GNOMON SCHOOL PRESENTS

# HIGHEND MAGAZINE

ISSUE NUMBER 4 • WINTER 2004

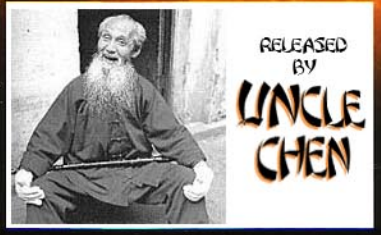
## SPIDER-MAN
THE ANIMATED SERIES

## Apple & Volkswagen
Not Farfrommakin'
Music to Your Ears

## Occlusion Tutorial

## Maya "Killer" Tips
Easy Step-by-Step

## DNA – Molecular Animation



# WELCOME TO ANOTHER INFORMATION-PACKED EDITION OF HIGHEND MAGAZINE

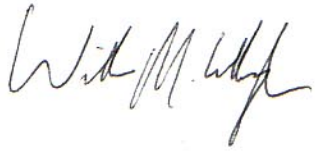
**Will McCullough**
Editor – Highend Magazine

Wishing all of you a great holiday season and hopefully some slowdown in work to spend time with your families. With four issues and year behind us, we feel grateful to have seen the kind of support and response we are getting from everyone out there. We are learning more with each issue, and developing our content to not only meet your requests but stay up-to-date with the trends in this digital artist world we support.

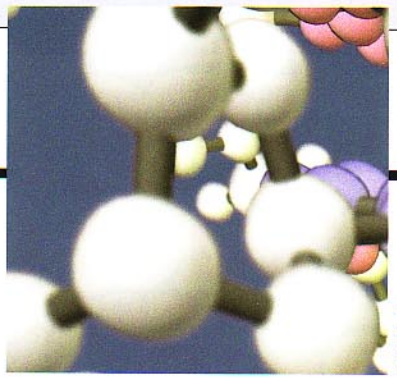
So far it has been an interesting journey to see the evolution of the tools we cover. A few years back when we came up with the idea for the magazine, the scope of our coverage was very limited. Now it has become apparent that we must cover a wider and wider range of not only tools, but, more interestingly, their application throughout the many industries they have evolved to be an essential toolset in. Now it is not only the world for animators and special effects artists. In this issue, you will see an example of how one artist works with macromolecular data in Maya, how a studio can create a fully animated toon-style Spider-Man series in 3D, and a follow-up article in character detail management that covers ZBrush, Maya, Renderman and mental ray. With additional images in this issue using Cinema 4D, XSI, 3ds max and Shade, it has become apparent that "Highend" is no longer the tool, but strictly the talent of the artist. From Industrial Design, Government satellite imaging, molecular data, film and TV special effects, to personal home production creating works of art – Highend magazine is here to give you the latest tips and techniques that you can apply to your own work.

While digital art and its tools and applications evolve, we will continue to as well. Trying to pack more information into each issue, trying to get more detailed and finding the best images and work to inspire and teach you. We are inspired with all the contributions, all the letters, and all the submissions we have been receiving. We thank all of you for helping make this the resource it has become. In this next year we look forward to more of you coming to us with your ideas, with your stories, and with your images, sharing your world with the rest of us. Write us at ideas@highendmag.com or visit our site at www.highendmag.com for more information on how to be in Highend Magazine!

To 2004 and a featured-filled Highend Magazine we say thank YOU!



# CONTENTS



PAGE 4

## FEATURE


**SPIDER-MAN CARTOON**
This animated series weaves a web of great characters
BY STEVEN WENDLAND
20

**VW & APPLE IPOD**
Music to your ears becomes candy to your iPod with a little help from VW
BY JOHN LAFAUCE JR.
26


## SPECIAL PREVIEW


**MAYA KILLER TIPS**
Take a trip through the author's newest book and get the latest "killer" tips on Maya
BY ERIC HANSON
32


## GALLERY


38

PAGE 20

PAGE 26

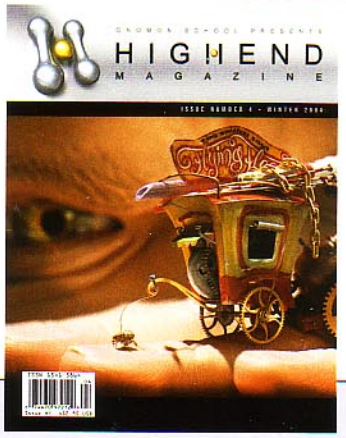
## TUTORIALS


**DNA**
Creating molecular animations in Maya and still allowing yourself a little artistic license
BY ERIC KELLER
4

**OCCLUSION**
Understanding occlusion passes and how they affect lighting. Setting up, rendering, and compositing them using XSI.
ANDREAS BYSTROM WITH RITCHIE L. ROBERTS
12

**CHARACTER DESIGN DETAIL MANAGEMENT PART II**
Part I in this series laid down the foundation of theory, while this part is a more detailed, tutorial-style look at the workflow.
BY PAUL HOURMOUZIS
44

**SIMPLE COLOR SUPPRESSION**
Building a simple macro in Shake.
BY MATT PLEC
52


**COVER ART:**
by Peter Fendrik


issue 4 • winter 2004

## CONTRIBUTORS

**ERIC HANSON** IS A VISUAL EFFECTS DESIGNER SPECIALIZING IN THE CREATION OF DIGITAL ENVIRONMENTS AND EFFECTS FOR FEATURE FILMS. HE HAS WORKED WITH VISUAL EFFECTS HOUSES SUCH AS DIGITAL DOMAIN, SONY IMAGEWORKS, DREAM QUEST IMAGES, AND WALT DISNEY FEATURE ANIMATION, HIS WORK CAN BE SEEN IN MANY FEATURE FILMS. ERIC IS ALSO AN ACTIVE INSTRUCTOR, HAVING LED COURSES ON ADVANCED DIGITAL EFFECTS TECHNIQUES AT SILICON STUDIO, GNOMON SCHOOL OF VFX, AND IS NOW LEADING A VISUAL EFFECTS CURRICULUM AT THE USC SCHOOL OF CINEMA/TV. ERIC IS A MEMBER OF THE VISUAL EFFECTS SOCIETY AND ATTENDED THE UNIVERSITY OF TEXAS AT AUSTIN. HE IS CURRENTLY AT DIGITAL DOMAIN AS A LEAD ON NEXT YEAR'S "THE DAY AFTER TOMORROW." HE WISHES HE COULD SLEEP MORE. FOR MORE INFO, VISIT HTTP://WWW.VISUALIMAGING.COM.

**ERIC KELLER** IS CURRENTLY A SCIENTIFIC ANIMATOR FOR THE HOWARD HUGHES MEDICAL INSTITUTE IN BETHESDA, MARYLAND. HE LIVES IN WASHINGTON DC AND HAS BEEN ANIMATING SINCE 1997. HIS FORMAL TRAINING IS IN CLASSICAL MUSIC, BUT HE SWITCHED TO CG SHORTLY AFTER RENDERING HIS FIRST CHROME SPHERE IN RAY DREAM STUDIO. HIS PREFERRED APPLICATION IS MAYA, BUT HE'S ALSO PROFICIENT IN LIGHTWAVE AND AFTER EFFECTS. HIS TRAINING IN SCIENCE THUS FAR HAS BEEN ON-THE-JOB. ERIC'S PORTFOLIO OF PERSONAL WORKS AND PROJECTS IS ONLINE AT WWW.KELLERRIFIC.COM.

**JOHN LAFAUCE** BEGAN HIS PROFESSIONAL CAREER IN 1991 MOONLIGHTING AS A PAGE FOR ABC TELEVISION BY DAY AND DIGITALLY COLORIZING CARTOON SHOWS LIKE "REN AND STIMPY" AT AN INK AND PAINT COMPANY AT NIGHT. HIS FIRST BREAK ON A FEATURE FILM CAME IN 1994 DREAMQUEST IMAGES HIRED HIM TO ROTO NATURE EFFECTS FOR "THE MASK," WHICH HE ACCOMPLISHED BY PAINTING CELS TRADITIONALLY AND ALSO DIGITALLY USING PHOTOSHOP. HE HAS SINCE GONE ON TO WORK AT SUCH FACILITIES AS SONY PICTURES IMAGEWORKS, RHYTHM AND HUES, PACIFIC TITLE DIGITAL AND CINESITE. HIS PROJECTS INCLUDE COMPOSITING ON OVER TWENTY-FIVE FEATURE FILMS. HE IS TEACHING "THE ART OF COMPOSITING WITH SHAKE" AT GNOMON AND CONTINUES TO FREELANCE. JOHN IS AN ACTIVE MEMBER OF THE VISUAL EFFECTS SOCIETY AND WAS RECENTLY SELECTED TO BE A JUDGE ON THE "BEST COMPOSITING" NOMINATION PANELS FOR THE FIRST ANNUAL VISUAL EFFECTS SOCIETY AWARDS.

**MATT PLEC** WAS THE PRIMARY SHAKE TECHNICAL SUPPORT POINT-OF-CONTACT FOR NEARLY THREE YEARS AND IS CURRENTLY RESPONSIBLE FOR 2D TECHNOLOGY AT ESC ENTERTAINMENT IN ALAMEDA, CALIFORNIA. HE CAN BE REACHED VIA EMAIL SENT TO MPLEC@DEEPMAJIK.COM.

**ANDREAS BYSTRÖM** IS A FREELANCE CG ARTIST FROM GOTHENBURG IN SWEDEN. HE FOCUSES MOSTLY ON THE MODELING PART OF CG, BUT IS ALSO VERY INTERESTED IN TEXTURING, LIGHTING, COMPOSITING AND RENDERING. ANDREAS STARTED OUT DOING CG ABOUT THREE YEARS AGO – BACK THEN HE USED 3D STUDIO MAX. ABOUT A YEAR LATER HE MIGRATED TO MAYA, USED IT FOR A FEW MONTHS AND THEN MOVED OVER TO XSI. TODAY, XSI IS THE ONLY 3D APPLICATION HE USES. AS SO MANY OTHERS, HE ALSO USES PHOTOSHOP TO PAINT TEXTURES; FOR COMPOSITING, THE FX TREE IN XSI OR AFTER EFFECTS STANDS UP TO THE TASK.

**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

**STEVEN WENDLAND** STUDIED MUSIC AT THE UNIVERSITY OF REGINA BEFORE ENROLLING IN THE SIMON FRASER UNIVERSITY FILM PROGRAM IN VANCOUVER. AN AWARD-WINNING SHORT FILMMAKER, STEVEN'S FILMS HAVE BEEN SCREENED AT MONTREAL, TORONTO AND OTHER FILM FESTIVALS AROUND THE WORLD. HIS EMPHASIS ON CINEMATOGRAPHY TURNED TO A FOCUS ON VISUAL EFFECTS. FOLLOWING A BRIEF PERIOD IN THE VISUAL EFFECTS DEPARTMENT OF THE TELEVISION SERIES "STARGATE SG-1," STEVEN JOINED MAINFRAME IN 1998. SINCE THEN HE HAS WORKED ON A TOTAL OF SIX ANIMATED TELEVISION SERIES, INCLUDING "TRANSFORMERS," "SHADOW RAIDERS" AND "HEAVY GEAR." HE HAS JUST COMPLETED PRODUCING THE FIRST SEASON OF THE HIT MTV TELEVISION SERIES "SPIDER-MAN" THAT MAINFRAME IS ANIMATING FOR SONY PICTURES TELEVISION / MTV.

**Writers** — Andreas Bystrom, Eric Hanson, Paul Hourmouzis, Eric Keller, John Lafauce, Jr., Matt Plec, Steven Wendland

CALL FOR WRITERS

If you think you are qualified to write for Highend Magazine, e-mail us at: **writers@highendmag.com**


GNOMON SCHOOL PRESENTS

# HIGHEND MAGAZINE

ISSUE NUMBER 4 • WINTER 2004



**Publishers**

CHARLES A. EDGIN charles@dmgpublishing.com
ALICE M. EDGIN alice@dmgpublishing.com

**Editors**

WILL MCCULLOUGH will@highendmag.com
ALEX ALVAREZ alex@highendmag.com
DARRIN KRUMWEIDE darrin@highendmag.com

**Managing Editor**

AMBER GODDARD amberg@dmgpublishing.com

**Technical Editors**

ALEX ALVAREZ alex@gnomon3d.com
DARRIN KRUMWEIDE darrin@gnomon3d.com

**Art Director**

CHRISTINA WILKINSON chris@sabredesign.net

**Design/Layout**

Sabre Design artists@sabredesign.net

**Highend Logo Design**

DMITRY SAVINOFF dmitry@savinoff.com

**ADVERTISING SALES**
Email: advertising@dmgpublishing.com
1-702-990-8656

**SUBSCRIPTIONS & BACK ISSUES**
Subscription information is on Page 56
Email: subscriptions@dmgpublishing.com
1-702-990-8656 or 1-888-778-9283



**HIGHEND MAGAZINE**
(ISSN 1536-3112)
is published quarterly by
DMG Publishing
2756 N Green Valley Pkwy #261
Henderson NV 89014 USA
Tel 1-702-990-8656 • Fax 1-702-616-9647

# Working With Macromolecular Data in Maya: DNA

BY ERIC KELLER

SCIENTIFIC ANIMATION IS A GREAT SOURCE OF WORK, AND ALSO A GREAT OPPORTUNITY TO FLEX YOUR CREATIVE MUSCLE. IT'S OFTEN A CHALLENGE TO CREATE SOMETHING THAT IS SIMULTANEOUSLY INSTRUCTIVE, VISUALLY ENGAGING, AND SCIENTIFICALLY ACCURATE. HOWEVER, ACCURACY IN SCIENTIFIC ANIMATIONS IS A TRICKY THING. MORE OFTEN THAN NOT, ACCURACY MEANS CREATING A NON-MISLEADING VISUAL METAPHOR RATHER THAN A PERFECT ATOM-FOR-ATOM REPRESENTATION. MANY TIMES THIS TRANSLATES INTO SACRIFICING REALISM FOR VISUAL AND CONCEPTUAL CLARITY.

**That being said, when the situation does call for accuracy in the form of a molecular model, it's good to know that you can achieve this in Maya fairly painlessly, with the help of a few scripts, some free applications, and enough artistic license as the situation will allow.**

This tutorial is broken down into several mini-tutorials designed to give you some techniques to build on when bringing macromolecular data into Maya. Specifically, we'll be working with the ever-popular, and often misrepresented, DNA. Of course these techniques can also be used on any of the thousands of molecules and proteins whose structure is freely available online at the Protein Data Bank website (www.rcsb.org/pdb/).

Most of the time the DNA depicted in movies, on TV, and even in publications, is wrong. DNA is often portrayed as a simple twisted ladder. Other times its structure is shown twisting to the left instead of to the right, or it has too many or too few nucleotides per bend. Sometimes it is shown with some horrible combination of all three mistakes.

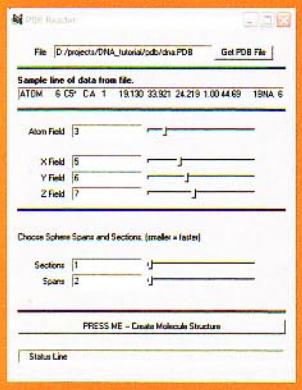



figure 1

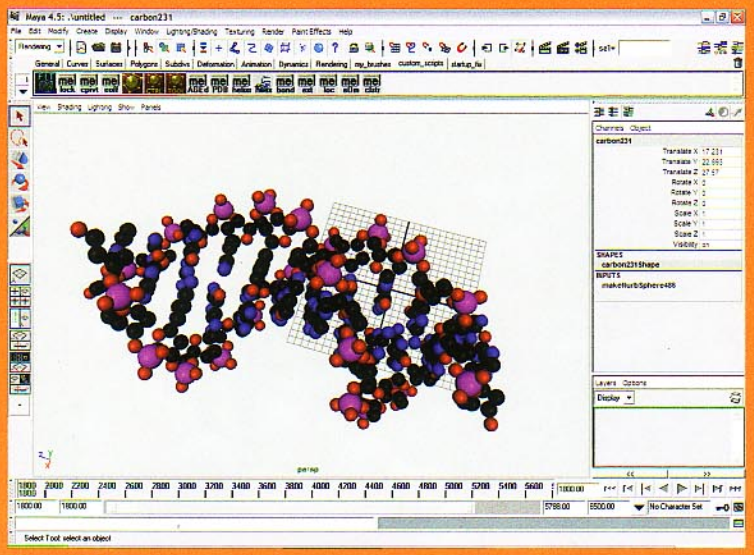


figure 2

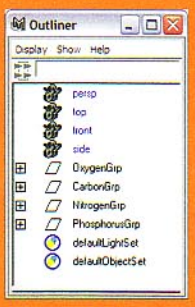



figure 3

The structure of DNA is very important – so important, in fact, that it earned James Watson and Francis Crick a Nobel Prize for discovering it. As an animator, you are responsible for correctly representing DNA's structure; fortunately, this doesn't require a Ph.D. By bringing in the actual DNA crystallography data from the protein databank, you can start out with the real McCoy and be assured that your final DNA animation should turn out to be correct.

OK, let's get to it...

Here's what you need to complete this tutorial:

1) Maya 4.0+ any platform (I believe any platform should work, and I've tested this on NT, XP and OSX)

2) These MEL scripts:

- pdbReader.mel v1.4 by Tom Doeden (download from www.highend3d.com Maya>Mel scripts>modeling)

- jPivToParticle.mel by Julian Mann (download from www.highend3d.com Maya>Mel scripts>dynamics/FX)

- particleDeformationPoly.mel by Alex Bigott (download from www.highend3d.com Maya>Mel scripts>dynamics/FX)

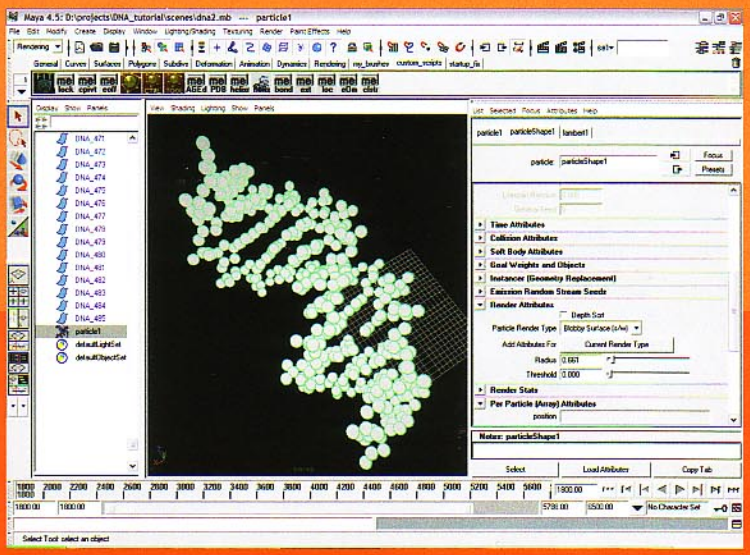


figure 4

- bond.mel and extrudOmatic.mel by me, Eric Keller, available on my site at www.kellerrific.com/scripts.html

3) One of these (free) programs:

- Chime (www.mdl.com/downloads/downloadable/)
- RasMol (www.umass.edu/microbio/rasmol/)
- Protein Explorer (http://molvis.sdsc.edu/protexpl/frntdoor.htm)
- RasMac (http://mc2.cchem.berkeley.edu/Rasmol/)
- PyMol9 (http://pymol.sourceforge.net).

