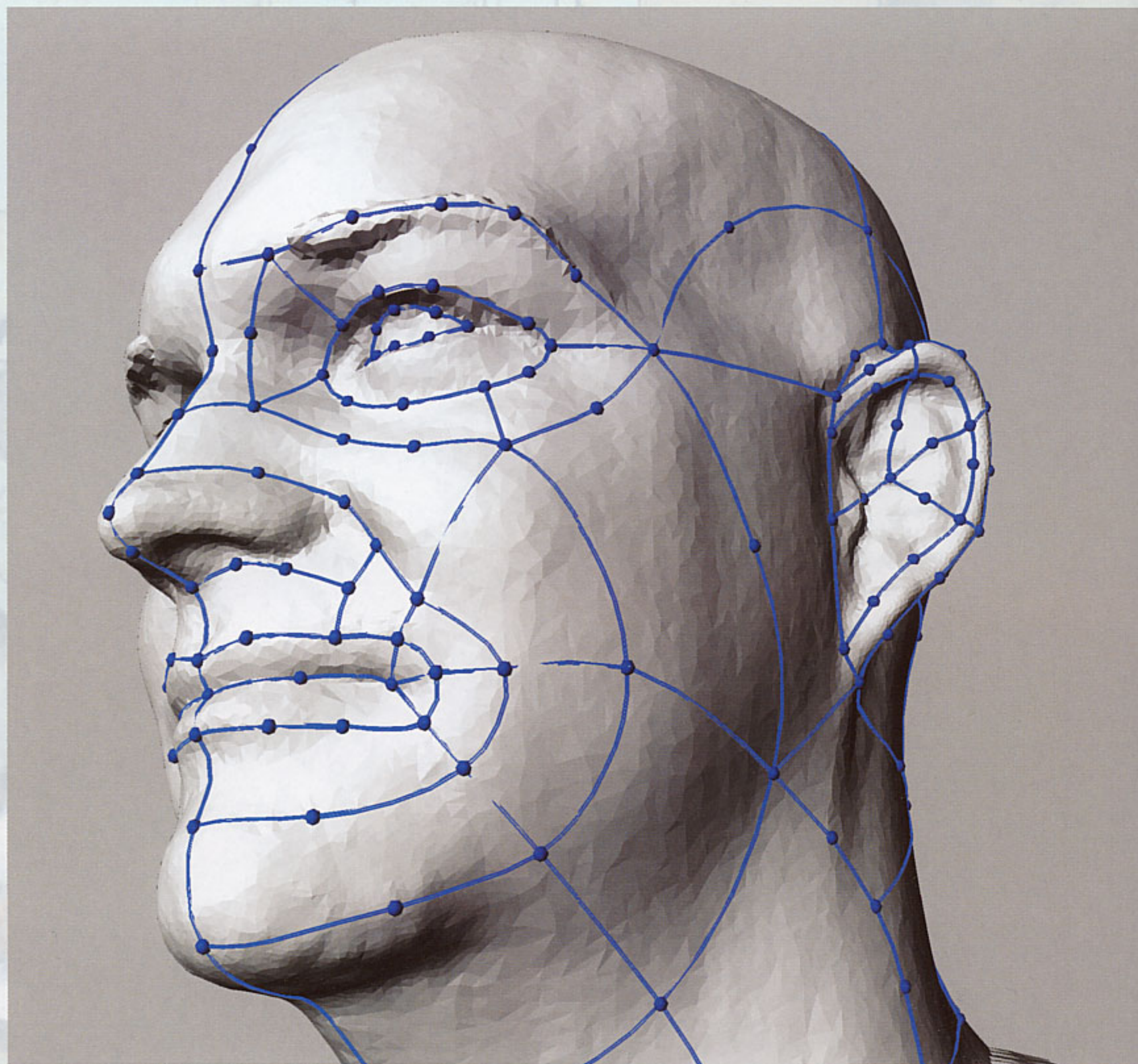


figure 2

scanned subject, ensuring a true representation in your final renderings. For the sake of simplicity, I will call this approach "Technique 1."

**The second approach is based on a technique developed primarily for the gaming industry, where details, in the form of displacement and normal maps, are "extracted or recovered" from a high-resolution character, then mapped onto a low-resolution version. A number of freely available tools are available for this technique. We will review a few, including Oliver Renouard's rayDisplace plug-in for Maya (available [www.highend3d.com](http://www.highend3d.com)), and video card manufacturer ATI's Normal Mapper program. I've named this approach "Technique 2."**

Although there are apparent similarities in each approach, the key difference is in the resolutions of your mesh — with the first approach, you begin with a high-resolution version and work on scaling it down. With the second approach, you begin with a low resolution and detail your character. Resolution-wise, they begin at opposite ends of a spectrum.

CySlice affords significantly more functionality and advantages over rayDisplace and Normal Mapper, especially if you don't have a low-resolution version of your character (which will almost always be the case if working from laser scans). Subsequently, this article will provide a detailed account of CySlice's functionali-