4) A PDB file of DNA which can be downloaded from this address: www.kellerrific.com/tutorials/dna.pdb

5) These websites:

- www.rcsb.org/pdb/
- www.highend3d.com

6) And an intermediate level understanding of Maya, especially regarding installing and using MEL scripts and some experience with dynamics. Knowing how to write MEL scripts is not necessary.

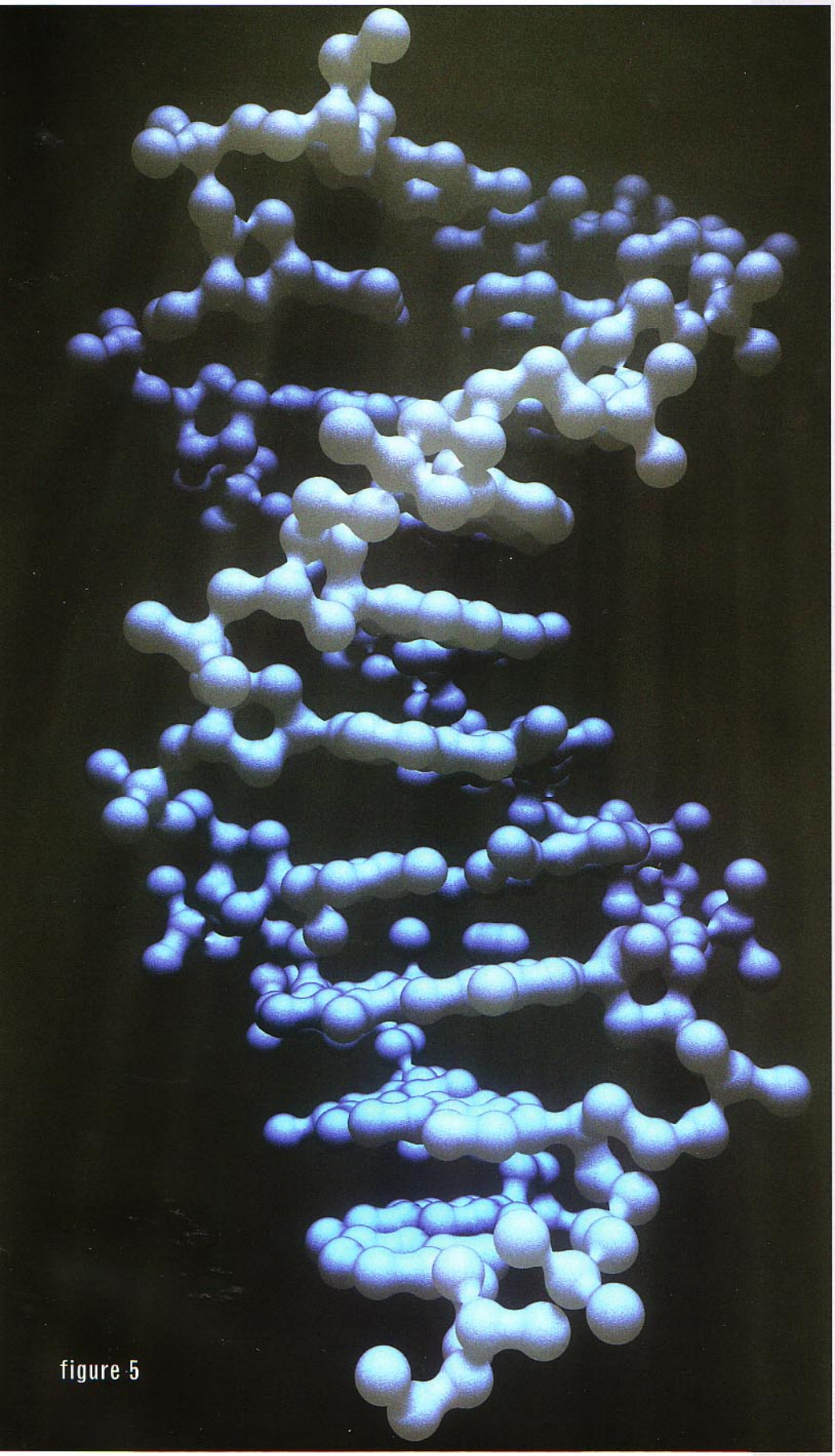


figure 5

## Part One: Getting Macromolecular Data into Maya

I am NOT a scientist, but my work puts me in a position in which there's almost always a scientist nearby. And that's great when I have questions or need to have something checked out. You'll definitely want to find a knowledgeable person to proofread the structure of your molecule, and it would be even better if he or she can help you find the molecular structure you need in the protein databank.

So where do you get this data? The protein data bank located at www.rcsb.org/pdb/ is an online database populated with thousands of molecules (22,516 at last count). These files are uploaded by scientists and are accessible to anyone. The files contain the actual crystallography data (meaning they contain the positions of the atoms in the molecules). The files themselves are just text files with comments and coordinates. Most often, scientists use a program like Chime, RasMol, or Protein Explorer to view these structures. Thanks to a handy script by Tom Doeden called pdbReader.mel, you can easily import the data into Maya.

Using the script is easy. However, before I get to that, let me give a few words of advice on using the protein databank. It's a good idea to know what exactly you're looking for before downloading the files. Plugging in a general term like "DNA" in the search field on the website is going to generate hundreds of results, many of which include molecules and data you may not want or need. This is true for any protein structure you may look for. The best advice I can give is to get some help from a scientist on your search. For instance, if you're working for a scientist, make sure he or she is very specific in what he or she wants. Furthermore, the databank has a lot of stuff, but it doesn't have everything. Some molecules haven't been structurally analyzed and entered in the database yet.

The databank website also has a great feature called "Molecule of the Month" that is definitely worth a read. The archives contain articles on DNA, bacteriaphage, p53, and many of your other favorites. (I found the article on DNA to be particularly useful.) In-depth information on what a PDB file is can be found on the site at www.rcsb.org/pdb/info.html.

It's also a good idea to get one of the free programs used to view the PDBs (Chime, RasMol, RasMac or PyMol9). I use PyMol on a Mac to both view the structure (to verify what I bring into Maya) and to strip out unnecessary data like comments (I just save the file under a new name and it often automatically removes extra code). Chime allows you to view the structure into a web browser, like Internet Explorer.

To make this tutorial easier to follow, I have a PDB file of DNA available at my site (www.kellerrific.com/tutorials/dna.pdb). Please feel free to download it. I've already stripped out the comments so you can bring it in to Maya without doing anything extra.

Okay, let's get to it. Open the DNA PDB file in a text editor and look at how the file is arranged. Some typical rows from our PDB file look like this:

```
ATOM   21 P    G A  2   22.409
31.286 21.483 1.00 58.85   1BNA 83
ATOM   22 O1P  G A  2   21.822
31.459 20.139 1.00 78.33   1BNA 84
ATOM   23 O2P  G A  2   23.536
32.157 21.851 1.00 57.82   1BNA 85
ATOM   24 O5*  G A  2   22.840
29.751 21.498 1.00 40.36   1BNA 86
ATOM   25 C5*  G A  2   23.543
29.175 22.594 1.00 47.19   1BNA 87
ATOM   26 C4*  G A  2   23.494
27.709 22.279 1.00 47.81   1BNA 88
ATOM   27 O4*  G A  2   22.193
27.252 22.674 1.00 38.76   1BNA 89
ATOM   28 C3*  G A  2   23.693
27.325 20.807 1.00 28.58   1BNA 90
ATOM   29 O3*  G A  2   24.723
26.320 20.653 1.00 40.44   1BNA 91
```

Each row in the file corresponds to a single atom in the structure. All we're really interested in is the atom's symbol and its X, Y, and Z coordinates. This is what PDB Reader translates into coordinates for the spheres it will create when you run the script. The atomic symbol (C for carbon, H for hydrogen, N for nitrogen, P for Phosphorus, and O for oxygen) determines which shader will be applied to these spheres and their relative size. The rest of the data is ignored.

**Tip 1:** What does "C3*" mean? In this case, the author of the PDB file has put an asterisk next to the atoms representing the backbone of the double helix. The numbers are merely a reference which pdbReader.mel ignores. If you strip the rows with the asterisks out of the PDB file, along with the rows containing P, O1P and O2P (a repeating Phosphorous with two oxygens), you'll be left with just the base pairs of the DNA model. Save these out to a new PDB file if you decide you want to do something special with them later. Remember to also make reciprocal PDB files of the data you've stripped out so you can have PDB files with just the backbone and no base pairs.

**Tip 2:** I have found that some atomic symbols, like Si (for silicon), will cause the script to choke. When this happens, I edit the PDB file itself and replace the atoms with an atomic symbol I know it will accept but doesn't occur too often in the molecule (like phosphorus) and then I change these phosphorous molecules back to whatever atom I need once the file has been loaded. It's an ugly workaround but it gets the job done.

It's important to know that all PDB files are not formatted exactly the same way and that sometimes these numbers appear in different columns.

Run the pdbReader.mel script in Maya and click the "Get PDB File" button. Browse to find the dna.pdb file you downloaded and select the file.

fig 1. The pdbReader.mel interface with the DNA pdb file loaded

The interface has a sample row from the PDB you're loading. Below this are fields for you to enter the atomic symbol and the X, Y and Z coordinates of the atoms in the file. There are default numbers in these fields, but you may have to change them depending on how the particular PDB file you're using has been written. It's safe to assume that the default numbers are NOT correct for your file. The simplest way to figure out what to put into these fields is to count over from the left in the example row and plug in the appropriate coordinates. So our example would look like this:

```
ATOM   5 O5* C A 1   18.935 34.195
25.617 1.00 64.35  1BNA 6 1    2 3  4
5 6    7    8    9   10  11   12 13
```

Weird spacing can make it a bit tricky to tell where one column ends and the next begins. Be prepared to try this a few times before you get it right.

In this example row, the atom is column 3 (O for oxygen), and the X, Y, and Z coordinates are columns 7, 8 and 9, respectively.

How do I know that the atom is in column 3 and not column 4? After all, "C" could stand for carbon. In this case, I've looked at the rest of the PDB file and noticed that column 4 contains ONLY "C," "G," "A," or "T." Since this is DNA we're dealing with, I surmised that the author of the file is using these symbols to indicate base pairs (Cytosine, Guanine, Adenine and Thymine, respectively) and not atomic symbols. Therefore, I guessed that O5*

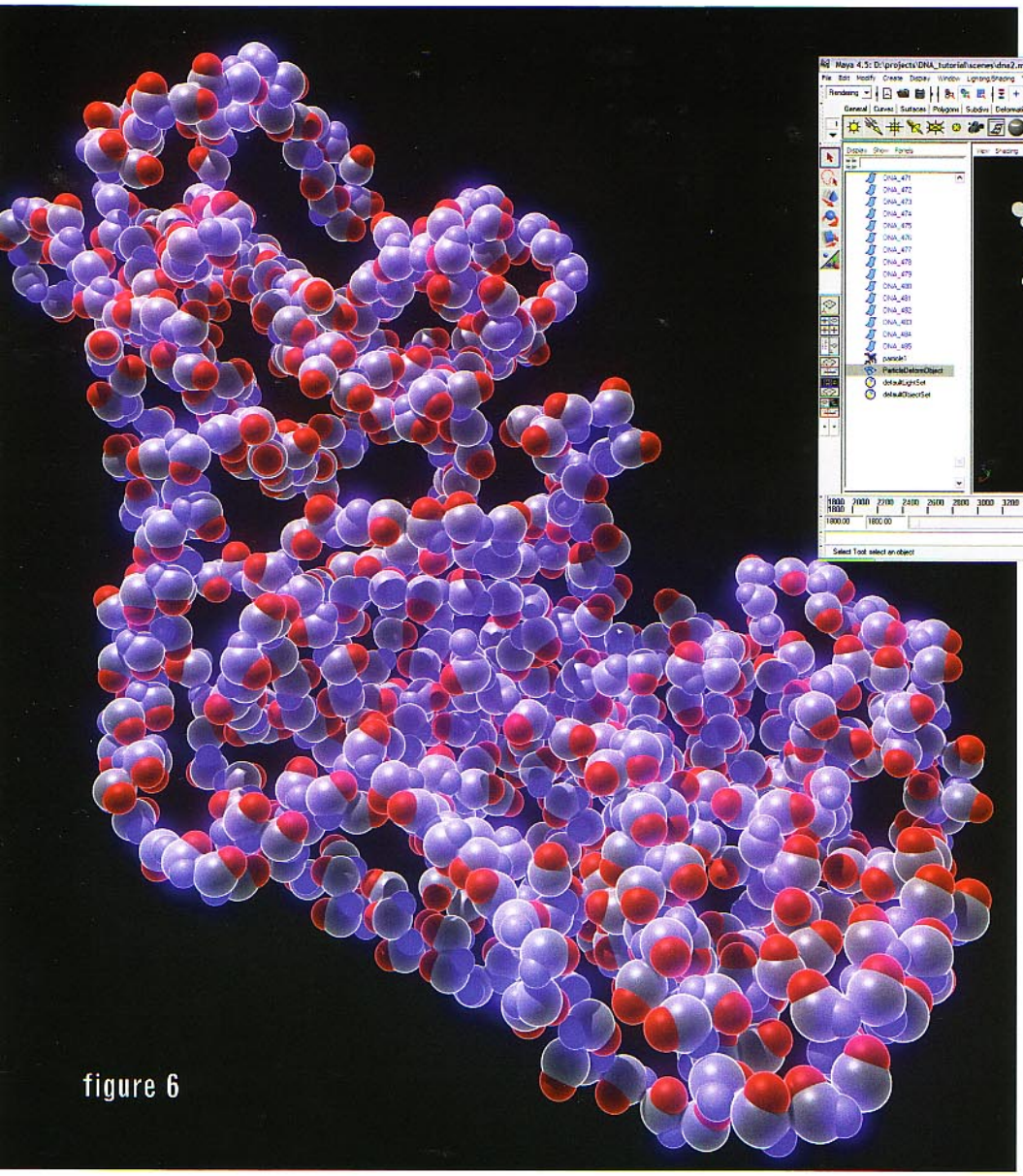


figure 6

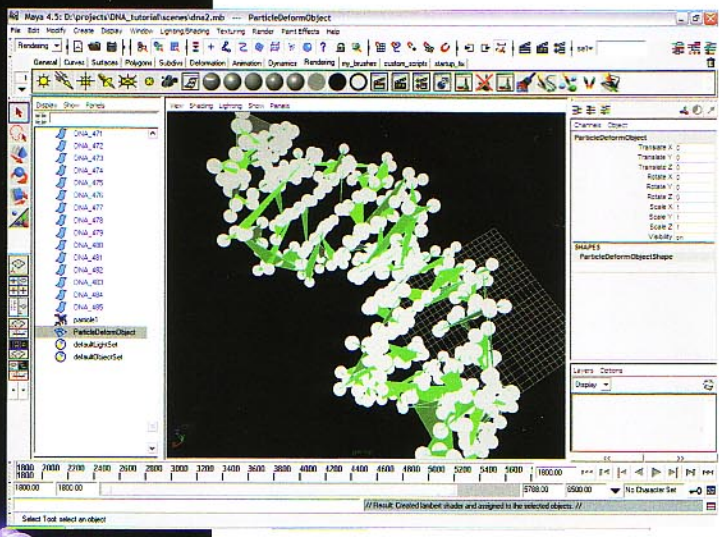


figure 7

stands for a particular oxygen atom in the structure. A little background info on your molecule's structure can go a long way in figuring this stuff out (as well as quite a bit of trial and error). Getting the right numbers into the right fields is the hard part; the rest is easy.

You can adjust the number of sections and spans in the spheres pdbReade.mel will create with the fields at the bottom of the interface. However, it's best to keep the resolution of the spheres as low as possible. If it's a big model, you don't want to bring your machine to a grinding halt because each of the 3,000 or so spheres has eight sections and 16 spans.

Try running the script with our DNA PDB file loaded and see what happens. Type "3" in the Atom Field and "7," "8," and "9" in the X, Y, and Z fields. Make sure you hit enter after you put each of these values in. Occasionally, I've typed in the correct number, forgotten to hit enter, and gotten very strange results.

If you look at the original PDB file you may find a group of rows at the end of the file that start with "HETATM." The script can read these rows but you may notice that the columns are shifted and thus the numbers the script are using for X, Y, and Z, are now off. The best way to fix this is to cut and paste these rows into another file and save that as a separate PDB which you can add to the scene once you have the molecule loaded. You may not need or want this data; ask you local science nerd if you're not sure. The DNA PDB file we're using for this example does not have these rows.

### Go ahead and run the script.

Calculating the molecular structure can take some time if it's a big model. Even on a fast machine, I've waited up to five minutes while it was building the molecule. This particular DNA structure is not big, so it should be quick.

fig 2: DNA successfully loaded into Maya as NURBS spheres

Once it's done, you may see nothing on your screen. The molecule may be very large and behind the camera, so zoom out to see if there is something there or not (Shift-A will focus the camera on the whole scene, of course).

If you look in the outliner, you'll see that the spheres are arranged in groups according to atomic symbol (nitrogen, carbon, oxygen, hydrogen and phosphorus). A phong shader has been automatically created and assigned for each type of atom. Congratulations –

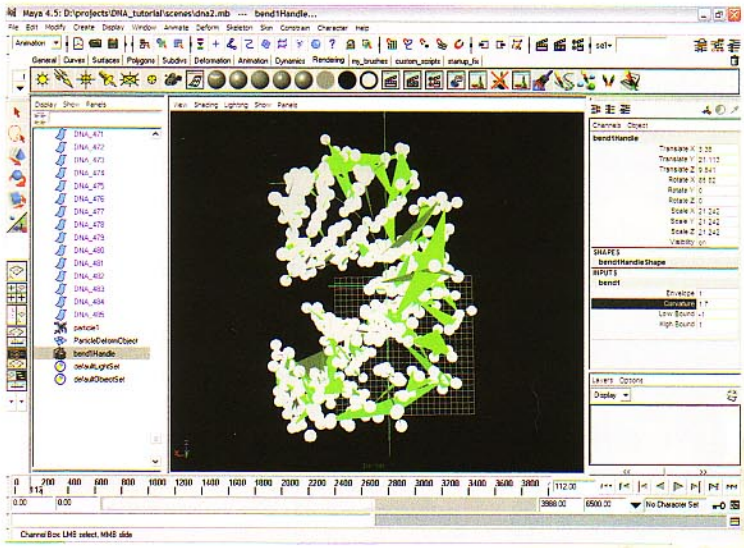


figure 8

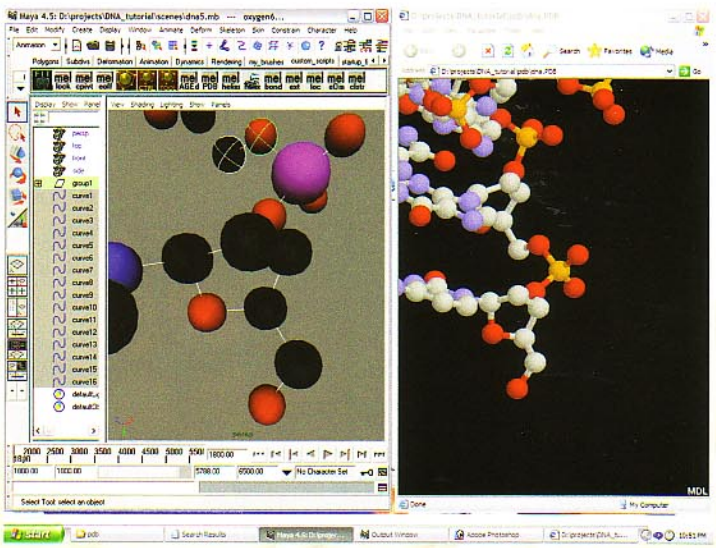


figure 9

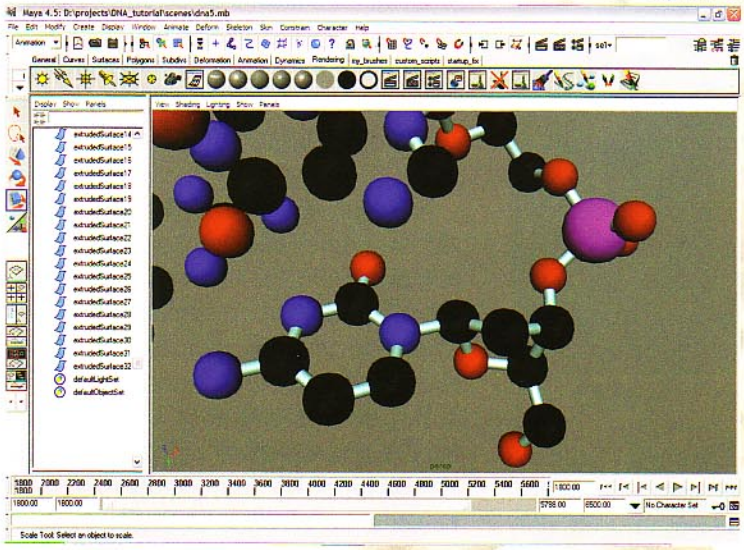


figure 10

you've got your DNA. Save the model as dna1.mb, and now let's have some fun with it.

### fig 3: The Outliner shows the atoms in groups

## Part Two: Space Filling Models Using Particles

There are several ways scientists represent molecular structure. These include wireframe, stick, ball and stick, space-filling, ribbon and backbone. If you open a PDB file in Chime or one of the other PDB displaying programs I've mentioned, there are usually display modes that allow you to choose from one of these types of representations. The pdbReader.mel script only brings molecules in as spheres, which is pretty close to the space-filling representation. There are no options for adding sticks or for an alternative representation. In this section, I'll show you how to convert the model into a particle object, which you can then render as blobby surfaces. Later, I'll demonstrate some ways to make a ball and stick model. There's no reason why you couldn't use this model the way it is or convert the NURBS spheres to poly spheres or whatever. It all depends on what you're trying to do.

Save your file as dna2.mb and make sure you have downloaded and installed the jPivToParticle script from www.highend3d.com (under the MEL scripts section>dynamics/FX) in your scripts directory. Select all the groups in the outliner and ungroup everything so that the outliner lists all the spheres in the model. Select them all and then type in the command line jPivToParticle "DNA_";. The script will take a moment to run. It will rename all the spheres DNA_1, DNA_2, DNA_3, etc. and will add a particle object. The script places a particle at each of the pivot points for all the spheres. Select all the NURBS spheres and either hide them or delete them. Save this file, and then save it again as dna3.mel. We want to have one version of the model saved at that last stage.

### fig 4: DNA as blobby surfaces

Center the pivot and move the object to the center of the grid. Scale it down a bit if you want to. I'm fond of rendering my molecule particle objects as a blobby surface. To do this, set the particle render type to blobby surface and hit the Add Attributes for Current Render Type button in the attribute editor for the particle shape node. Play with the radius and threshold attributes until you get something you like.

One trick I like is to add a radiusPP attribute and then write a short creation

expression to randomize the per-particle radius. To do this, go to the attribute editor for the particle shape node, scroll down and hit the "General" button under "Add Dynamic Attributes." In the dialog box, choose the particle tab and select radiusPP from the list. In the particle shape node per particle array attributes list in the attribute editor, you'll now see a new field for radiusPP. Right-click on the field and choose "Creation expression." In the expression editor, type an expression like particleShape1.radiusPP=rand(.5,1.5);. If you rewind and play the animation a bit you should see a randomization of the sizes of the particles. If you include this technique in an animation, just remember to render starting on frame 1 or higher instead of 0 so you don't have any weird resizing at the beginning of the animation.

This technique spices up the look of the model. But it also (artistically) compensates for the fact that by using the jPivToParticle script to turn the model into a single particle object, we've lost the relative sizes of the individual carbon, hydrogen, phosphorus, and oxygen atoms. We've also lost the individual shading for the atoms as well. What this means is that we've reduced our technical accuracy for the sake of aesthetics, but in some cases that might be OK (depending on what you're trying to show in your animation). Adjusting the threshold on the blobby surface render attributes will add that blobby look; when combined with an interesting shader and some lighting effects, this can yield some cool (but less accurate) results.

### fig 5: DNA as a single particle object, rendered as blobby surfaces

If you need to retain the relative atomic sizes and shading, you can use the jPivToParticle script to convert each atomic group. You could ungroup and run the script on the carbon, hydrogen, oxygen, phosphorous and nitrogen groups one at a time, and then regroup the resulting five particle objects together. If you set the particle render type to blobby surfaces or spheres, you can also set a radius and shading for each of these atom types. This is what I did for the anthrax toxin model pictured here.

### fig 6: Anthrax toxin rendered as four different particle objects, one for each type of atom

## Part Three: Adding Dynamic Interactions and Deformations to the Molecule

Another terrific script allows you to add the ability to deform the entire particle object as you would a piece of geometry. This is also useful if you'd like to add collisions to your molecules. The script creates a crude, simple polygon object based on the positions of the individual particles in your particle shape. You can set this object as a goal for the particles and then deform or animate the poly object. The particles will follow. Additionally, you can turn the poly object into a rigid body and have it collide with other objects.

The script is particleDeformationPoly.mel by Alex Bigott, which you can download from www.highend3d.com. To use it, go back to the dna2.mb file saved before you centered the pivot of the particle object or moved it. Select the particle object and then use the script editor to open the particleDefo rmationPoly.mel that you downloaded into your scripts directory. Run the script, and the crude polygon object should appear in a few seconds. Select the particle object, then the polygonal object, and under the dynamics menu select Particles>Goal. Set the goal for the particles to 1.

### fig 7: particleDeformationPoly.mel applied to the DNA particle object

If you move your polygonal object, you'll notice that the particles don't move. You're in dynamics land now so you have to remember that the particles won't update until you hit play AND you'll have to always rewind and play from the beginning. If you're making an animation, remember to give yourself some extra frames at the beginning so that the particle object can snap to the polygonal goal. Try putting a non-linear deformer like bend or twist on the poly object, animate it, and then press play.

### fig 8: A bend deformer applied to the polygon object. The result is a bent DNA

If you make the crude polygon objects rigid bodies, you can easily animate molecules bumping into each other. Apply a transparent lambert shader to the poly object so it doesn't show up in the render. Also keep in mind that the collisions will be calculated based on these polygons and not the particles themselves, so there may be some intersection of the particles at the point of collision.

An alternative to the particleDeformation.mel script is a little more mundane. You can "model by hand" a simple polygon object that roughly describes the shape of the molecule. Then parent the particle object to this dummy geometric object. Assign a transparent lambert shader to the dummy object and make it a rigid body. Deforming the polygon object using this method won't give you the same results as the particleDeformationPoly.mel script will, but it can help prevent strange intersections of the particles at the point of collision.

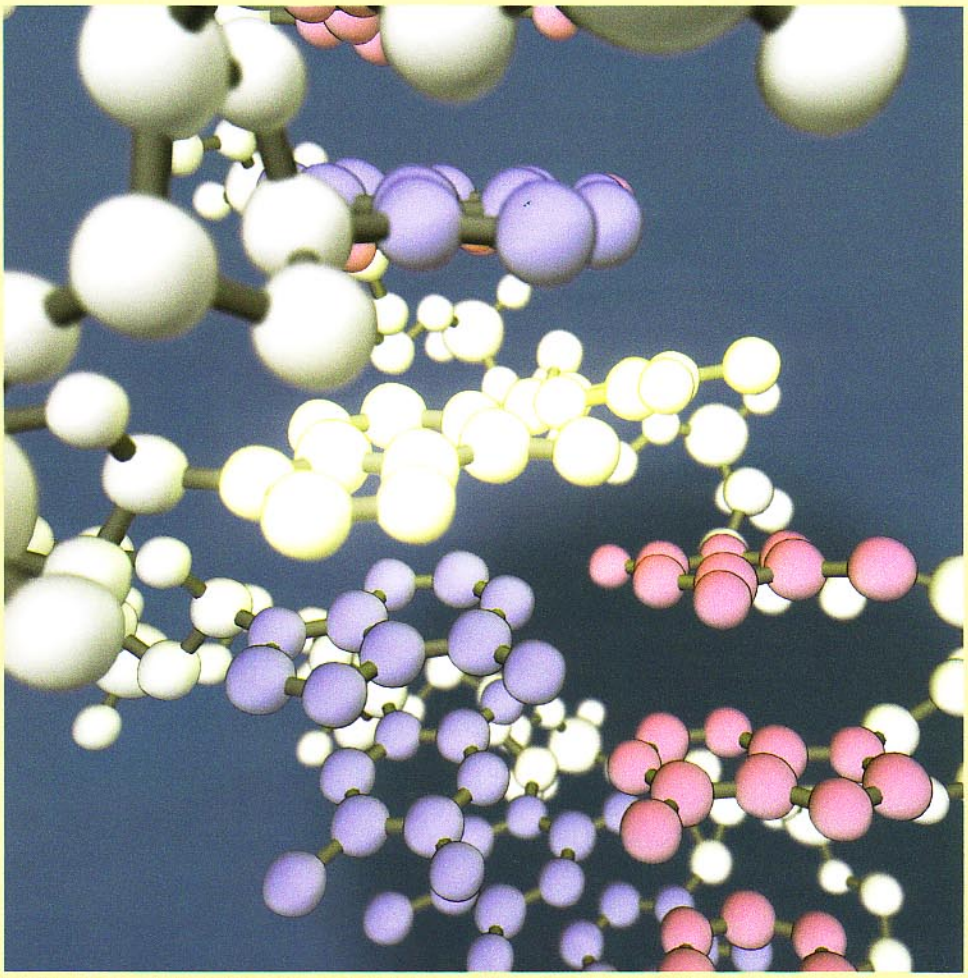
## Part Four: Creating a Ball and Stick Model

As far as I know, no one has written a PDB translator script or plug-in for Maya that will automatically create the stick bonds between atoms. The data for creating the bonds is often there in the file and certainly programs like Chime and PyMol can display these bonds. It would be an interesting challenge to add this functionality to pdbReader.mel. For the time being, there's an old fashioned bond creation method that involves making cylinders between the spheres. I've written a couple scripts that speed this process up quite a bit. They are bond.mel and extrudOmatic.mel. You can download them from my site at http://www.kellerrific.com/scripts. Install them in your scripts directory and then make a shelf button for them or a hotkey/marking menu.

Go back to the first DNA model file (dna1.mb), in which our nucleic acid is depicted as NURBS spheres. Select all the groups in the outliner and group them together. Center the pivot and move them to the center of the grid. In the perspective window, select all the NURBS spheres and scale them down to .74 on the X, Y, and Z scales.

Scale the Maya window so that you can view the original DNA PDB file in your protein viewing application (Pymol, Chime, etc,) side by side with the DNA model. The bond.mel script creates a one-degree curve between the pivot point of any two selected objects. We are going to make curves between each appropriate atom in the DNA model using the PDB loaded into our protein viewer as a guide. Start by selecting any two adjacent spheres and run the bond script. Repeat until the model in Maya looks like the one in the PDB viewer. Rather tedious, isn't it?

fig 9: Using the bond.mel script to make bonds between the atoms. The same model is loaded into Chime and zoomed in on the same spot for reference



THE BOND.MEL AND EXTRUDOMATIC.MEL SCRIPTS WERE USED TO ADD STICK BONDS TO THIS DNA MODEL.

You can use the extrudOmatic.mel script to quickly turn all these one-degree curves into cylindrical bonds. Make sure history is on and create a NURBS circle. Select the circle first (you have to select the profile curve first and then all the extrusion paths for the script to work properly) and then in the outliner, shift select all the curves. Type extrudOMatic.mel in the command line. Cylinders should appear around all the curves. As long as history is on, you can adjust the radius of all the bonds at once by scaling the NURBS circle. Finally, select all the cylinders and center the pivot.

fig 10: extrudOMatic turns all the curves into cylindrical bonds easily

After working through this tutorial, you should end up with at least three different versions of DNA and several options for working with molecular data. I've included renderings of several of my favorite molecules to demonstrate some creative ways Maya can be used to enhance the macromolecular structures. And if you've worked through this tutorial carefully, I hope you'll have enough of an appreciation for the proper structure of DNA to cringe the next time you see it misrepresented in a movie or an advertisement!

# Occlusion in Softimage|XSI

BY ANDREAS BYSTROM WITH RITCHIE L. ROBERTS

## Introduction

IN CONTRAST TO REAL GLOBAL ILLUMINATION (GI), WHICH WAS DESIGNED TO BE AS PHYSICALLY CORRECT AS POSSIBLE, OCCLUSION IS IMPLEMENTED TO TAKE ADVANTAGE OF AS MANY SHORTCUTS AS POSSIBLE WHILE STILL PRODUCING REALISTIC RESULTS. OCCLUSION IS ALREADY BEING USED BY MAJOR VFX COMPANIES SUCH AS INDUSTRIAL LIGHT + MAGIC AND BLUE SKY STUDIOS. THE METHOD IS DEFINITELY PRODUCTION-PROVEN AND SOMETHING WE WILL SEE USED MORE AND MORE IN THE FUTURE.

IN THIS TUTORIAL, I WILL EXPLAIN HOW TO SET UP OCCLUSION PASSES IN XSI, HOW TO RENDER THEM, AND HOW TO COMPOSITE THEM WITH OTHER PASSES USING THE BUILT-IN COMPOSITING SOLUTION IN XSI.

### How it works

Without going into the technical details, you can say that an occlusion pass will simply darken the areas where two or more objects come in near proximity to each other. Imagine a room with the only light source being the sunlight coming in from a window. In the real world, the photons would be emitted from the sun, travel through our atmosphere and finally, illuminate the room. Most of the photons would be blocked by the wall, yet some would find their way through the glass of the window and land on the floor in the room. From there they would bounce towards the walls and objects in the room. The areas that receive more photons will be brighter and the areas that receive fewer photons will be darker. For instance, if you had a bed in the room, the area under the bed would be darker, because the bed itself is blocking the photons that otherwise would have landed on the

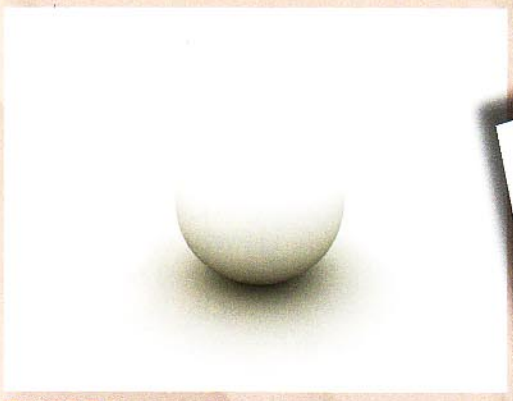


FIGURE 1

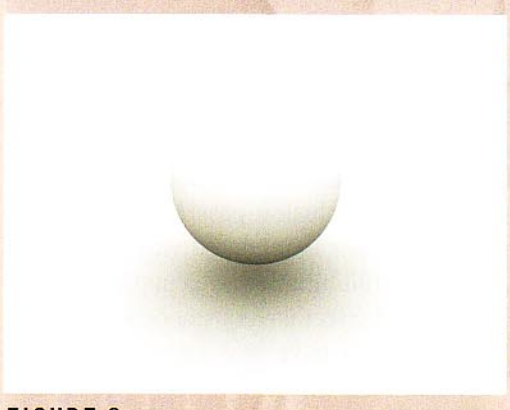


FIGURE 2

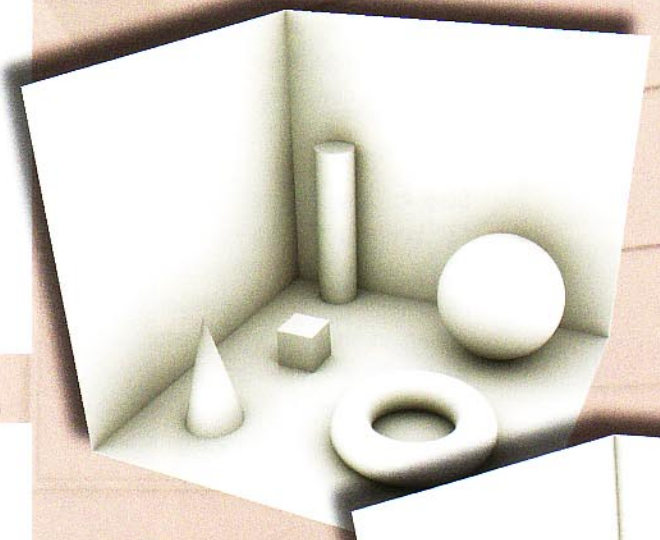


FIGURE 3

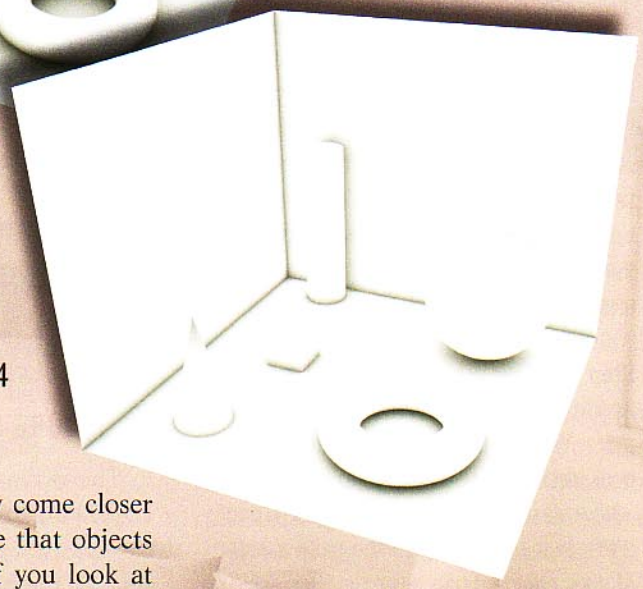


FIGURE 4

floor beneath. As I mentioned before, occlusion is about taking as many shortcuts as possible. Instead of computing billions of photons, it equates the brightness of a surface depending on the distance to the surrounding surfaces. If the distance is short, the surfaces will be dark. If the distance is large, the surface will be brighter.