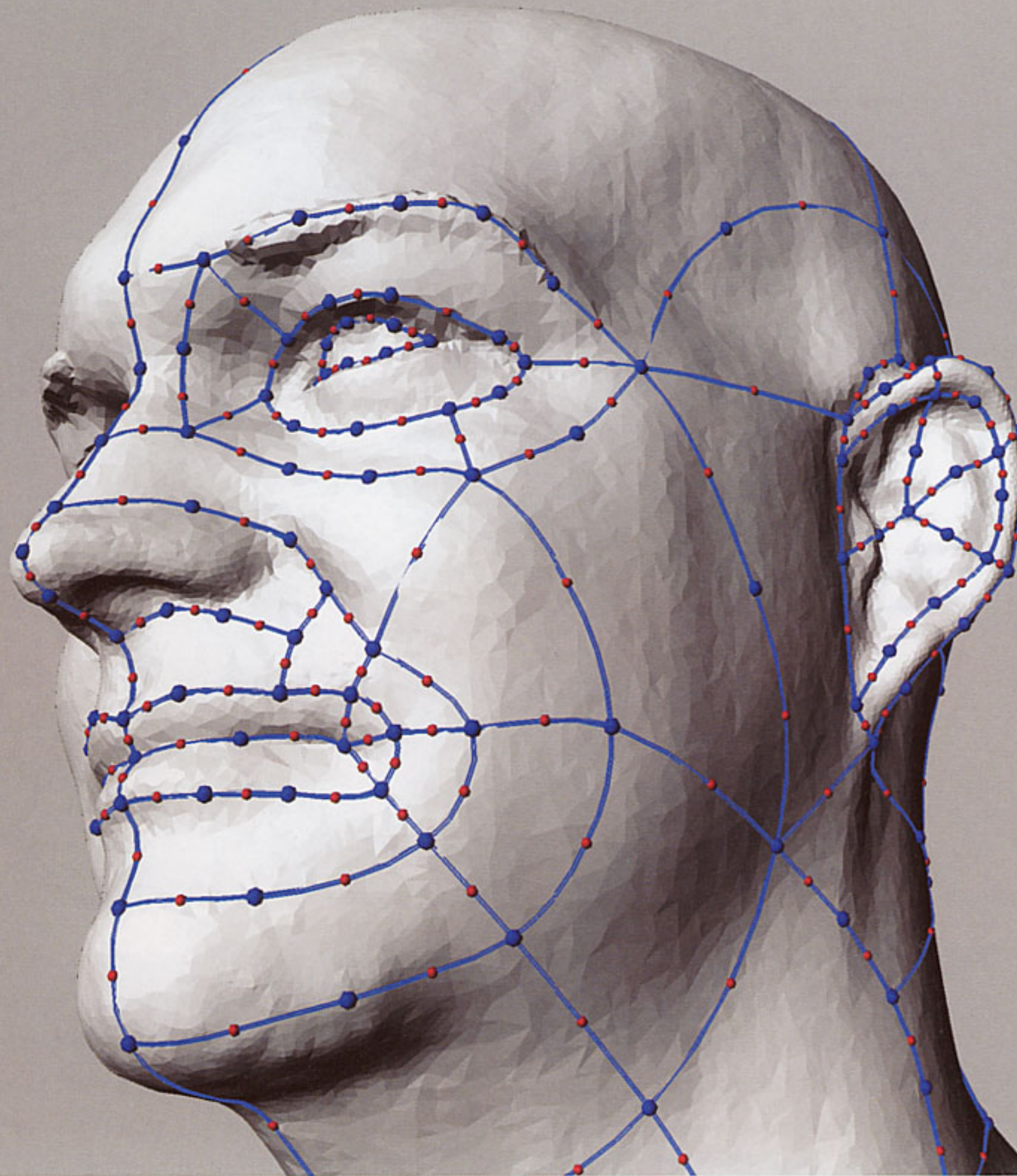


figure 3

Figure 1 shows a screenshot of the CySlice interface.

Be it intentional or otherwise, the user interface is ingeniously designed, as a top-down list of menus and functions within CySlice provides the user with a clandestine reminder of the sequence of required steps.

For example, the first is the Mesh menu, second the Display menu, third the Polymesh Edit menu, fourth the Networks menu and last, the Slicing menu (the slicing tool has actually been superseded with the Networks tools, but has been kept for users wanting to use it).

Each of these menus has sub-menus and actions you can perform, again, listed in top-down fashion to remind the user of the intended workflow within CySlice.

Beginning a session with CySlice starts by importing your object from the Mesh menu. You then adjust the display properties in the Display menu, alter mesh resolution with the tools found in the Polymesh Edit menu, then finally build a network with the tools found in the Networks menu. I cannot emphasize how much the logical ordering of the menu system helps in understanding the workflow.

Mtool, a free product that ships as part of CySlice, is the main conduit for getting your geometry into the format CySlice requires (PLY). It offers various geometry operations such as scaling, rotating and viewing your model. It also allows the export to various other formats such as IGES (for NURBS), OBJ for subdivision surfaces and standard polygons, as well as

## Overview of CySlice

CySlice was designed to manipulate high-resolution laser scanned objects, although recent advancements allow it to be used on meshes with lower density than a typical laser scan, broadening its appeal.

Upon starting CySlice, you are presented with a vastly different user interface that does away with the usual full screen display and typical menu system prevalent in today's applications. Rather, access to CySlice's interface and functions is by the way of a collapsible and floating menu system you can position anywhere on your display. Upon loading a model, a separate 3D window featuring your model is displayed.

ty and workflow. We will briefly review techniques for integrating rayDisplace and Normal Mapper into your pipeline.

## Technique 1 - Incorporating CySlice into your pipeline

This section will focus on reviewing the functionality of CySlice and how it typically integrates into a production pipeline. CySlice is a professional tool primarily aimed at the feature film market. Although its developer, headus, is listed as a Conductor under Alias Wavefront's partner program, CySlice can be used in conjunction with almost any 3D application.



DXF, 3DS and others, for use in your chosen 3D application.

Something to note is that CySlice performs best when your geometry is "real world" sized, so, if you are manipulating a human head, its dimensions should roughly be that of an actual human head.

## Building a network

As mentioned earlier, CySlice allows you to convert high-resolution geometry into a NURBS, polygon or subdivision surface, at the same time capturing the inherent detail in your high-resolution subject.

The process for doing so isn't as simple as pressing a button to convert; rather, it involves building a "network." Essentially, a network is a series of NURBS curves and patches fitted onto the surface of your high-resolution object.

The Network tool is the primary tool when working with CySlice. From the main menu, clicking the white right arrow to the right of the "Networks" label reveals its functions.

The process of creating a network is not too dissimilar to building a character from

NURBS curves and patches in a 3D application using image planes for reference. However, in CySlice, your reference object is in fact a 3D object from a laser scan, which you can pan, tumble and rotate in a 3D view as you edit the network.

Those who have built characters using a series of stitched NURBS patches would feel very much at home with this application, especially Maya users, who are well supported with an additional plug-in for moving data between the two applications. Also, keystrokes and combinations for navigation around your model are the same at Maya's default perspective viewport.

Building a network begins with placing points on your object. A series of points are interconnected to form a curve. For best results, the placement of points, and

hence shapes of the curves, should be based upon the main feature and flow lines of your subject. This is especially important if your subject is a head that needs to be animated, where an accurate typology of curves that follows main features and muscle lines results in a more convincing model for animating facial expressions.

Once these major flow lines are fitted, you can begin building the interconnecting curves between major feature and flow lines. The primary aim here is to complete the network ensuring that four-sided patches result. This ensures the cleanest possible mesh when exporting.

*Figure 2* shows a scanned head surface, with major feature and flow lines fitted to the surface using the Network tool.

Once this is done, you can define the number of subdivisions along each curve.

This may involve some experimentation; however, you have two strategies when selecting the subdivision spacing. If you