In this first example, I have made a simple scene with a ball and a plane. If you study fig.1 you will quickly see that the further away from the middle of the sphere, the brighter the surface will be. This is because the distance between objects is increasing.

Look at fig.2 and you will see that I have rendered the same scene with the sphere moved up slightly in the Y-axis. Since the distance from the sphere to the floor now is greater, you will not get as dark of an area under the sphere anymore.

I have also assembled a slightly more advanced scene that shows more clearly how the different objects will darken each other as they come closer together. You will also see that objects can darken themselves. If you look at fig.3, you will notice that the corners of the room the objects are placed in are darker than the other areas of the wall. This is a good example of objects darkening themselves. Obviously, the reason why the corners get dark is because the distance between the different walls there is smaller than anywhere else is. Remember what I said before: All occlusion does is darken surfaces that are close together.

Most occlusion shaders will give you the option of adjusting the attenuation or falloff. I am not going to go into details on how to adjust these parameters, as you might be using a different occlusion shader than I am. It is important to know that the parameters control the attenuation or falloff for the darker areas. In the shader I am using, you have the ability to control both the "near" and "far" attenuation. The "near" parameter will control where the effect starts the attenuation and the "far" parameter will control where the effect ends the attenuation. In fig.3, I used a "near" value of 0 and a "far" value of 10, while in fig.4, I used a "near" value of 0, but I decreased the "far" value to only 1. This makes the darker areas much smaller.

### Setting up an occlusion pass in XSI

You should now have a basic understanding of how occlusion works. The next step is to learn how to set up an occlusion pass for rendering within XSI. XSI does not come with an occlusion shader, so we need to download one first. Daniel Rind (http://www.animanium.com) has been kind enough to write one and release it for free. You can download it here: http:

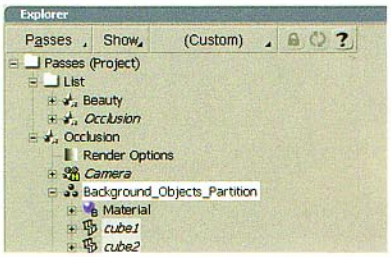



FIGURE 5

//animus.brinkster.net/downloads/Dirtmap.zip. Please follow the installation instructions in the readme file that comes with the shader. It is important that you install the shader properly, as we will use it later in this tutorial.

**Step 1:** Download and extract occlusion_tutorial.zip from http://www.renderart.net/temp/occlusion_tutorial/occlusion_tutorial.zip. Go into the Scenes folder in the project and open occlusion_room_step1.scn in XSI.

**Step 2:** If you are not already in the rendering module, press the "3" key to go there. We will now create the start of our occlusion pass. To do so, go to Pass->Edit->New pass->Empty. In the property page (PPG) that pops up, give the pass a name of your choice: I named mine "Occlusion." Now close the property page.

**Step 3:** Press the "8" key to open up the explorer. In the explorer, press the "A" key to go to the passes section. Now you should see an item called Background_Objects_Partition. There is something called a partition in XSI, which is a group that is used for passes. This partition contains all the objects in your scene. You can define your own partitions by going to Pass->Partition->New; however, in this example, we only need the default partition. As you might know, you can use normal groups to share a material between several objects. You can do this with partitions as well, and this is exactly what we are going to do. If you assign a material to a partition or group, it will automatically override any local material that the objects themselves might have. A partition will also override any group sharing a material between several objects. Make sure the partition that is called Background_Objects_Partition is selected and go to Get->Material->Phong. You should now have a setup like the one in fig.5

**Step 4:** Close the explorer and press the "7" key to open the rendertree. This is where we will load the occlusion shader. In the rendertree, go to Nodes->Texture->More. In the window that pops up, click on the Paths button and locate the item called "Dirtmap" from that menu. You should now see a file called "Dirtmap." Select it and press the Ok button.

**Step 5:** Unplug the Phong node from all the slots and plug the Dirtmap shader into the Surface slot of the material. Your rendertree should now look like Fig.6

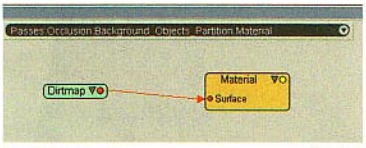



FIGURE 6

The occlusion pass is now completed. If you draw a render region in the camera viewport you will see the effect. As you might have noticed, the values of the shader need some further tweaking. For this I have created a preset file. To load it, double-click on the Dirtmap shader in the rendertree and press the Load button. Locate the file dirtmap_preset. Preset from the project folder and press Ok. Now if you draw a render region again you will see that the effect looks much better. For a more detailed explanation of the different parameters in the Dirtmap shader, see the manual that comes with it. Now that you've set up the occlusion pass, you will see that if you click on the dropdown menu seen in fig.7, you can switch between the different passes. If you select the beauty pass and draw a render region in the camera viewport, you will see that it renders the currently selected pass. Once a pass is selected, you can go to Render->Render->Current pass to render it out to a file.

FIGURE 7

## Compositing

Load occlusion_room_step2.scn. The only difference between this scene and the one you just completed in the previous step is that it has the proper render settings for the occlusion pass. In the previous step, you created a new pass and because of that you got the default settings. If you wish to see the rendering settings for a pass, make sure it is selected in the pass dropdown menu and go to Render->Render->Options.

In this example, we have two different passes. The beauty pass (see fig.8) and the occlusion pass (see fig.9). The beauty pass is rendered with the use of Global Illumination and Final Gathering. Yet, the effect is very smoothed out and we do not have any details. There are hardly any shadows under the objects, so this is what we will use the occlusion pass for. We will use it to darken some areas and add more definition to the details in the scene.

The passes are already rendered out and included in the project, but if you want to render them yourself go to Render->Render->All Passes.

This is when we will start doing the actual compositing work. XSI has a built-in compositor called the FXTree, but you can use any compositing application for this. To open up the FXTree, go to View->Views->FXTree. In the FXTree, go to Passes->Get All Passes. This will create a new FileInput node for each pass. To combine these passes, create a MathComposite node by going to Ops->Composite->Math Composite. Connect the beauty pass to the left green arrow on the MathComposite node and the occlusion pass to the right green arrow. Now hover your mouse over the MathComposite node and press the round button where it says "e" (edit). This will bring up a property page for the node where you can see all of its properties; click on the dropdown

FIGURE 8

FIGURE 9

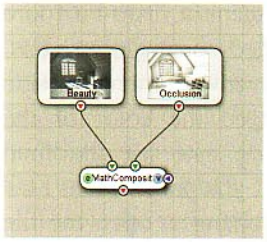



FIGURE 10

menu in the property page and change the mode to Multiply (A*B). When you are done you should have a tree as seen in fig.10 There are a number of different ways you can composite your occlusion pass, one of which is known as Ambient Occlusion. This is when you use the occlusion pass to "weight" the ambient and diffuse pass together. What we are doing in this example can be called Diffusion Occlusion, since we use the occlusion pass to occlude the diffuse or beauty pass.

In the FXTree, go to View->Viewer. This is the window where you will be able to preview your composite. To view the composite, hover your mouse over the MathComposite node and press the round button where it says "v" (view). As you might notice, the occlusion pass will darken some areas which are obviously supposed to be bright. In fig.11, I have highlighted an obvious area where you have direct sunlight hitting the room's floor and it's still being darkened too much by the occlusion pass.

To get rid of this problem, we will do a little adjustment to the occlusion pass before we multiply it with the beauty pass. Delete the MathComposite node, as we will need it again later, but we will remove it for now to keep a clean workspace while we are doing the rest of our compositing. Create a ColorCorrect node by going to Nodes->Color Adjust->Color Correct. Connect the beauty pass to the left green arrow of the ColorCorrect node. Open the property page for the ColorCorrect node and go to the Histogram tab. In there, adjust the histogram as seen in fig.12.

How you adjust the histogram depends on the look you are after. Adjusting it the way I did in this example will cause the beauty pass to look as it does in fig.13. As you can see, only the brightest parts of the pass are left. These are the parts where we also want the occlusion pass to be bright so as to avoid darkening on these areas in the final composite.

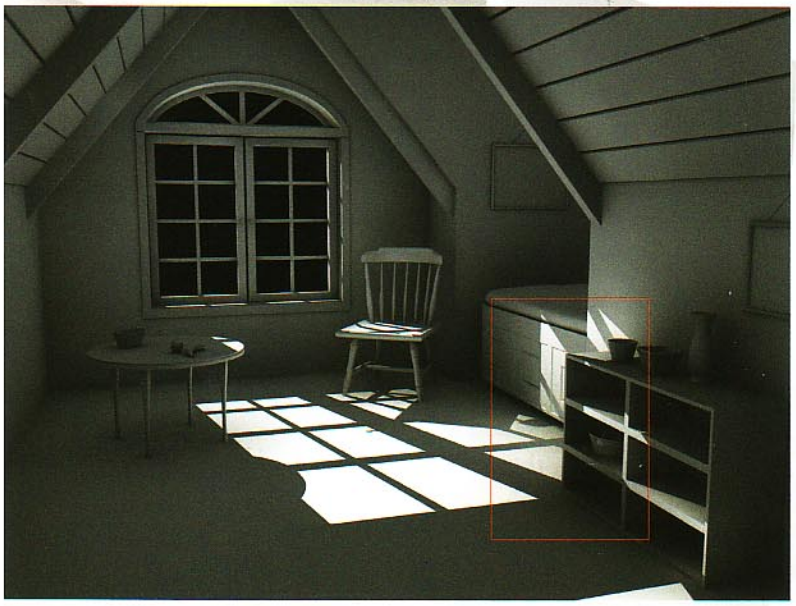


FIGURE 11

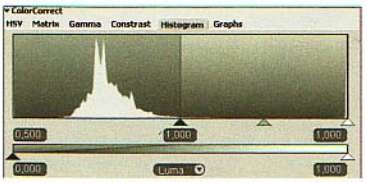



FIGURE 12

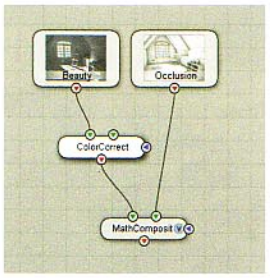



FIGURE 13

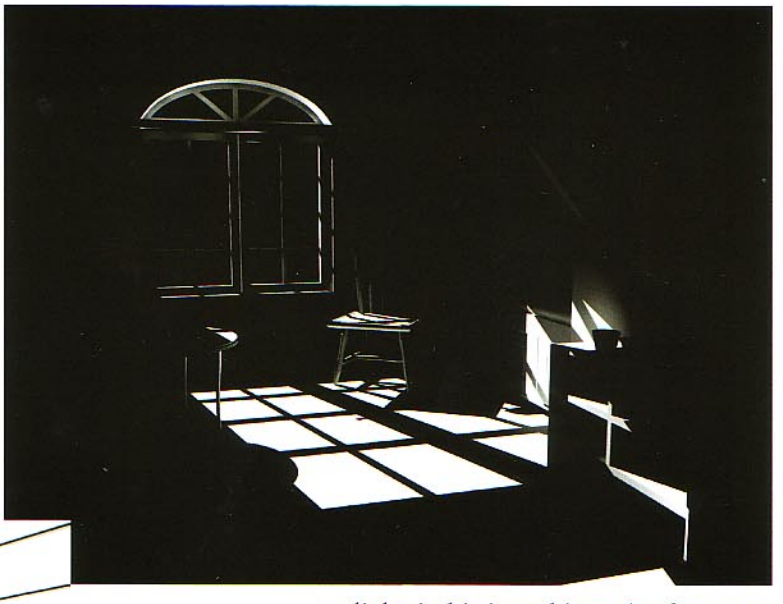


FIGURE 14

FIGURE 15

Next step is to create a math composite node, again by going to Ops->Composite->Math Composite. The default mode for the MathComposite node is: Screen (A+B-A*B), which is exactly what we want. So you don't need to change it. Now connect the output (red arrow) of the ColorCorrect node to the left green arrow of the MathComposite node and connect the occlusion pass to the right green arrow of the MathComposite node. Your FXtree should now look like fig.14.

Hover the mouse over the MathComposite node and press the "v" button to preview it. It should now look like fig.15. As you can see, we used the adjusted version of the beauty pass to brighten parts of the occlusion pass.

Now we will multiply this adjusted version of the occlusion pass with the beauty pass again. To do so, create a MathComposite node by going to Ops->Composite->Math Composite and setting its mode to Multiply (A*B). Connect the beauty pass to the left green arrow of the MathComposite node and connect the output (red arrow) of the previous MathComposite node to the new MathComposites right green arrow. You should now have a FXTree as seen in fig.16.

In fig.17 you can see the final composite. Note that the areas where direct sunlight is hitting objects in the room are now always bright.

To make the scene a little bit more interesting, I added some glows/halos, and I made the environment outside bright and changed the color of the whole image. You can see the final result in fig.18. If you want to see how I did these final changes, open the file named occlusion_room_finished.scn. I did not change anything in the scene itself. All the changes are made using the FXTree. Feel free to continue experimenting on your own; it is a great way to learn new things!

Below you can see a more advanced scene where I used an occlusion pass. This scene was actually rendered out in nine different passes that were all composited together using the FXTree. (club_final.tif)

I have put together a small compositing breakdown to show what is going on in the image above. First we have the beauty pass. It was rendered with the use of Global Illumination and Final Gathering. Again, I used a very smoothed out Global Illumination solution, and as you can see, the pass lacks definition of the shadows as a result. (beauty.tif)

Next comes the occlusion pass. This pass is being composited exactly as shown in this tutorial. (occlusion.tif)

OCCLUSION

You might not notice this in the final composite, but there is actually a little fog in the scene. This pass was "screened" over the whole compositing to brighten objects further away from the camera. (fog.tif)

This is the reflection pass. I rendered this pass to gain better control of the reflections in the scene. It helps balance the reflections and allows me to add effects such as glows to them specifically. (reflection.tif)

This pass is there to give more control to the look of the curtains. It was used to brighten them and to create the glow effect around them. (curtains.tif)

This is a simple mask for the curtains. I used this to constrain some of the effects to the upper parts of the curtains only. As you can see in the final composite, parts of the curtains hang over the wall, so I did not want to add any glows or brightening to that part. (curtains_mask.tif)

As it turned out, all the glows and brightening of the curtains made the windows way too bright. To fix this problem, I rendered out this simple mask of the windows. I used a MathComposite node set to Subtract (A-B) to composite them over the other passes. (windows.tif)

This is another pass I used to correct parts that had been too bright. This is a mask for the curtain hangers. Without this pass, a very small but important pass, the curtains appeared to be floating in the air. (curtains_hangers.tif)

This is a mask for the pillars in the scene. I used this on several places to constrain different effects such as the glows around the reflections and around the curtains. (mask.tif)

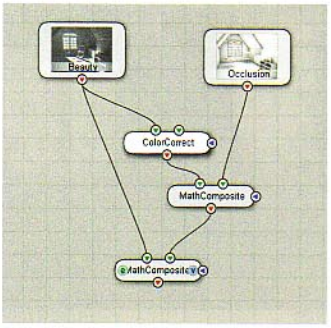
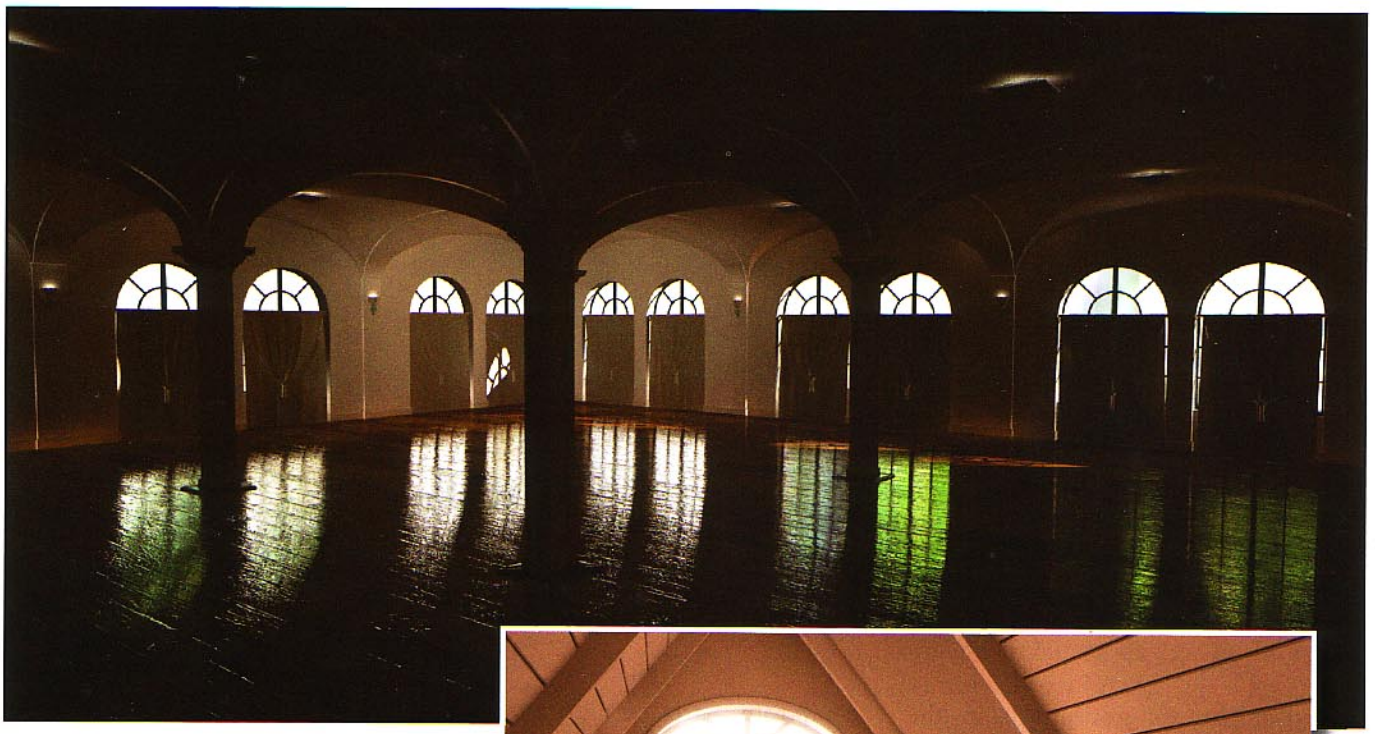