figure 4

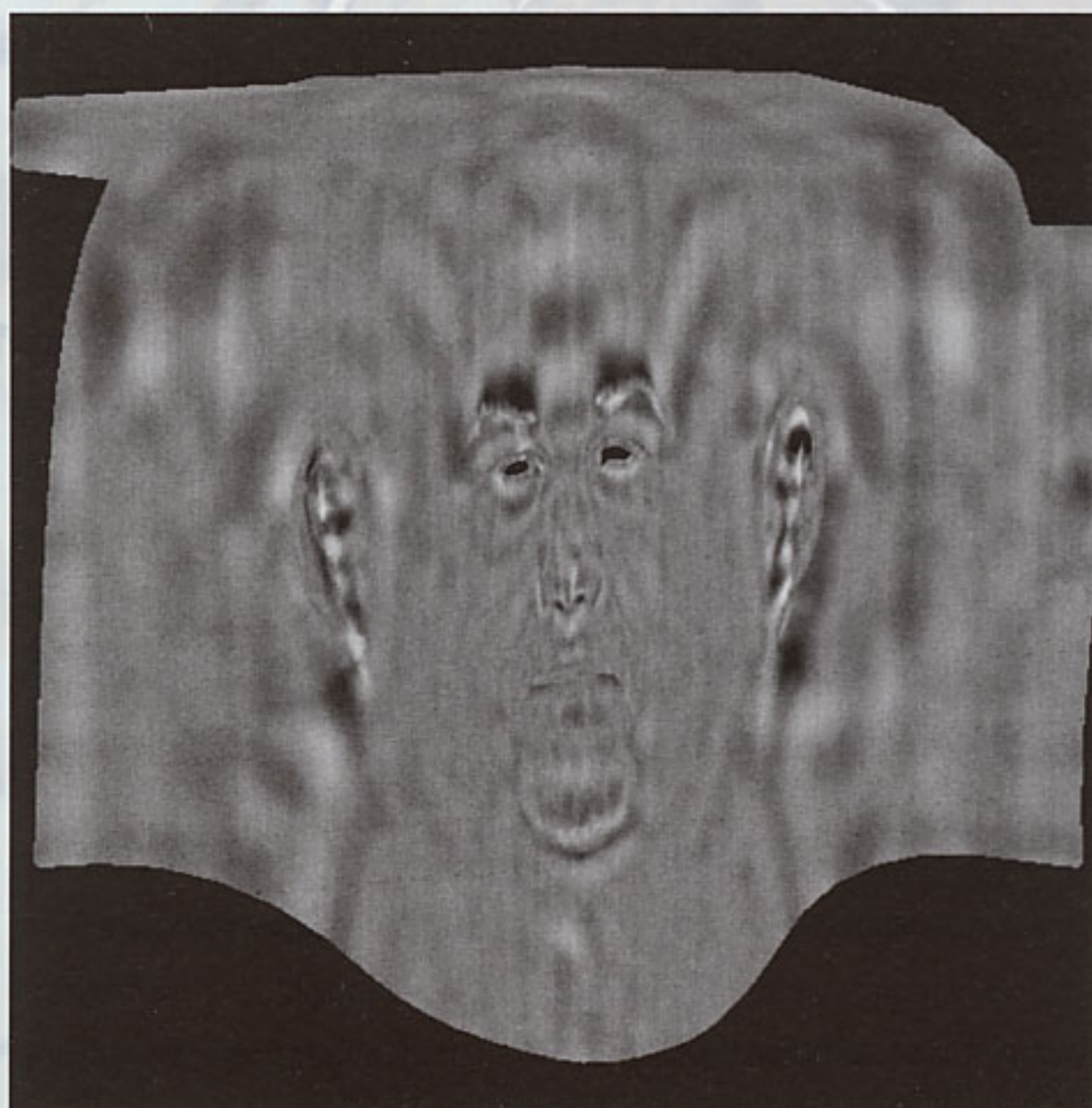
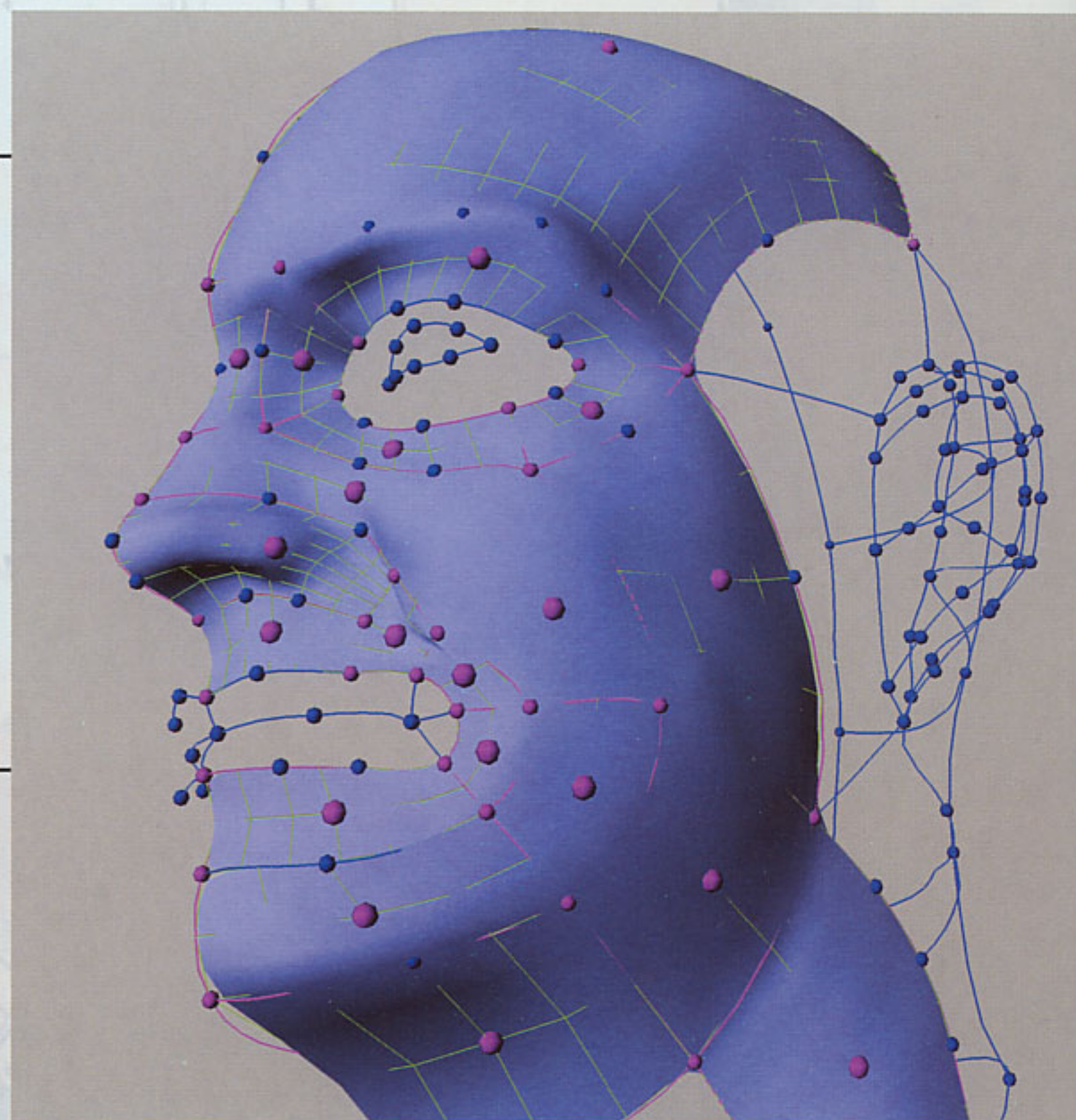