FIGURE 16

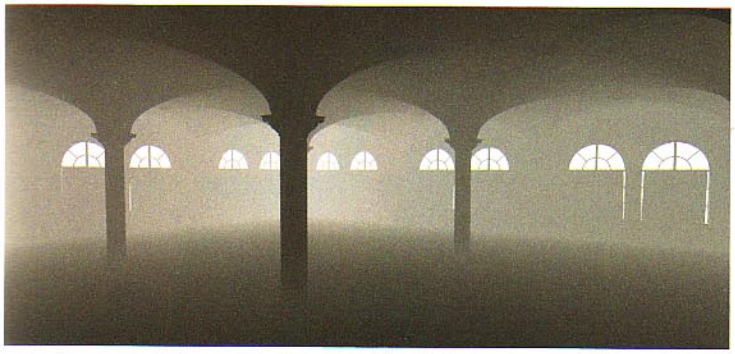
FIGURE 17



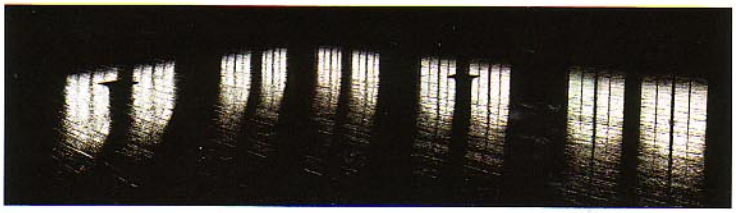
BEAUTY

FIGURE 18

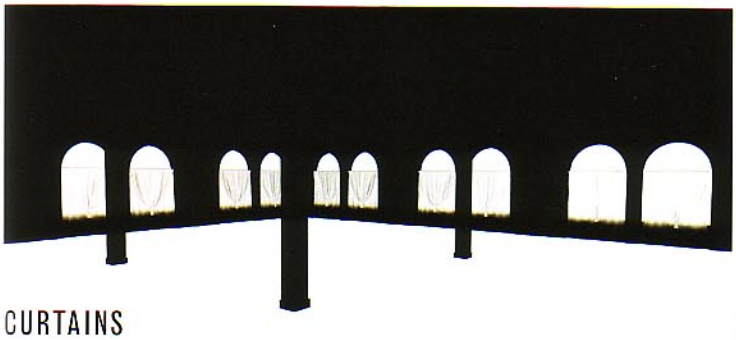
CLUB FINAL



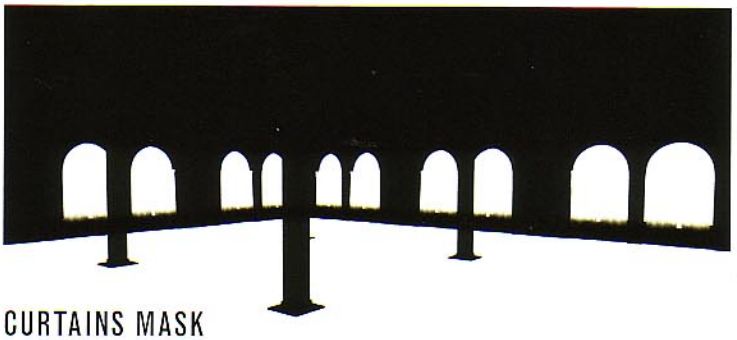
FOG



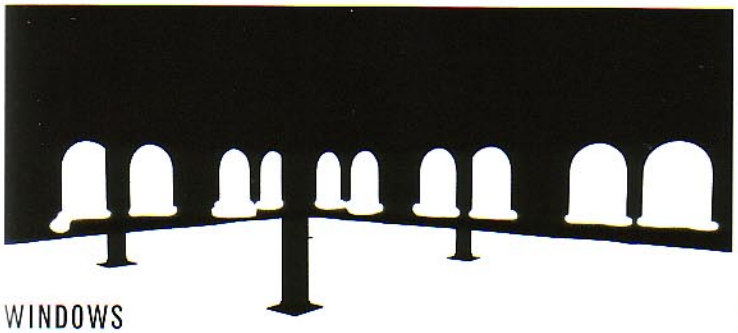
REFLECTION



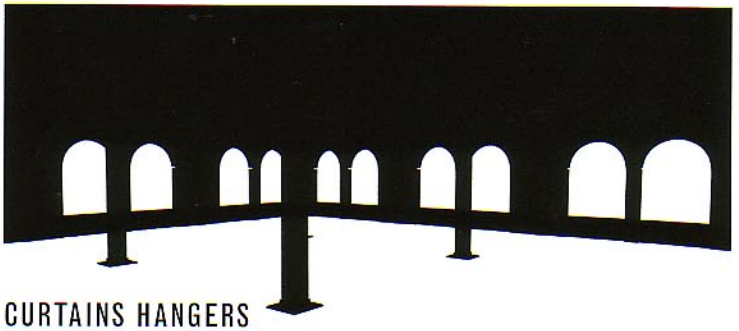
CURTAINS



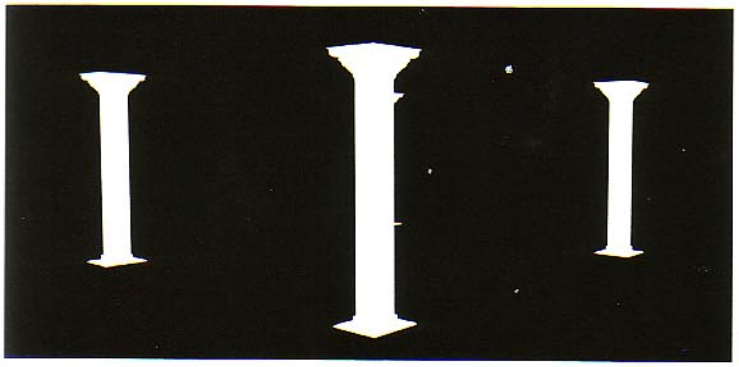
CURTAINS MASK



WINDOWS



CURTAINS HANGERS



MASK

## Conclusion

You have now learned how to set up an occlusion pass for rendering as well as how to use occlusion passes in compositing. Like I mentioned before, there are several ways to composite an occlusion pass, and I have only showed one of them. Feel free to experiment with different approaches yourself.

Using an occlusion pass together with a beauty pass that has used Global Illumination, as I did in this tutorial, might seem a little weird, as occlusion is supposed to do most of the job that Global Illumination does. However, when dealing with Global Illumination, you usually undersample the solution a lot, causing most of the details to be lost. You can render Global Illumination solutions with very detailed and well-defined shadows, but it is a very slow process. Depending on the renderer used, occlusion passes will render relatively fast and will create dark shadows under even the smallest objects. This will create great contact shadows and prevent objects from appear to be "floating."

If you have any questions or other thoughts regarding this tutorial please send an e-mail to me at andreas@renderart.net

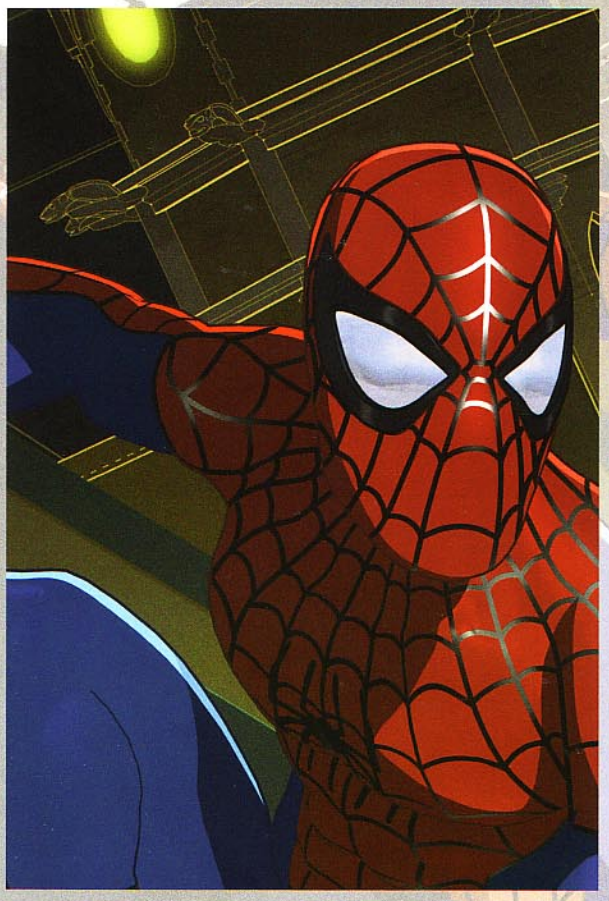
# Character Modeling for "Spider-Man" the Cartoon TV Series

BY STEVEN WENDLAND
PRODUCER FOR "SPIDER-MAN," MAINFRAME ENTERTAINMENT

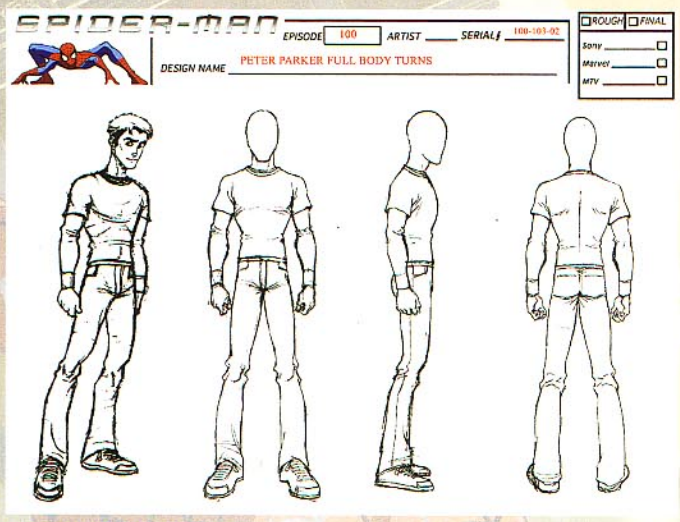
LIKE MANY OF US, I GREW UP WATCHING THE EARLY SPIDER-MAN TELEVISION SERIES and reading the comic books. He was my favorite out of all the superheroes, though I never really had the opportunity to emulate him as I might have liked. When Halloween came around, my mother would insist that Superman was a much more likeable superhero. The feature film results at the 2002 box-office would tend to disagree with my mother's point. However, I suspect that the real truth was that Superman was a far easier costume to make than Spider-Man – all you really needed was a red towel, blue long underwear, red shorts, and an "S" pinned to your shirt. People everywhere would recognize you as Superman. On the other hand, to achieve the look of Spider-man, the entire costume, from head to toe, would need to be detailed and original to be convincing. With limited resources, talent and tools, the result would be a laughable mess. I suspect our Mainframe modeling team would concur.

When we at Mainframe first received word that Sony Pictures Television might be doing the Spider-Man animated series, we had already been experimenting with 2D toon-like aesthetics for several years, but had never had the opportunity to implement or refine the style for a television series. This changed when we were awarded the contract of the follow-up animated series to the Sony feature film. They told us they wanted to capitalize on the dynamic that 3D CGI could bring to the product, but in a style that looked like a comic book come to life. This challenge affected each of the departments on this television series in a unique way, but I'll focus this particular discussion on the work done by our modeling department.

During the development phase of the series, we experimented with several different models, attempting to flatten the look of an inherently three-dimensional medium. Our early development work gave us some lead in the process, which allowed us time to concentrate on refining the style. This discussion will focus on character models, since that is where the majority of our time was spent. The first two models we built for this production – Spider-Man and Mary Jane - can best be used to demonstrate our process.



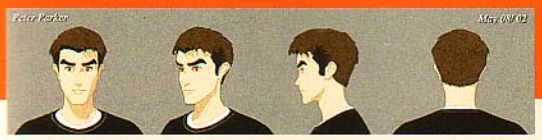
It was important to Marvel and to Sony that this Spider-Man, though fairly youthful by most comic book renditions and standards, be very toned and muscular. They also felt that it was necessary that the muscles be 'real' geometry, and not enhanced with textures. This might seem like an obvious request, but as we began work on the geometry shapes, we found that we needed to exaggerate the muscle tone of the model beyond that which it might have possessed with typical three-dimensional render software. This was done for two reasons: 1) Our 2D render technique tended to flatten the shapes, and 2) we needed exaggerated forms in order to bring out the perimeter paint lines, or contour lines.



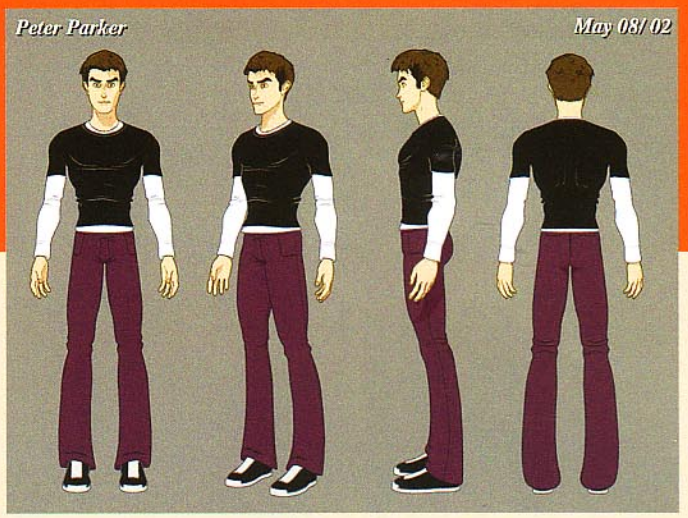

## PETER PARKER

Creating the skeleton for Spider-Man, and the subsequent weighting and model testing, was interesting because we needed to build a character that was human, but one that moved just beyond the ability of a human, all the while remaining natural-looking. This balance was achieved due to the close working relationship between the modeling and animation departments. Fred Pashe, the modeling supervisor, and Clayton Ryan, our lead Spider-Man animator, sat in the same room for the duration of the development process. Clayton put the Spidey model through as many possible poses and action scenarios that he thought we'd encounter on this series, and would then send these results, with any problems, back to Fred for additional model adjustments. One example of an area that we retooled by this process is Spider-Man's shoulders, which were a feature unlike anything that we'd modeled here at Mainframe in the past. We needed to ensure that the shoulders had the ability to rotate in a variety of directions, while retaining appropriate volume and mass. Another example of the model department purveying animation requirements can be found in the backbones of the model. We changed the center of gravity on Spider-Man's spine from the hips, to just below the head – effectively inverting the spine. This made it easier for the animators to execute the most convincing swinging action possible for this series.

Spider-Man's textures are curious for the reason that he possesses some aspects which are true to the comics (like the bold two-dimensional style) and some which are true to the feature film. It was important to Sony that he posses some of the more striking and memorable features of the film version. We made several attempts at incorporating these features, but in the end we opted for a much more simple and effective model. For instance, early versions attempted to raise the black web-line texture that covers his entire suit, as demonstrated in the costume of the film version. Some of our versions used geometry to build this, while others used a displaced texture. In the end,




PETER PARKER HEADSHOT




NEW PETER PARKER APPROVED ARTWORK

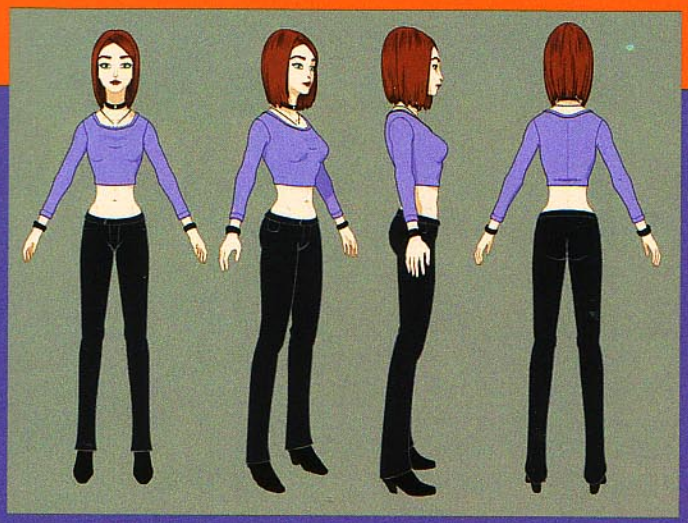
it was decided that it didn't work for a couple of reasons. First, the effect was one that seemed far too 'busy' for the style of this show. This added geometry took too much away from the two-dimensional style that we were working so hard to achieve. Second, the added weight of polygons of the model made it unnecessarily render heavy from the perspective of a television production schedule. Instead, we opted for the integration of a specular map over the flattened black web-line texture. This was consistent with the movie costume and proved to be aesthetically complementary, or at the very least, a nice balance to the two-dimensional style. Another example of texture consideration that was unique to this model is in the wrist and forearm regions, where special weighting needed to ensure that the web-line texture would not stretch to abnormal lengths. The result is one that gives the appearance of a spandex-like covering that fits comfortably around an imaginary body.

One notable difference from the feature film version is Spider-Man's facial shapes. While both the TV and film versions have a moveable jaw to help express dialogue, the costume of the feature version was severely limited in its ability to emote, due to the fact that the eyes of the mask were rigid. Obviously it would have looked very odd and distracting if someone had decided to include animated eye rigs on a live action model. However, our model was not constrained by this degree of real-world physics. The face is actually capable of a broad range of emotions, though in most cases we tended to keep the eye animation fairly subtle. Anyone familiar with the comic book versions of the character will recognize this leap in logic as a device that's used all of the time. The result is an assortment of remarkably convincing emotional expressions that help to tell the story – we're not just staring at a rigid mask, but a real character.

The Mary Jane model was equally challenging, though for different reasons. To begin with, Rak Tafarodi, our other modeling supervisor, quickly built a rough model that proved to be quite convincing when rendered with some of the 2D features that our software and lighting departments had been successful at achieving. Unlike Spider-Man, we were not constrained with this model by the need to recreate or copy the Mary Jane of the feature film. As well, throughout the comic books, the only thing that really remains constant is her red hair. However, with this freedom also came much revision. MTV was very interested in portraying her as, shall we say, a woman with swimsuit model proportions. In our first model she was much shorter and cuter, emphasizing her adolescence. This model went through at least three distinct model passes that included everything from smaller revisions, like the textures on her jeans, to much larger revisions, like longer legs and breast size. Throughout this process, Rak was very mindful of the fact that the two-dimensional render process required additional attention to certain points of the model's geometry. For instance, her head would render beautifully as a still, but a moving turnaround would reveal the difficulty of implementing the paint line or contour line shader. In particular, Mary Jane's nose needed to be pinched in such a way as to artificially bring out an edge that the contour shader could read. This, as you might expect, resulted in a model that wasn't particularly attractive when rendered in standard 3D settings – she looked a little in need of some plastic surgery. This surgery came in the form of two-tone paint shading (more on that in another discussion).