figure 5



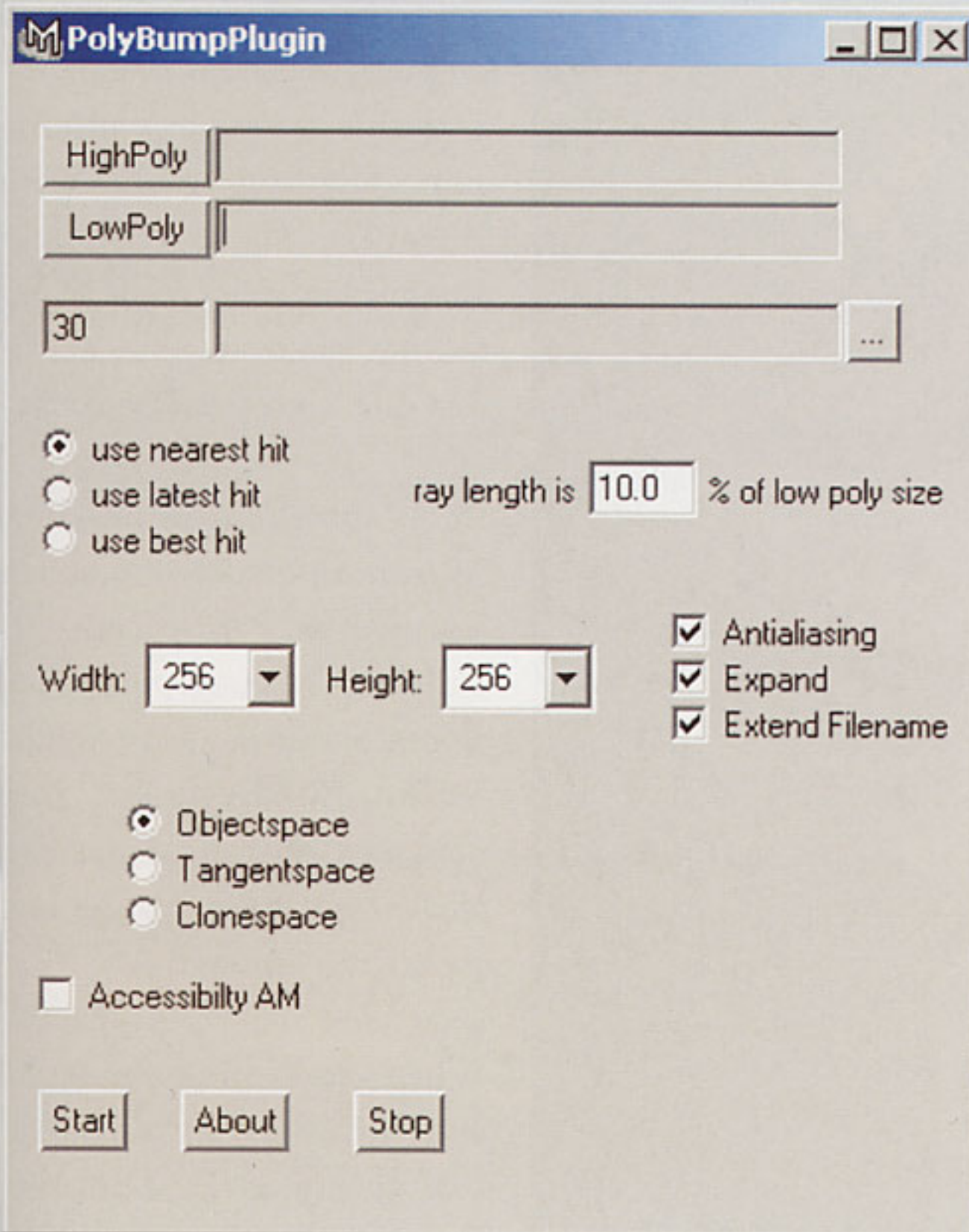


figure 6

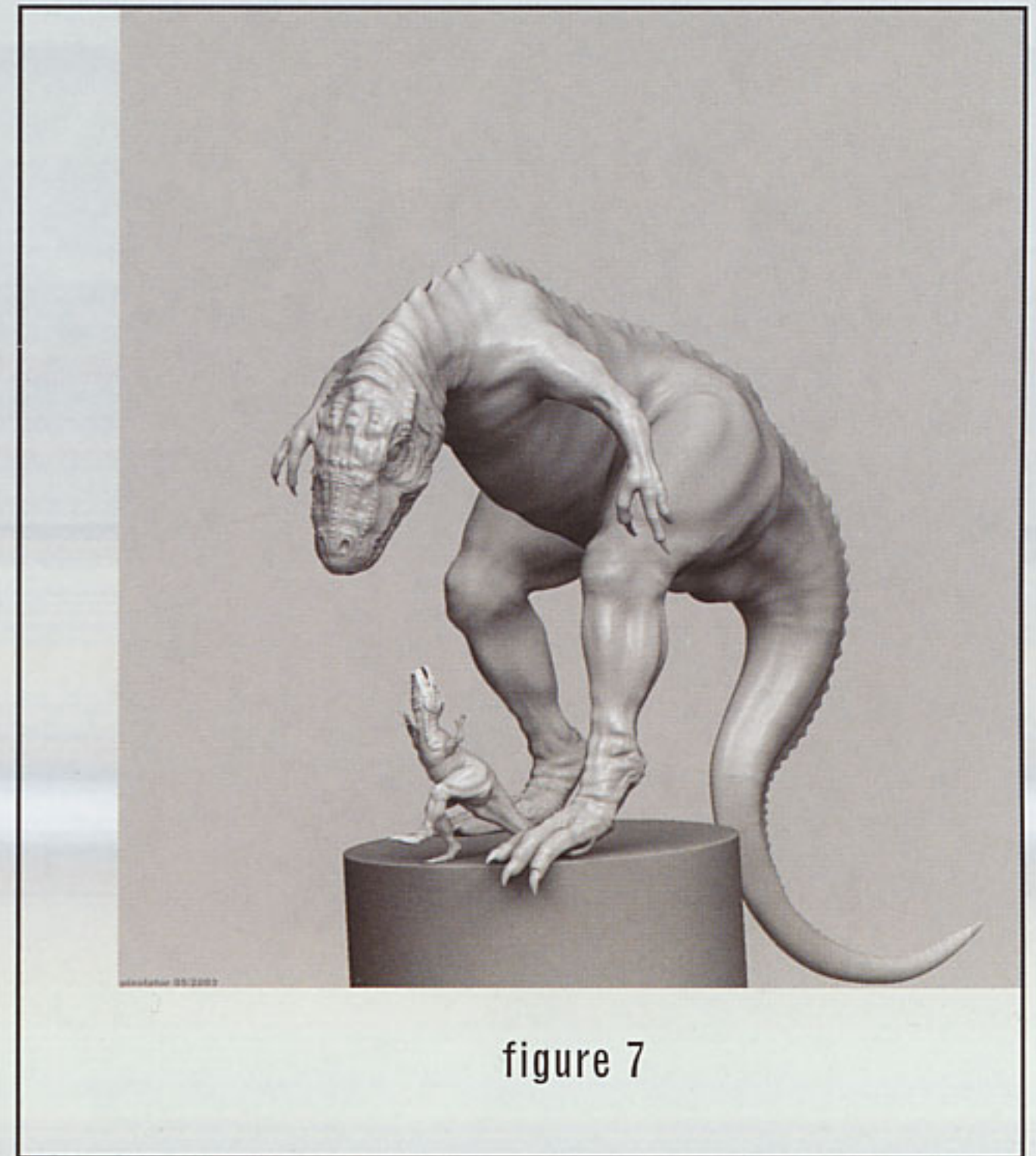


figure 7

intend on using displacement maps to define the detail of your model, use a low or moderate division factor. If you don't intend on using displacement mapping, choose a higher number that will capture the detail of your subject.

As our main aim is to provide a model that is animation friendly, we choose the former and utilize displacement mapping to generate the inherent detail of the model.

Figure 3 shows the completed network, with a division level of 2, which would be suitable for animation purposes. Division points are displayed in red.

After your network is defined, the process of creating the patches is next. Patches can be NURBS, subdivision surfaces or polygons. This process is highly iterative and involves placing a "seed point" with a boundary of curves. In essence, the seed points searches for a valid boundary of curves. If CySlice finds a valid boundary, then a patch is created.

Using a repeated sequence of key and mouse strokes, you define patches for each boundary set for your entire model. Any problems with the underlying network construction are identified and require rectification to proceed. Luckily, you can move network points throughout the process by simply switching editing modes.

Figure 4 shows the network with SubD patches being fitted. The larger pink points are the seed points that were placed to generate the patches.

Once patches are fitted for your model, you can then export to the format of your choice. If you fitted NURBS patches to your subject, then you will in fact be exporting a series of individual patches that will require stitching in your 3D application. Maya provides a number of stitching options, and I find that Global stitching works well, depending on how well your network was constructed. CySlice does its best to maintain tangency between patches — in fact, it does an exceptional job.