**It was important to Sony that he posses some of the more striking and memorable features of the film version.**

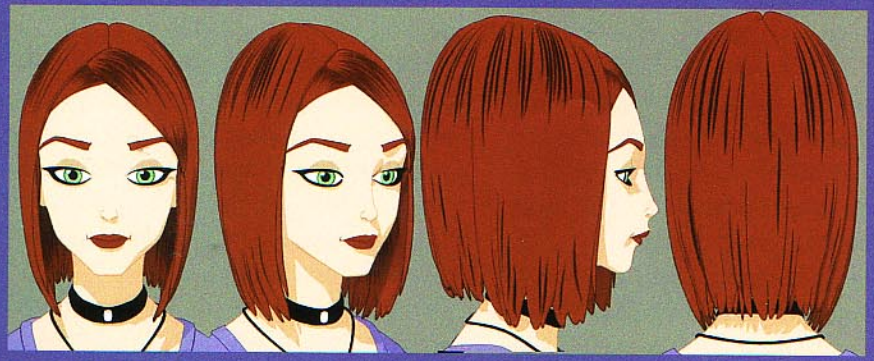
Another problem to be solved was Mary Jane's hair, which Sony really wanted to flow like real hair. We achieved this by keeping the geometry to a bare



MARY JANE "MJ" FULL-BODY

minimum, with no intersecting pieces, adding bones and a bone rig for movement, and accentuating the geometry with simple cartoon-like textures for the illusion of hair mass. The hair rig proved to be quite successful. The rig is based on a time-delay of the head movement, which gives the illusion of lightness and a reaction to the force of gravity.

One significant aspect to the modeling style of this series was the importance that Sony placed on a populated world. They wanted New York to look alive and busy. This population came chiefly in the form of characters and vehicles. For character population we created three different types of characters. In other words, three model categories of varying degrees of complexity. First, we had what we referred to as foreground characters. These were made up of characters like Peter, Mary Jane etc., but also included secondary characters (secondary, meaning those characters that required integration into the area directly surrounding camera, but were not principal to story). These secondary foreground characters were built with full facial animation capability, which then created a perfect resource for additional characters that might have smaller parts in future episodes. Not only was the style of the series conducive to building upon these pre-existing elements, but we built these characters in such a way as to allow for possible interchangeable parts. In other words, the head, torso and legs can, with minimal amounts of time, be interchanged with other models. As a final adjustment, changing a few of the colors, like hair and shirts, created what looked like new characters; characters that belonged in the same homogenous world. This approach held an appropriate degree of similitude of series style, while being efficient to the vigorous needs of a television production schedule. Second, the so-called 'mid-ground' characters were ones built specifically to populate the area at a depth just beyond the foreground focus. These characters retained a sufficient degree of detail and textures, but were not capable of facial animation. Third, the background characters, or 'drones,' as we referred to them, were the simplest of geometry and utilized only one color. There was just enough geometry to imply a head, body, arms and legs. We added crowd variety by creating heads that looked like a hat on a head, and bodies that looked like they were carrying a school backpack, etc. They were all essentially colored silhouettes. We were inspired by a similar technique often used in comic books to populate a frame while keeping the viewer's attention on the area of most importance. These drones were essentially walking skeletons – the geometry was the skeleton. Thanks to Jason Osipa, our technical director, these drones gave us the ability to add vast numbers of characters, easily 100 or more, with minimal difficulty to the scene and the render time. Additionally, both the mid-ground and background characters were almost always animated using motion capture data. This data was stored in a library of generic motion capture moves that the directing teams (the director, supervising animator, and the editor) could easily reference and access for appropriate shot-by-shot implementation. One notable feature of these characters was that the modelers had to avoid the use of expressions on the skeletons (expressions, in this case, being mathematical equations to compensate for a weighting problem outside a typical skeleton). This was done for the purpose of a smooth implementation of the motion capture data.

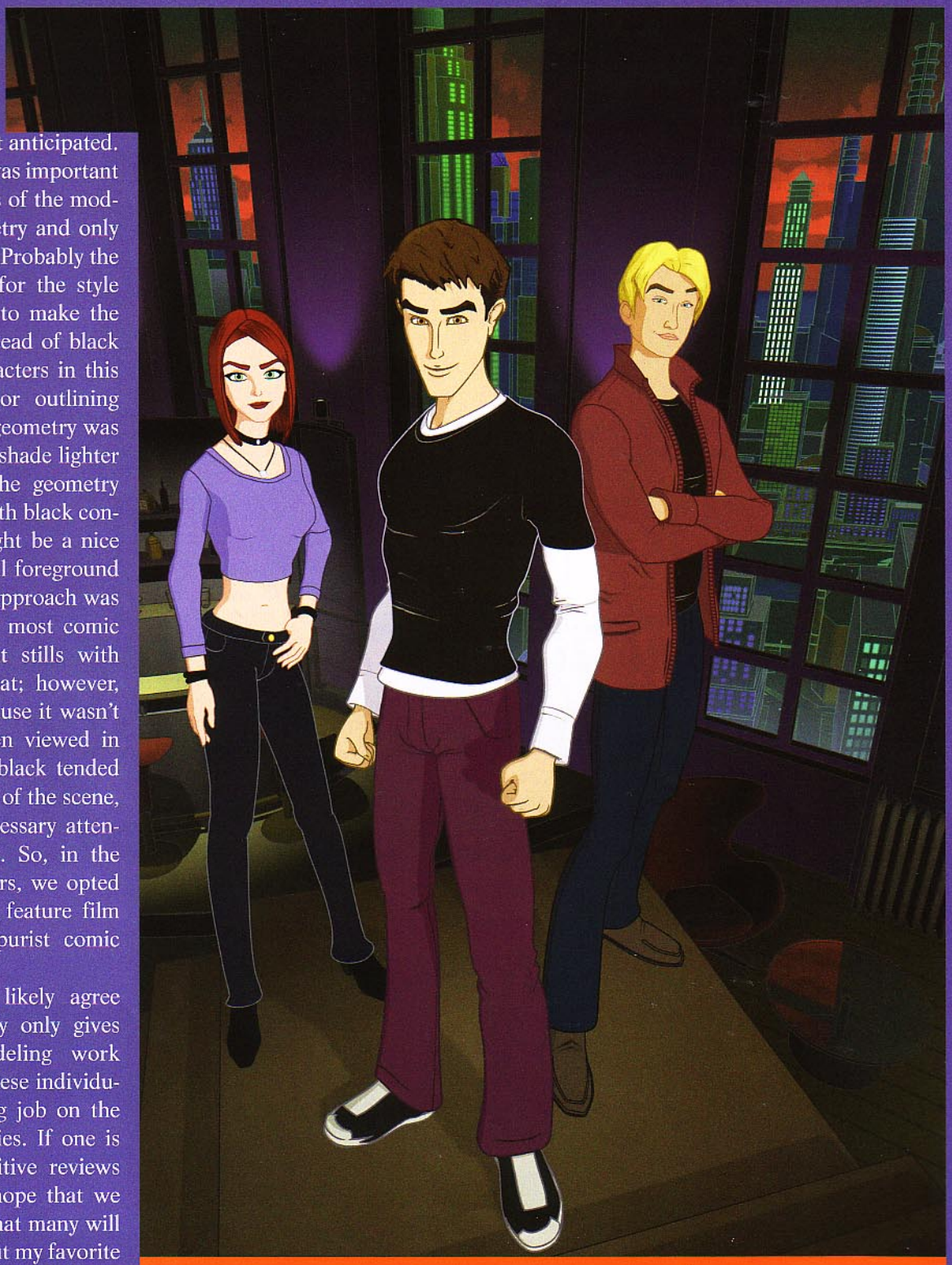


MJ HEADSHOT

With respect to the sets built for this series, we tended to do a lot of the detail work with textures, though

not as much as we had first anticipated. As with the characters, it was important to Sony that the set details of the models be created with geometry and only accentuated with textures. Probably the most important decision for the style of sets in this show was to make the contour lines colored, instead of black or gray. As with the characters in this series, the perimeter color outlining the edge of each piece of geometry was either a shade darker or a shade lighter than the base color of the geometry itself. We experimented with black contours, thinking that it might be a nice compliment to the colorful foreground characters. After all, this approach was typical of backgrounds in most comic books. Early development stills with this approach looked great; however, we decided against it because it wasn't nearly as convincing when viewed in motion. We felt that the black tended to muddy the composition of the scene, as well as bringing unnecessary attention to the contour lines. So, in the end, as with our characters, we opted for more of an animated feature film aesthetic rather than a purist comic book approach.

Our modelers would likely agree that this discussion really only gives some idea of the modeling work involved on this series. These individuals have done an amazing job on the Spider-Man television series. If one is to believe the many positive reviews about this show, then I hope that we have built a Spider-Man that many will enjoy for years to come. But my favorite critique is by none other than Stan Lee, who told the New York Post that, "this one has received the care, the attention, the love and the expertise it deserves." As far as costumes go, this one seems to fit pretty well. Mothers everywhere should be proud.



SPIDERMAN CAST SHOT

# PODS & TALENT UNITE

## An interview with lead artist GEOFF McAULIFFE of Brickyard VFX on bringing the Volkswagen Beetle and APPLE IPOD TOGETHER.

BY JOHN LAFAUCE, JR.

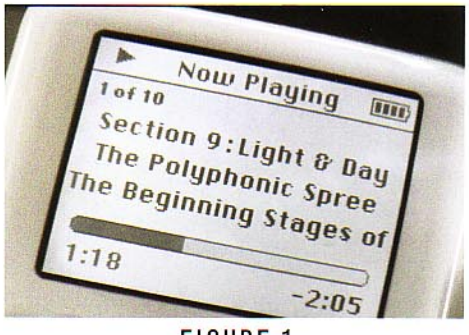



FIGURE 1

**WHEN APPLE COMPUTER AND VOLKSWAGEN TEAM UP TO CREATE A TELEVISION SPOT AND OFFER YOU THE GIG TO DO THE VISUAL EFFECTS WORK ON IT, WHAT DO YOU DO? YOU DON'T BLINK, OR WIPE UP THE COFFEE THAT YOU JUST SPIT ALL OVER YOUR DESK. YOU SIMPLY SMILE AND ASK, "WHEN DO WE START?"**

One would figure that these two mega-companies, which don't just sell products, but are cultural icons, could have their pick of the vfx litter. After all, "Pods Unite" is one of their biggest ad campaigns: Buy a new German-engineered Volkswagen Beetle, complete with avant garde stylings, and get the latest rage in digital music players - Apple's new

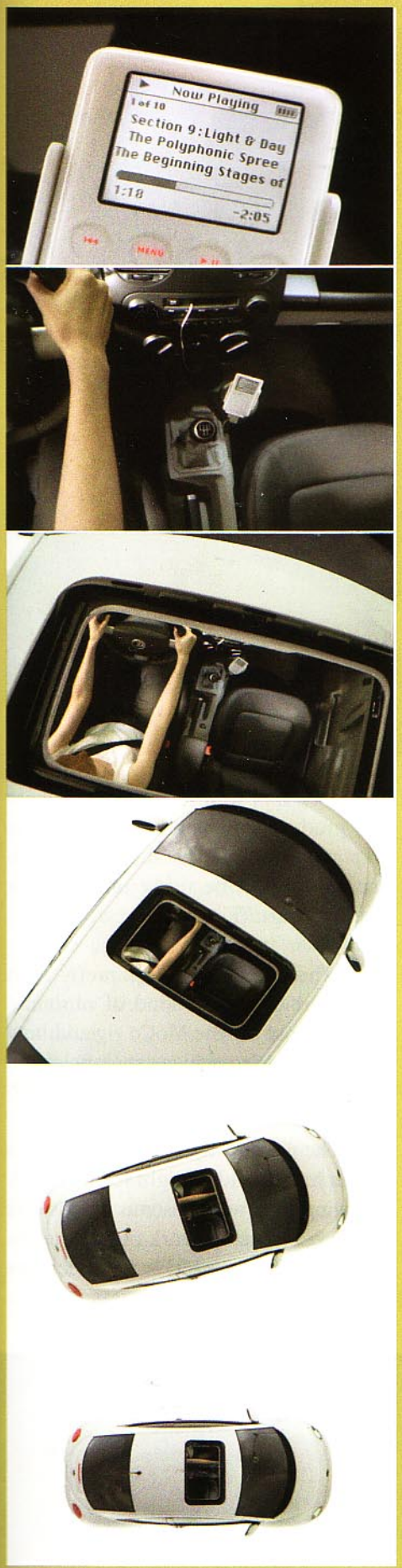



FIGURE 2 - THE CAMERA BEGINS TIGHT ON THE IPOD AND PULLS THROUGH THE ROOF AND HIGH ABOVE THE CAR.

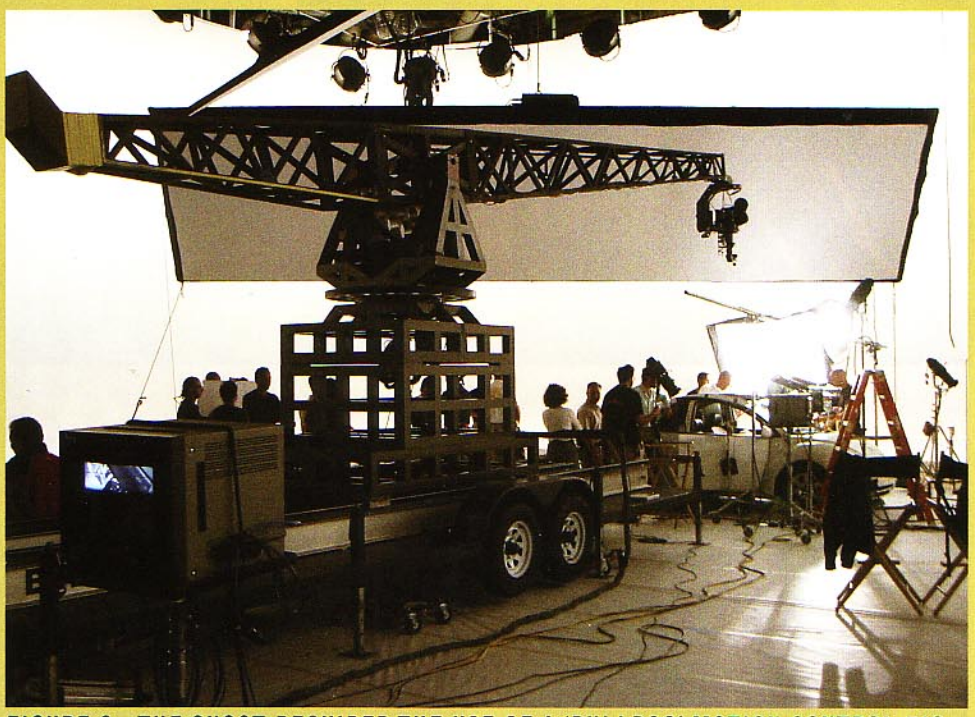


FIGURE 3 - THE SHOOT REQUIRED THE USE OF A 'BULLDOG' MOTION CONTROL RIG PROVIDED BY IMAGE G.

FIGURE 4 - THE CREW PREPARE THE MOTION CONTROL RIG FOR THE "CORKSCREW" CAM THROUGH THE CAR

iPod. Look past the heavy hitters in LA and New York; those effects companies didn't get the job. The company that did get the gig is the industry's best kept secret, Brickyard VFX, located in Boston, Massachusetts.

Brickyard is a small boutique shop that specializes in digital finishing, effects and compositing using Flame and Combustion. One of their main tasks for this VW-iPod TV spot was to achieve a "corkscrew" camera shot

FIGURE 5

FIGURE 7

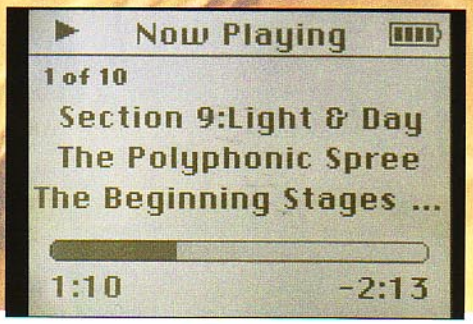



FIGURE 9

FIGURE 6

FIGURE 8

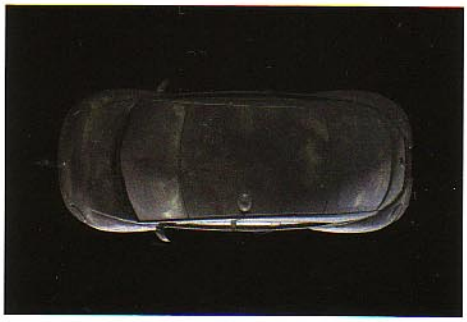


FIGURE 10

FIGURE 11

beginning on a tightly framed iPod inside the VW Beetle. (See Fig. 1) The camera would then slowly pull away from the console, up and out through the roof, twirling higher and higher until the entire car is revealed. (Fig.2) It sounds easy and Brickyard's flawless work makes it look easy ... but we all know better. Getting a camera attached to a spinning crane through a little car sunroof is impossible. So, I set out to see just how they pulled this off, and what other tricks they used to bring the Beetle and the iPod together.

Dave Waller (Brickyard's founder) and Brian Drewes (vfx producer) graciously invited me down to their studio on Newbury Street to chat with their lead artist on this project, Geoff McAuliffe. I soon learned that this TV commercial entailed 200 hours of digital work and involved the use of motion control, CG models, reflection mapping, 2D compositing and 3D tracking. Geoff was eager to share his experiences on this daunting project.

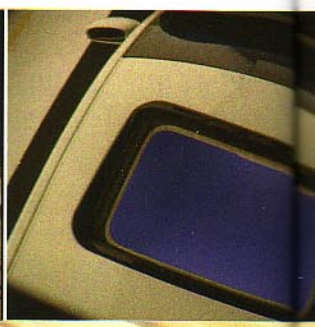
HIGHEND MAGAZINE: What made you decide to use motion control for this corkscrew camera move and where did you shoot it?

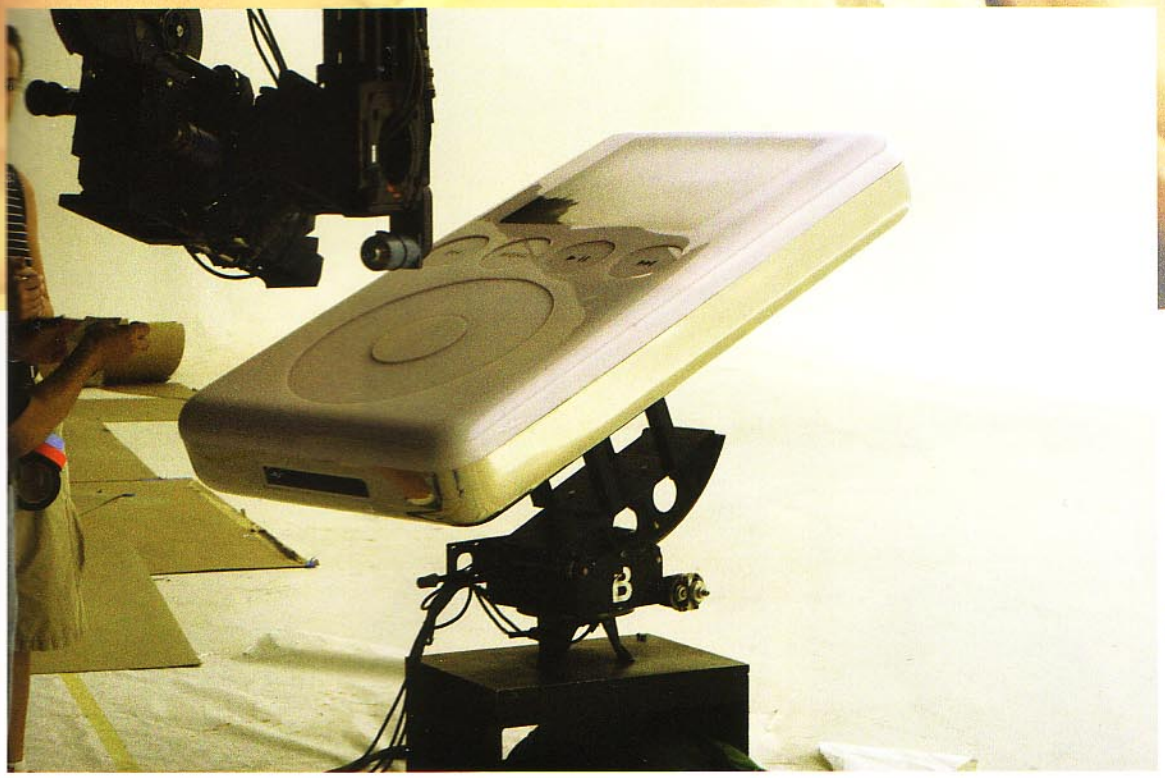
GEOFF MCAULIFFE: After looking at the boards and talking with the agency, it was apparent that motion control would be needed to accomplish this shot. We shot it in Los Angeles, where the production company enlisted the help of 'Image G' as the MoCo vendor. We would be using one of their 'Bulldog' rigs. (Fig. 3)

HM: The camera tilts up pretty high above the car. What kind of obstacles did that pose for the MoCo rig and how were you able to achieve such height in the confined studio space you were in?