## Exporting Texture Maps

Here's where the real value of CySlice comes into play. After defining patches, you can now generate displacement, color and normal maps for your object. Unlike the highly interactive process involved in defining networks and patches, generating these maps is a matter of a few mouse clicks.

Figure 5 shows a displacement map generated for our SubD head.

This key feature allows you to render color, displacement and/or normal maps, which, when correctly used in your shading network, generate the detail of your subject at render time. This frees the animator to work with a low-resolution version of your character.

Using the tools found in the Texture Maps and Rendering menus, you can define map sizes, bit depth and various other accuracy settings when generating your texture maps. There are many other parameters and options available when rendering maps, most of which are beyond the realms



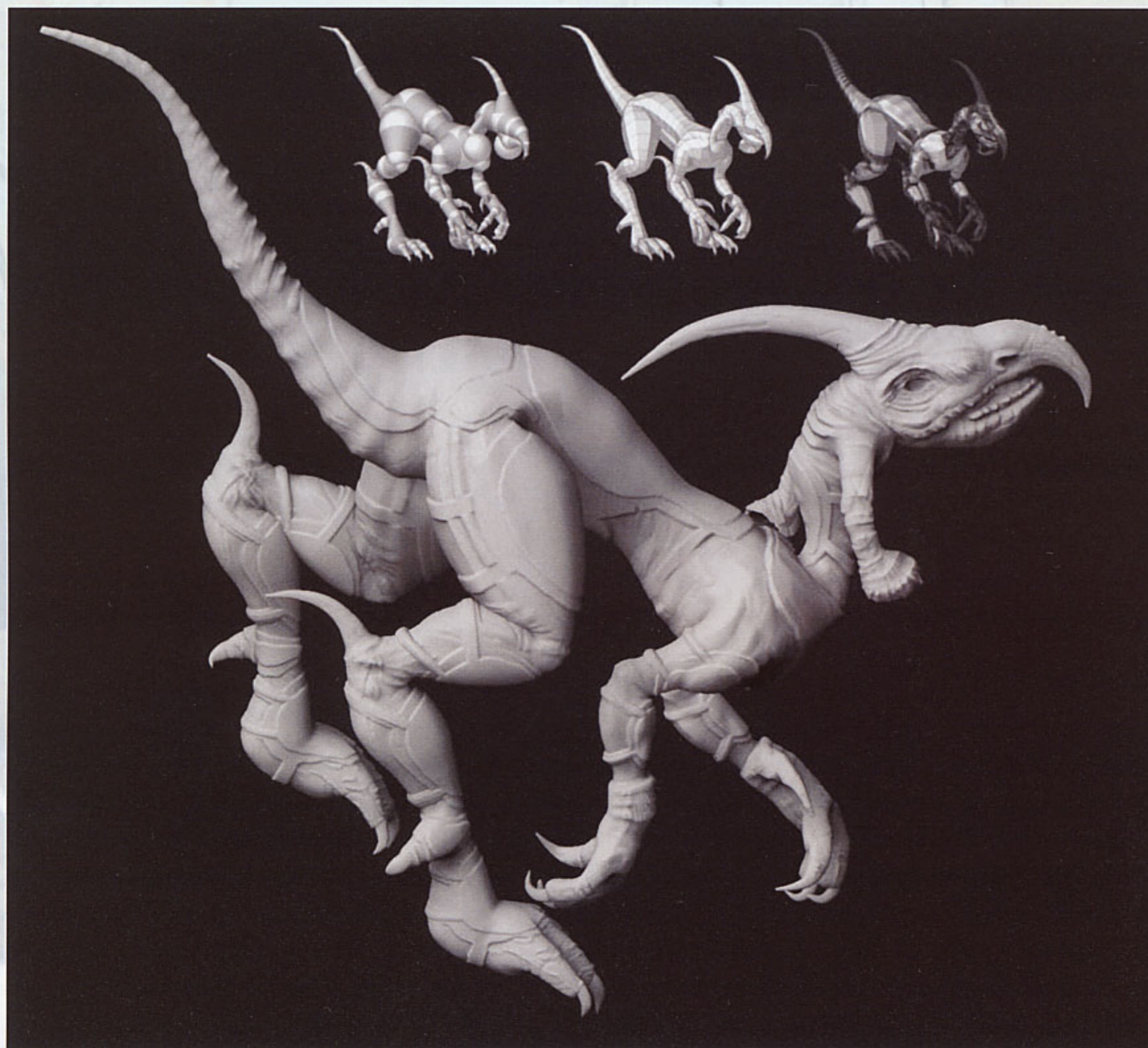


figure 8

of this article; however, it is sufficient to say the tool set is well and truly complete.

When generating textures maps for subdivision surfaces and polygons, UV's are neatly laid out in a tile fashion (very similar to ZBrush's UV tiles feature). Alternatively, you can use your own UV set. A special feature when rendering maps is the feature to "bleed" the displacement and normal mapping information across UV borders. This is a required feature for proper displacement and overcomes a sampling "issue" in most modern rendering systems.

When rendering maps for use on NURBS patches, you will produce an equal number of image files for attachment to your shaders. If you have 30 NURBS patches,

you receive 30 displacement maps. When rendering for NURBS, you have the option to generate MEL scripts within CySlice, that, when executed within Maya, completely automate the process of attaching the correct texture file to the appropriate patch. This is very helpful.

### The Result

When combining your new geometry created in CySlice with the rendered displacement and/or normal maps, the end result is a highly animate-able character that renders indiscernibly from that of your originally scanned object. The speeds at which today's digital doubles and characters are made are a reflection of this technology.

The other advantage of this technique is

that is allows you to utilize a product such as Pixologic's ZBrush as a serious sculpting and detailing tool. With ZBrush, a tremendous amount of detail can be added to your character, but at the cost of a high polygon count.

In a short space of time, ZBrush is quickly becoming accepted as a first class 3D sculpting package, allowing you to sculpt in a very natural fashion. WETA digital have even used ZBrush for detailing of their characters for the *Lord of the Rings* trilogy.

When used in conjunction with CySlice, you can subdivide your mesh in ZBrush to your heart's content, with little or no concern about typology at the sculpting stage, as the correct typology of your model is defined when developing the network within CySlice.

### Summary

To summarize, a typical workflow incorporating CySlice includes:

1. Obtain your high-resolution subject, be it from a laser scanning service or from detailing a model in an application such as ZBrush, then import into CySlice.
2. Build a network of curves that defines your character's detail along main feature lines and muscle groups.
3. Define a network division that will provide a low to moderate resolution model when exported.
4. Select the desired output type (NURBS, SubD or Polygons) and create patches for your model, then export the model.
5. Render out displacement, normal and/or color maps to define the detail for your object.
6. Attach the texture maps to your object in your favorite 3D application.
7. Animate your object using the low-resolu-



tion mesh.

8. Render out your animation - the result should resemble your original detailed version.

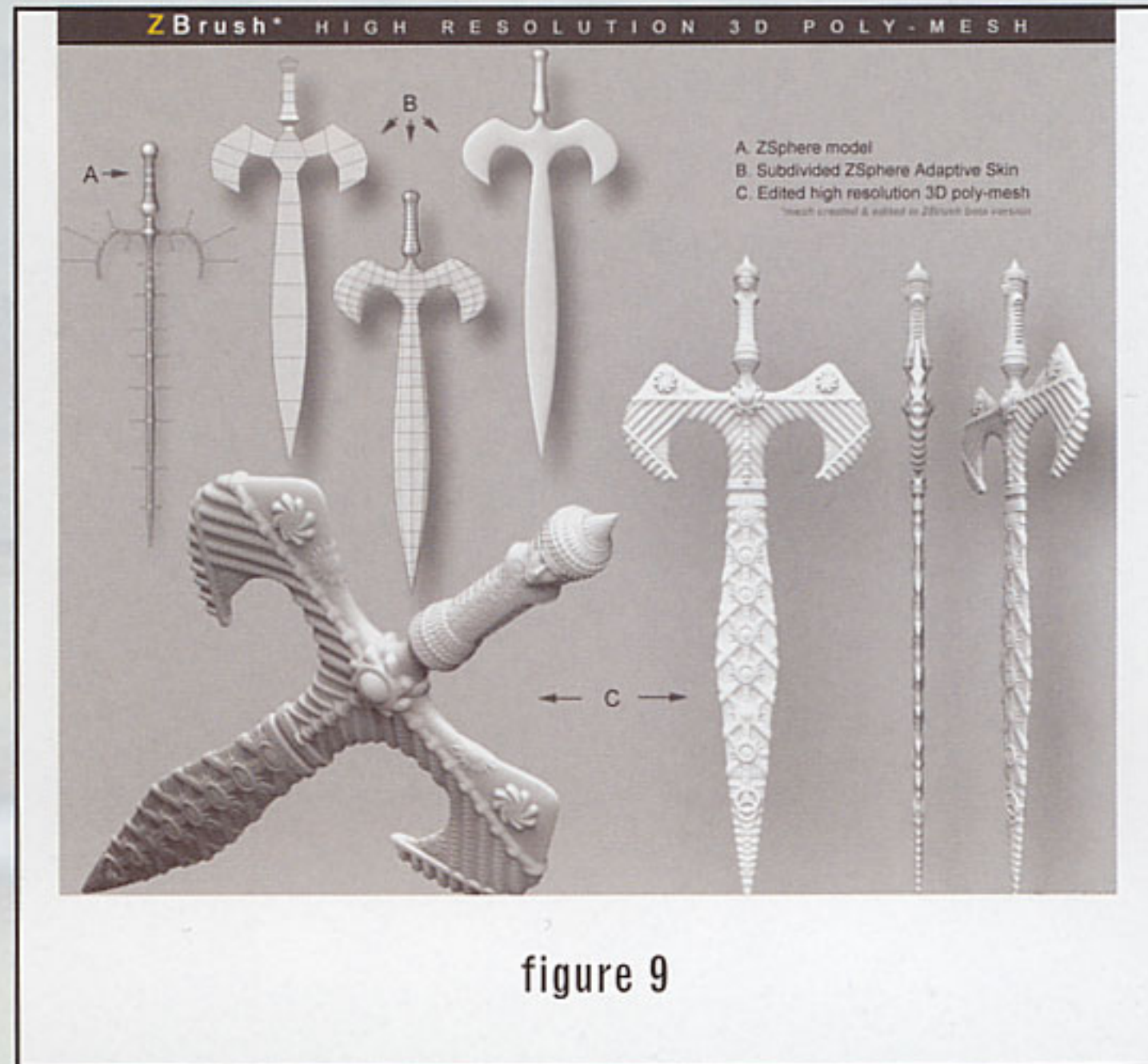


figure 9

**Technique 2 - Recovering detail from a high-resolution mesh.**

Another technique gaining momentum involves the recovery of detail from the differences between low and high-resolution polygon objects.

This technique is now commonly used in games, where support for real-time displacement and normal mapping is prevalent in today's leading video cards from companies such as ATI and NVidia.

In essence, this technique compares a low-resolution version against a subdivided and sculpted version, and, similar to CySlice, outputs displacement and normal maps. Typically, rays are cast from the surface of the low-resolution object, which then strike a point on the high-resolution object, providing the normal and displacement information, which is then converted to texture maps.

These maps can be applied to the low-resolution model that when rendered, provides a result similar to the high-resolution model.

There are a number of tools that allow you to do this, and the good news is most are freely available.

One of these is a Maya plug-in called rayDisplace, available from [www.highend3d.com](http://www.highend3d.com). By integrating a rayDisplace, displacement and normal maps are baked, then applied to your low-resolution object. See [www.highendmag.com/links/](http://www.highendmag.com/links/) for a

comprehensive tutorial, sample scenes and explanation of rayDisplace by its creator, Olivier Renouard.

For those not using Maya, a number of other options exist. ATI have a freely available command line utility called Normal Mapper, for which the complete source code and binary files are available from [www.highendmag.com/links/](http://www.highendmag.com/links/). The downloadable files include a comprehensive PowerPoint presentation on the theory behind Normal Mapper. Normal Mapper takes as input your low and high-resolution geometry files and creates both normal and displacement maps that again can be applied to your low-resolution geometry.

Being open source, Normal Mapper has been further developed — an alternative version of Normal Mapper is available [www.highendmag.com/links/](http://www.highendmag.com/links/). This version accepts standard OBJ files as input which was added recently.

And finally, another alternative is a commercial plug-in available for Maya called Polybump, available from Crytek [www.highendmag.com/links/](http://www.highendmag.com/links/). Using similar techniques, Polybump features a simple interface within Maya (see Figure

6) that allows the user to select your high and low-resolution geometry, set the appropriate parameters for map resolution, then render the maps. For users wanting simplicity, this may be the way to go. There's also a version for 3DS Max users.

## And now, the future:

Undoubtedly, new techniques for representing high-resolution geometry with their low-resolution cousins will surface.

One currently in development is by Pixologic Inc., the makers of ZBrush.

Although details are scarce at this stage, the next release of ZBrush will include an advanced displacement mapping feature. Some initial results being posted by its genius creator, Ofer Alon, are more than promising.

Figures 7, 8 and 9 show examples of ZBrush's upcoming displacement mapping technology.

See [www.highendmag.com/links/](http://www.highendmag.com/links/) for comprehensive discussion at Pixologic's ZBrush Central forum website.

The speed at which character designs are taken to final product could almost be considered proportional to the rate of these developing technologies.

Combine the tools and techniques highlighted with ever-increasing graphics power of today's machines, and photorealistic characters within our games, not to mention an ever-increasing quality of digital characters in film, are only our imagination away. 🍌



# *Artist Spotlight* *vincent thomas*



**Name:** Vincent Thomas

**Age:** 27

**Family Status:** Single

**City of Residence:** Paris

**Length of Time Doing Visual Effects:** 9 years / more than 5 years as a professional

**Past Workplaces:** BUF, "La Maison," The Mill, Mac Guff Ligne, Medialab...

**Email:** [vinthoma@imaginet.fr](mailto:vinthoma@imaginet.fr)

**Professional Status:** Freelance





### 1) What are your favorite movies?

I admire the perfectionist Kubrick, the enigmatic Lynch, the eccentric Terry Gilliam, the spiritual Woody Allen, old films or others like "The Hours," "Playtime," "Blade Runner," "My Neighbor Totoro," "Nostalgia".... Now, it's curious to see how my appreciation is changing with time. I suppose that's how you could recognize a cult movie: it could stand as its best against time.... It's also true for almost every art form.

### 2) What is your focus/position at your current job?

Well, while I'm more focused on the rendering/lighting side, I love to deal with many different kinds of tasks. So, most of the time, I'm responsible for a shoot, so I could model, matte paint with my favorite stylus, light and compose myself...