GM: The director felt that shooting on a 14mm lens would be right for this shot. Knowing this, we did some rough tests on the computer to work out the height that the camera would have to achieve to get the shot.

BEFORE AND AFTER COLLAGE OF CAR COMP INCLUDING MATTE ELEMENTS





AN OVERSIZED MODEL OF THE APPLE IPOD WAS USED TO ALLOW FOR CLOSE UP CAMERA SWEEPS.

After our tests, we realized that we would not be able to get to the optimal height (over 30 feet). We were able to get the camera to 20 feet maximum, (Fig. 4) and then we would have to 'continue' the move after the shoot in computer.

The corkscrew was shot in several different passes for ease of lighting, and the small fact that the camera and rig could not fit through the sunroof into the car, so we had to remove the roof for any interior shots.

HM: There certainly are a lot of variables there that could contribute to some stabilization problems. How were you able to obtain such a fluid camera move and cover all that distance in only a few seconds?

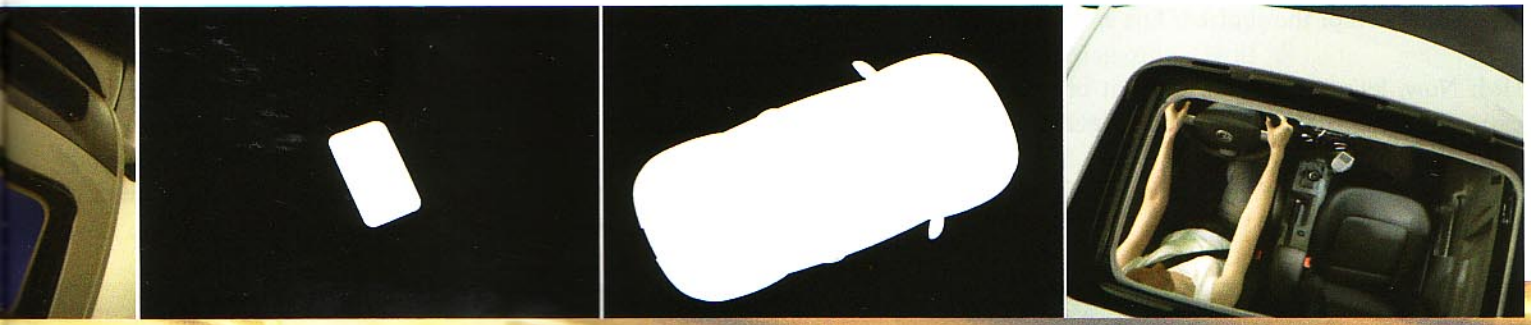
GM: There was a limit to the speed that the MoCo rig could travel without getting undesirable wobbles, etc., so we shot the move at one-third speed at 15 fps. Discarding the extra frames in post would give us our move at the correct speed when running at 30 fps. We were able to preview the move at (sped up) real-time on set through digital video playback to check that the take was okay.

HM: Please describe the passes that were shot and generated that make up the elements.

GM: We shot;

-A beauty pass on the car interior with driver in-situ. (Fig. 5)

-A tracking pass with no driver, tracking marks on the iPod face and lighting change to take unwanted kicks off the gear shifter which we would strip in later. (Fig 6)



-A beauty pass on the iPod, with its own lighting rig. (Fig. 7)

-A beauty pass on the car exterior with the sunroof open and blue screen filling the aperture. (Fig. 8)

-A screen insert shot for the iPod. (Fig. 9)

-We also shot a pass of the sunroof closing. (The client later decided that the sunroof should remain open for the entire duration of the shot.)

HM: Now, let's discuss the amount of extra finessing that was needed to get these elements ready to composite. I understand you used Flame for most of the compositing work?

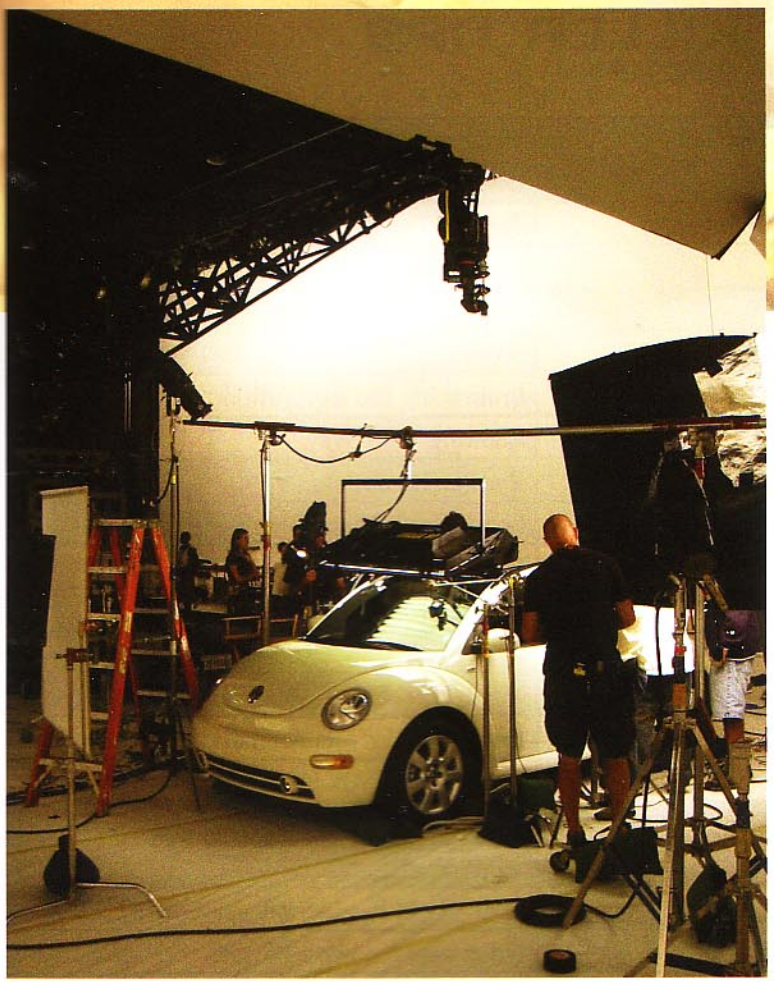
GM: Yes, that's right. Once we got the elements into Flame, we could see there were some minor camera shakes and inconsistencies that we needed to remedy. The client had also requested that we do a huge blow-up of 186% on the head of the shot (the iPod face), so we needed to get a high-res transfer that we could transition into the NTSC footage. Not a problem! Flame 8 will happily accept any resolutions into the same partition. I used HiRez to comp, and output at NTSC.

HM: You mentioned previously that you continued the camera move in the computer. Can you elaborate on that?

GM: We really needed to add extra camera height to the shot and add extra spin to the car. This proved to be tricky, as the camera rig changed speed slightly towards the end of its move in both rotation and boom, so I couldn't simply pick up the move where the camera ended.

HM: That must've posed a challenge. How were you able to extend the MoCo camera move into a digital camera move?

GM: I had to stabilize out the rotation and scale of the MoCo move so that I could re-animate the car flawlessly in Flame. A little time consuming, but worth the effort! Once this was achieved I was able to gain full control over the movement of the car, so I could repo and spin to the client's content.



HM: Ever considered 3D tracking as a solution?

GM: I had originally intended to utilize the data from the MoCo rig to assist in tracking or in 3D, but I was able to generate a good track to our stabilized elements with Boujou, a most excellent software. This I used to composite the iPod screen readout. We had shot the screen insert on the stage, but later we were able to make a much cleaner element by shooting HiRez digital stills and adding animation.

HM: There are some nice street and cloud reflections on the car and it's apparent some aesthetic liberties were taken here, because the car is not on a street at all but in an all-white environment. It is interesting visually and I was wondering how the decision to do that came about?

GM: The client wanted to maintain an ethereal, spaceless quality to the spot. We tried a hint of road surface but it grounded the car and some magic was lost. We settled on reflections of sky and lampposts to suggest some forward motion (we hinted at the lampposts in a previous shot). After picking a nice sky, I modeled some lampposts in LightWave. I imported a VW Beetle model into LightWave and animated a simple scene, with the car traveling past staggered lampposts. The resulting render was included as part of the final Flame composite. (Fig. 10)

HM: Any finishing touches you'd like to share?

GM: Well, with all of the elements together now, all that was needed was some color tweaks to the car exterior, and some localized color correction to areas inside of the car. Film grain was added back in to marry the comp, a nice simple graphic and ripple were added to the end and voilà! (Fig. 11)

HM: Thank you for contributing your time and sharing your tips with Highend Magazine, Geoff.

GM: You're quite welcome.

# Book Preview: MAYA 5 KILLER TIPS

ERIC HANSON, AUTHOR
ERICK MILLER AND ERIC PENDER, CONTRIBUTORS
PUBLISHER: NEW RIDERS; (DECEMBER 19, 2003)
ISBN: 0735713731

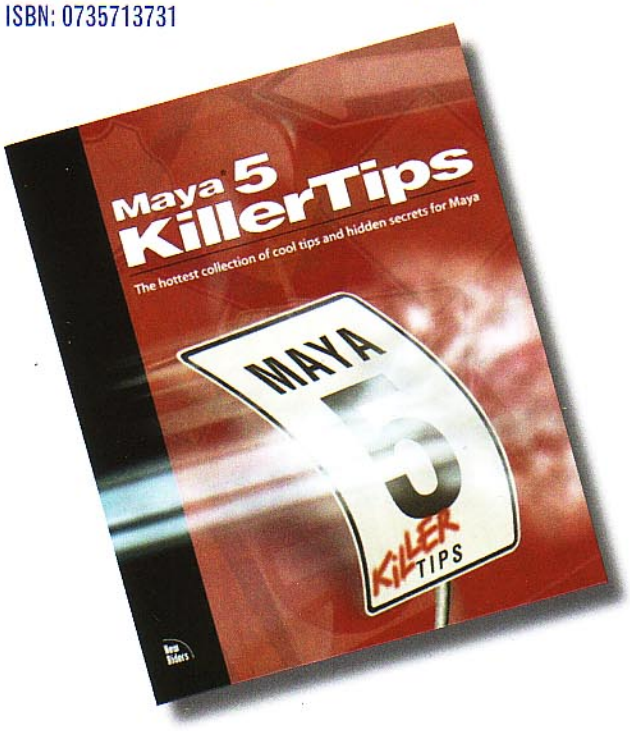



TABLE OF CONTENTS


| | |
|---|---|
| Chapter 1 | Deciphering The Hieroglyphics |
| | Unearthing the Secrets of the Maya |
| Chapter 2 | Taming The Beast |
| | Customizing Your Interface |
| Chapter 3 | The Glamorous World of Modeling |
| | Work Smarter, Not Harder |
| Chapter 4 | Embracing the Revolution |
| | Lighting Your Way |
| Chapter 5 | Rags to Rendering |
| | Getting a Grip on Shading |
| Chapter 6 | From Home Movies to Hollywood |
| | Camerawork Basics |
| Chapter 7 | Building Character (Animation That Is) |
| | More Than Skin-Deep |
| Chapter 8 | Dangerous Effects Animation |
| | Things You CAN Do at Home |
| Chapter 9 | Getting Under the Hood |
| | Mud Wrestling with MEL |
| Chapter 10 | Real World Production |
| | Otherwise Known as Stress |


**3D artists have it tough.** We wade through seemingly endless amounts of technical minutia from dreary manuals, off-the-shelf books thicker than the yellow pages in a grimy Manhattan phone booth, and well-intentioned but often awkward online tutorials. We then navigate bleary-eyed through the endless layering of cryptic commands in our applications, wrestling such terms as "Directed Acyclic Graph," "Non-Rational Uniform B-Spline" and "Non-Manifold Topology."

We endure this toil in order to satisfy our implacable creative urges to muster up fascinating imagery out of the Dark Well of Technology. 3D animation and rendering is certainly the high ground of computer graphics, pushing one's capacity for technical concepts and jargon to the limit. And, like your grandfather in his youth mythically trudging through miles of snow on his way to school, we like it that way. So, what better reason than to create a book that assembles concise technical tips in order to speed understanding of the arcane, accelerate your workflow, and peer into techniques professionals use daily in the visual effects field? When I learned of Scott Kelby's "Killer Tips" series, made popular in the publishing crowd, I recognized that it would be an instant fit within our 3D CGI world, like beer and pretzels, or in our case, hot pockets and programmers. Can you get a job working with Steven Spielberg or Jim Cameron after you read it? Why not, stranger things have happened in Hollywood. Steven Spielberg started as a squatter in a studio lot, and Jim Cameron was originally a truck driver. I would say with that in mind, you yourself can certainly help make a mark in Hollywood armed with the aid of the book. One of the truly great aspects of the visual effects field is that ultimately it is only about your talent. Well, sure, there are some power lunches involved, but mastering a difficult package like Maya is a surefire and road-tested route to contribute to the history of the silver screen, and not a bad 8-5 gig. Well, perhaps more like an 10-12 gig ... In any case, here is a preview of a few of the tips contained in the book, set to release on December 19.

## 1. Use The Fields, Luke

Selecting objects are one of the key Maya skills in life, like learning to not spill on yourself during meals. For those unruly cases where you have 5,000 elements called fooThing01 through fooThing5000, you could lasso all successive nodes in the outliner, but it is much more expedient to type this bit of MEL scripting in the script window: select " fooThing*". Quicker yet is to type the same phrase sans the " " into the selection entry field in the main menu strip for ultra-deluxe picking speed!

## 2. Strawberry Outliner Fields Forever

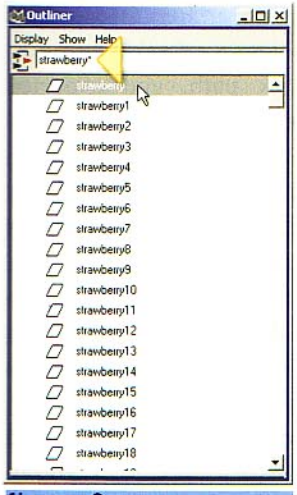



figure 2

Another great place to practice your fields is the unnamed entry bar at the top of the Outliner. Rather than tediously hunting for a particular name nested within an extensive list of nodes, type in wildcards of your choice to limit what the outliner will now return back to you. Hit the button to the left of the field to return back to the full listing.

## 3. Field Operators Are Standing By

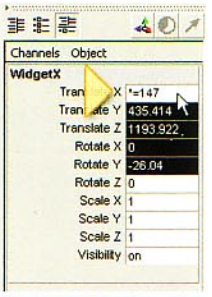



figure 3

Here is a hot tip for those who have had trouble with basic arithmetic past grade school. Let's say that random transform values are set in the attribute fields of an object, but you are asked to alter it by 147% (150% would be too easy in this case). Rather than take time to break out what my very "old school" engineering professor used to call those new "electronic hand-held calculators," you can highlight the field(s) in question and use *=147 to magically multiply all the values by that amount. Other arithmetic operators such as +=, -= and /= can all be used to avoid that handheld marvel of technology!

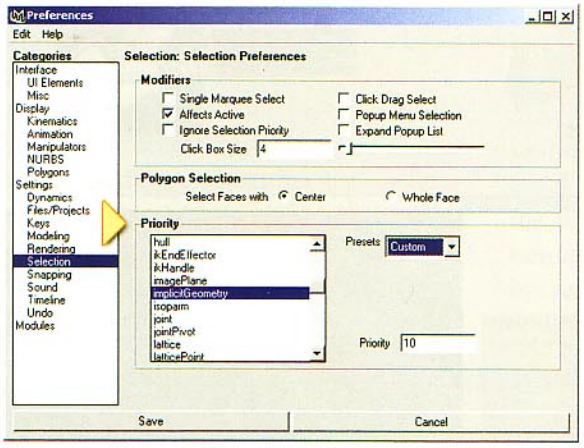



figure 4

## 4. Top Priority Clearance

Maya has a dirty secret — it gives preferential treatment to NURBS over polys in priority of what gets selected when clicking on overlapping objects. Shocking I know, but you can actually choose which gets picked first from Window, Settings/Preferences, Preferences, Settings, Selection, Priority. Just pick an entity type, choose Custom Preset, and raise or lower priority to your liking. At last, equal rights for Implicit Geometry!

## 5. Randomize Those CV's

Here is a great trick for deforming objects without deformers. Say what? Actually it's a very easy way to slightly perturb surfaces without the time to tweak a deforming operation. Simply take a NURBS or poly surface which you want to randomize, select all CVs or vertices and set a Keyframe on them. Now open the Graph Editor and use a MEL randomizing script such as RandKey.mel found online at a site such as Highend3d.com. With the script, randomize the value, not time, and update on the Time Slider. The surface will either be too subtle or totally wrecked, so adjust accordingly. Once to your liking, delete animation on the surface, as you will be carrying a lot of animation baggage you don't need.