I suppose these things happen more in Paris studios than in big American places. I always have a global approach to image, in that I'm only concerned by the result. I'm a rounded artist, and I love that!

### 3) How do you conceptualize your work/ideas?

Being precise in your ideas without losing your concept is not easy, because when focusing on fixing minor problems, (most of the time, fighting with the computer) you could simply lose the essential. Then the

final result looks more like a profession of unjustified effects than an emotive vision. It seems obvious, but I have just "really" realized I must stay abstract and global even when working locally.

Well, in addition to my professional career as a computer artist, I'm experimenting with my own films. I love to write short fiction, especially dialogue. Every time a new idea comes to my mind, I write it on my carnet. You never know, you have to be patient, but it could be useful later : )

### 4) How do you cope with challenges and roadblocks when trying to create realistic images?

First thing is observation. Second is a reflection based on your observation, your technical knowledge, what you wanted to express in your image, and well ... the time available : ) Third is organization, focusing on only one thing at a time. You have to think about how the eye is looking at the image. What are the strong compositing masses? Do you think the effect will still look great to you when you watch it two weeks later? Sometimes, you leave your job proud, and the next morning, you can't even believe you have such a shitty work. Oh well, maybe that's just me : )

### 5) Do you feel satisfied with your results when attempting to create realistic images?

<continued>







Only for a few, I must admit. For half of them, I can see too many problems now to be proud:)

Now that that's been said, I'm not especially motivated to make realistic pictures, just to making rich and captivating pictures.

**6) What are your hobbies or favorite things to do when you're not working?**

Well, when I'm not working for a studio I'm working for myself ; ) Seriously now, I'm interested by music, reading, traveling and photography

**7) What was your dream when you started in 3D?**

I don't remember having a dream when I started. I think it was more just a creative

compulsion. Dreams came with maturity, and with my growing interest in writing, music and video editing. Without being original, authoring and directing little shorts are what I'm looking for in the next 20 years.

**8) What is your take on the current status of the 3D animation industry?**

It's a difficult industry as seen with every production job, where you can be extremely busy one day and looking for a job the next day. That's exciting and stressful at the same time. The job asks you for some sacrifice in your own life, and a constant investment to stay sharp with the new techniques. Before it was more a craft profession, now it's more like any industry.

**9) What advice would you give to all the college students, aspiring programmers and artists coming into the industry?**

Try to have a rounded education before being specialized. Don't be afraid to switch off the computer : )

**10) Do you have any philosophies that you live by, helping make you the person you are?**

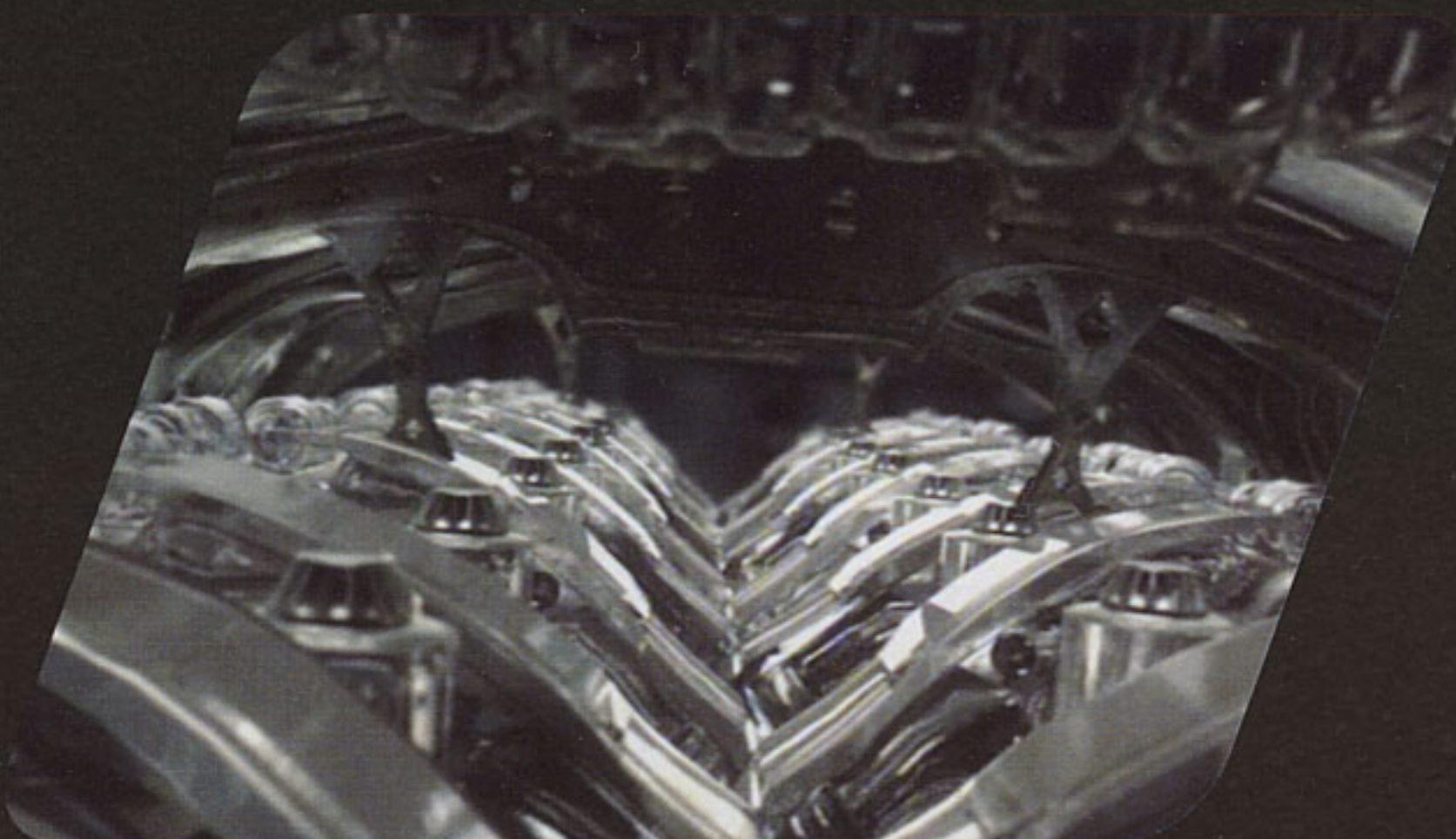
I have some, like this proverb from Pearl Buck: "The true wisdom of life is to see the extraordinary in the ordinary." Some others I will advise to anyone, but have some difficulties to apply to my own life ; )

**11) What would you like to see added to 3D software?**

Automatic UV mapping, true muscular systems, but more friendly oriented tools that are really useful in a production context. Oh wait.... May I mention a "make client happy" button? ; )

**12) What is your take on 3D actors in film?**

I'm a bit pessimistic on this. It will take time and anyway, would never be seen as essential as an actor is in my mind. Now, some authors build a character around an actor's personality. No doubt that with the new CG possibilities, some could make some very interesting applications of this. The charac-







ter "Gollum" in "Lord of the Rings: The Two Towers" was a good glimpse and "Final Fantasy" was the bad example.

### 13) Where do you find inspiration?

In any art forms or simply observation. Sometimes simply a few musical notes can inspire me, and with my imagination I can build something around them.

That's a completely unpredictable process. Sometimes unproductive, sometime explosive. One thing I have observed: Some conditions, like staying "out of the world" can help you to be creative and concentrated. Keep experimenting and try to stay as objective as possible.

### 14) How does being a photographer help you in your digital work?

Photography helps me to better understand how framing, the light and shadows give mood to a scene. And now my recording camera helps me to better understand the motivation of a director. All of this is a slow realization.





# SUBSCRIBE NOW!



# H I G H E N D

M A G A Z I N E

## HIGHEND MAGAZINE — STANDARD

- ☐ Highend USA 1 year (4 issues) ..... \$45.00
- ☐ Highend Canada/Mexico 1 year ..... \$50.00
- ☐ Highend All Other Countries 1 year .... \$50.00

## HIGHEND MAGAZINE — RUSH

- ☐ Highend USA 1 year (4 issues) ..... \$60.00
- ☐ Highend Canada/Mexico 1 year ..... \$65.00
- ☐ Highend All Other Countries 1 year .... \$50.00

## KEYFRAME MAGAZINE — STANDARD

- ☐ Keyframe USA 1 year (6 issues) ..... \$54.00
- ☐ Keyframe Canada/Mexico 1 year ..... \$72.00
- ☐ Keyframe All Other Countries 1 year... \$84.00

Keyframe Magazine, created for the LightWave community, offers a solid collection of professional-level tutorials, unbiased and in-depth product reviews, high-profile, industry case studies, and feature articles of lasting value.

## SUBSCRIPTION & RENEWAL FORM

Name \_\_\_\_\_

Company \_\_\_\_\_

Delivery Address\* \_\_\_\_\_

Phone \_\_\_\_\_ Fax \_\_\_\_\_

Email \_\_\_\_\_

\*If the Delivery Address is in Nevada, please include the 7.25% Nevada sales tax.

☐ Charge my card ☐ Visa ☐ MasterCard ☐ American Express ☐ DiscoverCard

Card # 

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|--|

Expiration Date \_\_\_\_ / \_\_\_\_

Cardholder Signature \_\_\_\_\_

Name on credit card (if different) \_\_\_\_\_

Credit card billing address (if different) \_\_\_\_\_

Cardholder email address (if different) \_\_\_\_\_

☐ I'm enclosing a check or money order in U.S. Dollars. (payable to **DMG Publishing**)

### ONLINE:

[www.dmgpublishing.com](http://www.dmgpublishing.com)

### FAX:

Fax 1-702-616-9647

### PHONE:

Tel 1-702-990-8656 or 1-888-778-9228

### MAIL:

#### DMG Publishing

2756 N Green Valley Pkwy # 261  
Henderson, NV 89014-2120 USA

### EMAIL:

[info@dmgpublishing.com](mailto:info@dmgpublishing.com)



# GNOMON

## WORKSHOP

Training DVDs, Tutorials and Tips

Modeling • Texturing • Lighting • Rendering • Animation  
Kinematics • Constraints • Dynamics • Skinning

The Gnomon Workshop exists to provide the Maya community with training DVDs and written tutorials. There are currently 38 DVD titles available for ordering.

- » Based on Gnomon School curriculum
- » Created by industry professionals
- » Production specific tips and techniques
- » 9 Maya areas of focus
- » Over 100 minutes of instruction each
- » Convenient online ordering

*"Gnomon has always been a great resource for our in-house artist development program... With our busy schedules, we often use these DVDs for individual artists to explore new ideas and quickly update their skills at their own pace"*

*Sande Scoredos, Director of Training  
Sony Pictures Imageworks*

Images © Copyright A. Alvarez, D. Krumweide



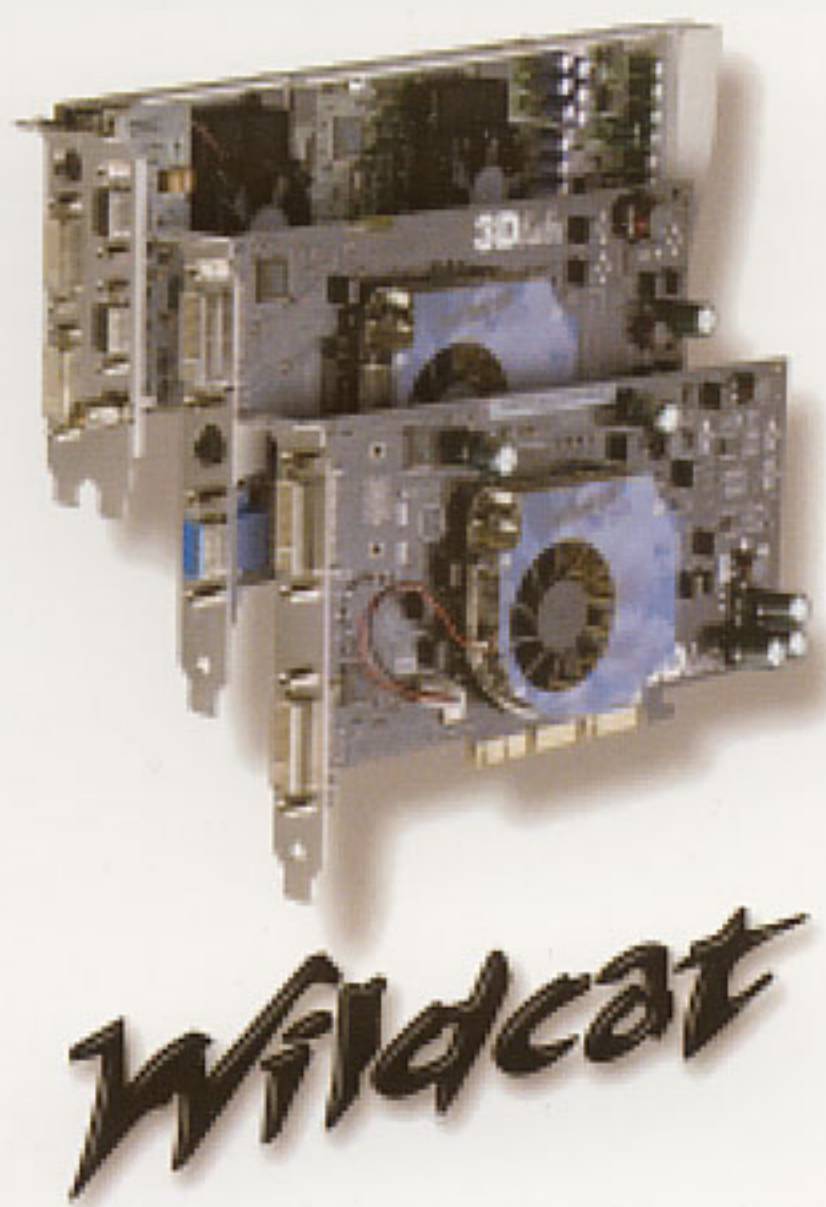
www.thegnomonworkshop.com



# GRAB A WILDCAT!



## ...and Unleash Your Creativity



When you've really got to attack that design, grab a Wildcat® from 3Dlabs®. Real-time rendering and proven performance let you unleash your creative beast. Dependability and agility help you gain ground on your schedule. So, no matter your budget or the complexity of your project, you can put the unmatched power of a Wildcat to work for you.

We have a full line of professional graphics accelerators to choose from:

- Wildcat VPs combine programmability and performance with affordability
- Wildcat4s put ultra high-end performance and functionality at your fingertips
- Wildcats offer custom drivers for select CAD and DCC applications

Visit [www.3dlabs.com/family](http://www.3dlabs.com/family) to see all your options.



A CREATIVE® Company