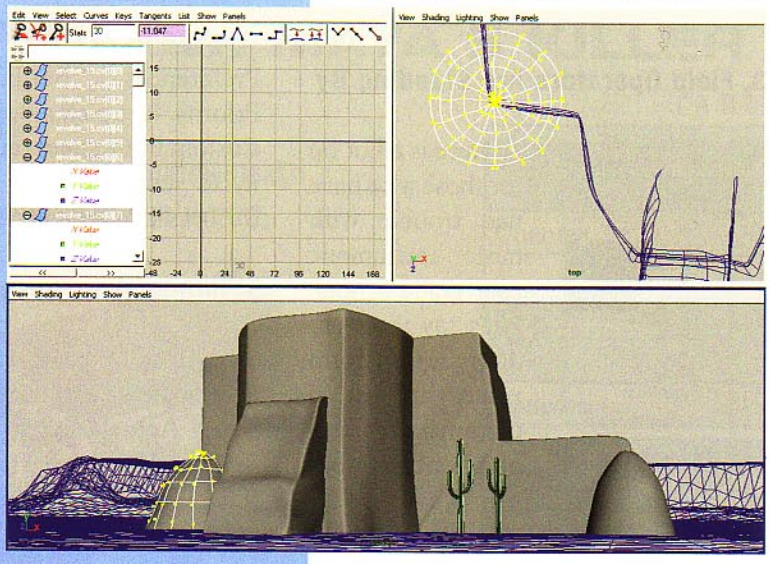


figure 5

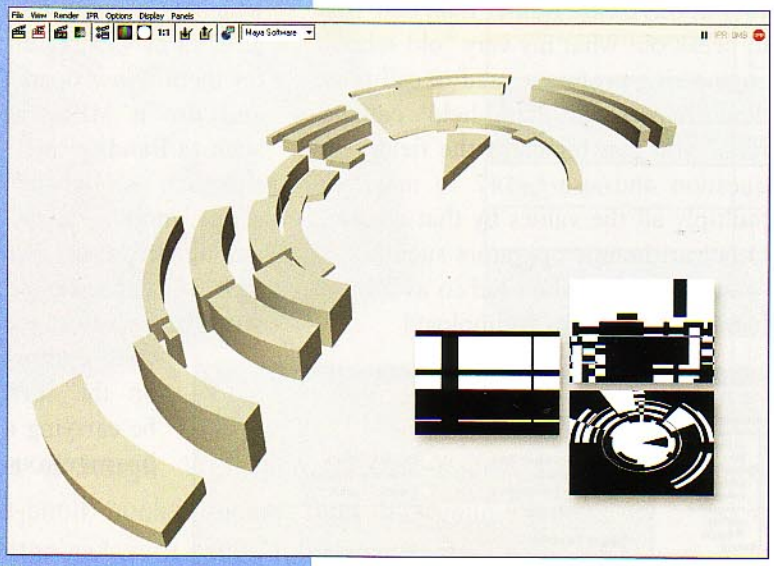


figure 6

figure 7

## 6. Use of Photoshop as a Modeling Tool

Photoshop a 3D tool, you say? No, it is not a sneak preview of version 10.0, but rather using an often overlooked filter in Photoshop since v1.0, Filters, Distort, Wave. Sometimes intricate design work needs to be generated for 3D modeling details, and this trick can speed up that process. First, start by drawing basic linework in Photoshop using marquees or lines. Then use the Wave filter with very little Number of Generators, perhaps 1-2. Choose Square as the Wave shape, and Repeat Edge Pixels. You should have a nice garbled and unusable image at this point. To make it shine, turn the Wavelength and Amplitude down very low and tweak as desired. Once an aesthetically interesting pattern is formed, this can be transformed into Maya curves by converting it into a selection and then a resultant work path, and exported out as an .ai or Adobe Illustrator file. It can now be imported in your model and extruded into what the FX industry affectionately calls "Greeblies," or sundry miscellaneous tech detail.

## 7. Cookies and Gobos

Sounding like something from your childhood, these are rather old Hollywood terms for ways of manipulating the spread of light. Cookies and gobos are actually paddles or cards with random holes cut out to deflect key lights into a draping of light and shadow, or to soften the spread. Defining the signature look of the film noir period, they are useful for dramatic effects or simply to richen up a scene. Cardinal Rule #1 in CGI is that Imperfection Is Your Goal, so here is a great way to emulate that technique using a simple CGI spotlight. First, paint a few random black and white organic patterns in Photoshop and save out as a tiff file. Now, create a spotlight in Maya and in the Attribute Editor, map a file texture

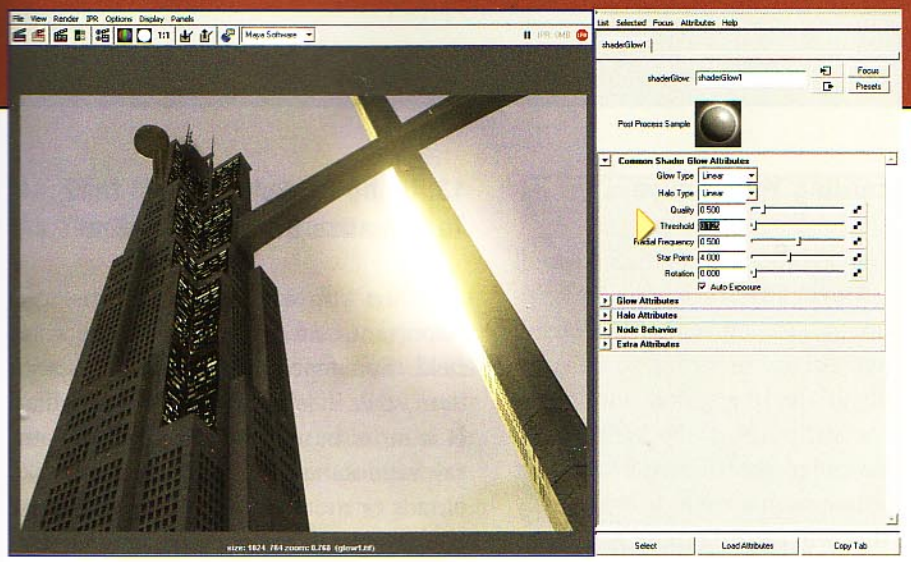


figure 8

Ramp Texture on the Color Offset can taper it down to create drip marks. Age can be accelerated by using a subtle noisy hue Ramp Texture mapped to the Color Offset of the color map to discolor and slightly yellow. Color Remap can be used to further alter the original color spectrum. Erosion can be made on the Bump Map with a Color Remap as well, altering and intensifying the Bump Map. Rust and iridescent oil can be created by color mapping a noisy hue Ramp Texture on top of the rest using another Layer in the shader. The beauty of using solid procedurals in all this is that they will not display UV mapping artifacts and will scale properly in distance. Who knew you would grow to love dirt this much?

to the color channel. Choose a gobo file of your choice and adjust Filter Offset in the Spotlight Attribute Editor to add blur and softening to the sharp contrast. Further interest can be made by adding warm hue into the Texture File Color Gain and cool hue to Color Offset.

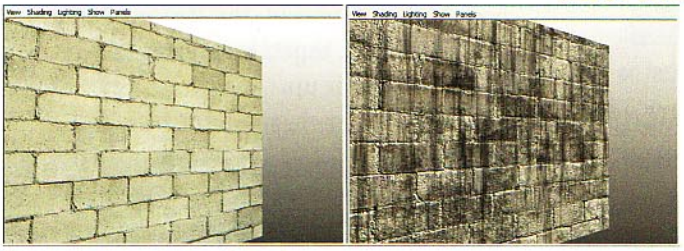


figure 9

## 8. Use of Thresholded Shader Glows

Shader Glow is a common effect added to a surface that aids the appearance of self-illumination, such as a glowing television screen or car headlight. It adds the appearance of the "blooming" of light due to high contrast or strong intensity. The problem arises as the entire surface, by default, glows equally, creating the look of an unfortunate nuclear accident wherever you apply it. A little known control exists in the Shader Glow global attributes, found as an extra shader ball in the Multilister or Hypershade. Locate Threshold in the Common Shader Glow Attributes, and you will find that as it is raised, only the specular highlights will then bound the apparent glow effect. The end benefits of this technique are to show a light blooming effect in your renders and impress others at cocktail parties.

## 9. Embracing Dirt and Grime

Dirt and grime are not only an unavoidable part of life (especially with kids), but are actually encouraged in CG rendering. A common method is to dirty up your texture maps, but an easier and more spatially coherent method is to use 3D procedural methods in Maya's shader networks. Create a basic Lambert shader with a dark black color, and then use a 3D Solid Fractal Texture as a Transparency Map. Use this shader as an overlay to any other base shader underlay in a Layered Shader. Adjust the Ripples of the 3D Fractal Texture to 1.0, 0.1, 0.1 to get the proper vertical streaking and density. Use of a grayscale

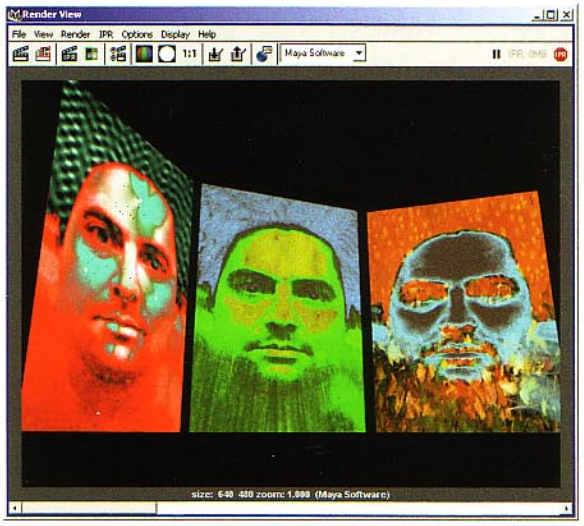


figure 10

## 10. Color Remapping

Color Remapping is a great command tucked away that many pass up on their road to rendering. Color Remap is primarily used for replacing a texture's tonal range with one from another file for surreal or abstract effects. For example, green alien skin can be made by remapping a facial skin tone map with lizard skin. The hue and value of the original file are remapped to a color ramp, with hue assigned to U and value to V. The original color hue is lost as it connects to a V ramp, but new color values or textures can be substituted. Start with an image assigned to a File Texture, then in its Attribute Editor, choose Effects, Insert Color Remap. A Ramp Texture is then displayed, altering the color of the original texture. Reduce the ramp to a black to white grayscale, and observe that the tonal range is alterable similar to the curves function

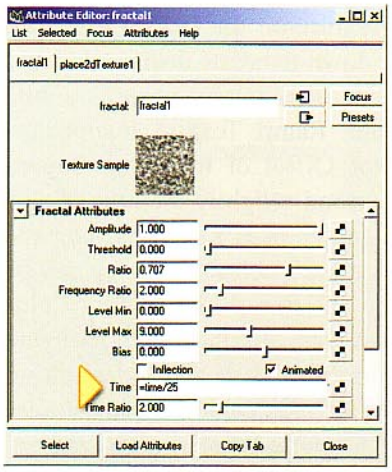



figure 11

in Photoshop. Now, add multiple sliders with various hues or textures mapped to them, and the fun begins. This is a great way to age, alter, or discolor exisiting maps, or create a little psychedelia along the way.

### 11. Texture Map Your Animation

Here is a genuine killer tip for accessing random functions of 2D textures straight into animation channels of objects. Start by creating a sphere and opening the Attribute Editor, as this hook will not be accessible through the Channel Box attribute fields. Right mouse click in the Translate Y channel field, and choose Create New Texture. Pick a 2D Fractal Texture and turn on the Animated check box In Fractal Attributes. Playback of the timeline will not yet create motion, though, as we need to provide keyframing of the Time field or simply place a simple expression such as =time/25 directly into the field. Once that is entered, play back for erratic motion. Now, modulate fractal parameters such as Ratio for smoothness or Alpha Gain or Offset for intensity. Other options exist similarly with the 2D Noise texture, which has interesting options such as Perlin, Billow, Wave, or the intriguingly named Space-Time.

### 12. Recording Your Mouse

Here is a hidden gem tucked well away in Maya. Often it could be helpful to use mouse movement to create animation curves in order to provide a "handheld" feel or erratic motion to objects or attributes. Deep inside MEL is a node called the Record Node that will facilitate such a trick. It is basically a node that will record mouse actions as keys and pipe those curves to another node. Once captured, the Record Node can be deleted. Start by entering this bit 'o MEL to set it up:

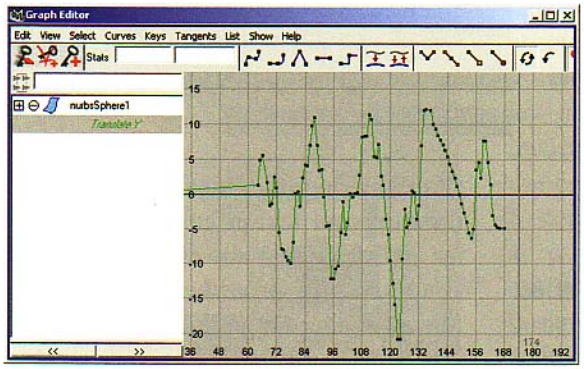



figure 12

```
sphere;
createNode record;
connectAttr
nurbsSphere1.ty
record1.input;
```

Now set your playback speed in the Animation Preferences to 15 fps in order to not outdistance the record process. Begin the capture with this command:

```
play-record;
```

Other attributes can have concurrent connections as well to save cut and pasting of curves. Bear in mind the attributes chosen can be broader than just translations.

### 13. Using Particles To Visualize Noise vs Random

When you are new to effects, one important concept to grasp is the noise field. Some people describe it as random, but it is actually not quite that. It is more based on a predictable natural randomness, which can look like clouds or mountains depending on how you use it. Here is a simple way to visualize the difference. Make a grid of particles with the Particles, Particle Tool options. Turn on Placement, With Text Fields, then set the Minimum Corner to be 10,0,-10, the Maximum Corner to be 10,0,10, and the Particle Spacing to be .2 . This will give you a plane of particles. Now select the Attribute Editor for the particles and under the Per Particle Attributes right mouse click on Position and select Runtime Expression. Enter this:

```
position+=<<0,noise(position),0>>;
```

Play a single frame only by keying in 2 in the current frame. Now you will see the particles raise or lower into a hilly pattern. Rewind the scene and replace the rule with:

```
position+=<<0,rand(1),0>>;
```

Now play one frame. Notice how the rand calculation is much more random while the noise is like periodic rolling hills.

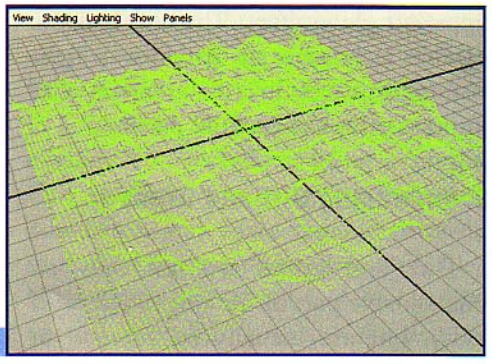



figure 13

## 14. Emit More Particles As The Emitter Goes Faster

Here is a fairly advanced trick you may find a use for. Maya typically does not let you vary emission rates with velocity, but this will allow such an effect. Create a sphere and add an expression to its translationX attribute, tx=sin(time*3). As the scene is played, the ball should oscillate back and forth smoothly along X. With the ball selected, choose Particles, Emit From Object options, setting it as a Surface Emitter Type, setting the Rate (Particles/Sec) to 2000, and turning the speed to 0. In the particle1 Attribute Editor, turn Inherit Factor to 1 under Emission Attributes. Now, under Lifespan Attributes, select lifespanPP only. Go down to the Per Particle (Array) Attributes and right mouse click on lifespanPP and select Runtime Expression. Enter this expression:

```
$magv=mag(velocity);
if (rand(1)<linstep(0,1,$magv/3))
lifespanPP=1;
else
lifespanPP=0;
```

Now when you play the scene you will see that the emission happens a lot more when the object is moving fast. As the object slows down, there is very little emission taking place. To apply this to any other scene, you will need to know how fast the cutoff velocity is. To change that, change the numeral 3 under $magv/3 to another integer value. The trick only works due to Maya letting us see the inherited velocity in the Creation Expression. Perfect for salt shaker or ketchup bottle animation!

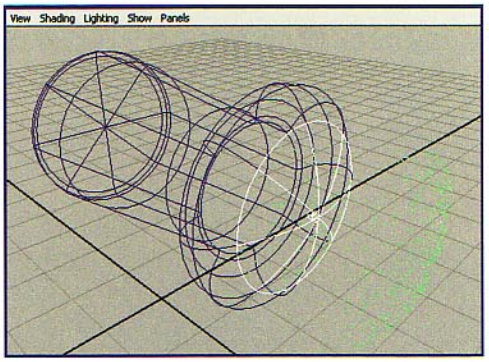



figure 14

## 15. Rendering Larger Than 8K

It seems there is always someone out there wondering how to create a rendering for a 5' x 10' poster at 300 dpi. This is generally a bad idea, as higher dpi's are normally set for small print work, not billboards. But, there are occasional uses for very large renders for illustration work or fine art. Maya tops out at an 8k x 8k resolution, so if more resolution is desired, here is a clever workaround. In the command line render options, there exist flags for x and y resolution, but also for sub-regions of the overall field (xLeft, xRight, yHigh, yLow). Maya only cares about the 8k limit for its current render, so by breaking up a render into smaller sub-region components, it can be assembled in a paint package to the full resolution. So, for a 10,000 x 5,000 pixel render, the usage command for the two would be:

```
Render –x 10000 –y 5000 –xl 0 –xr 5000
–yl 0 –yh 5000 –p first_piece scene.mb
Render –x 10000 –y 5000 –xl 5001 –xr
10000 –yl 0 –yh 5000 –p second_piece
scene.mb
```

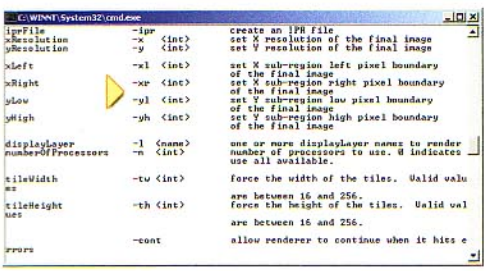



figure 15

## 16. A Final Render Checklist

As a summary of some of the many tips in the book, here is a checklist I use frequently before an expensive overnight render at the eleventh hour of a project, along with throwing a chicken bone over my left shoulder for good luck:

- ❒ Is all geometry as optimal as possible (no wasted, flat, or excessive CV's or faces)?
- ❒ Is NURBS tessellation all set to Advanced and adjusted to screen space?
- ❒ Is camera clipping not exceeding a 0.1 to 20,000 ratio?
- ❒ Is Auto render Clip Plane turned off in the camera attributes?
- ❒ Is bump mapping depth set around .1-.2 versus 1?
- ❒ Are file textures filtering set not below 1?
- ❒ Has a test render of 2-3 frames been done to check texture aliasing?
- ❒ Is Reuse Shadow Maps used for static shadows?
- ❒ Is Dmap Auto Focus turned off and set manually in spotlight shadows?
- ❒ Are all history and construction curves deleted?
- ❒ Has File, Optimize Scene Size been done?
- ❒ Have Multilister, Edit, Delete Unused been done?
- ❒ Are BOT textures used?
- ❒ Are textures sampled down to screen space and at powers of 2?
- ❒ Is animation padding set to 4?
- ❒ Do you feel lucky today?

# Gallery

To have your work considered please submit to
**www.highend3d.com/artists/add.3d**

***Park Woo Young***

**Workplace:** Korea **E-mail:** aici8@hanmail.net
**Software used:** 3ds max 4 **Special techniques:** Final Render use
**Hardware:** Pen III 256 ram
**Render time:** 15 min

# Outcast


***Heiko Tetzner***

**Workplace:** Germany/Student
**E-mail:** wireframe@ray-trace.de
**Highend Username:** Wireframe or Acid

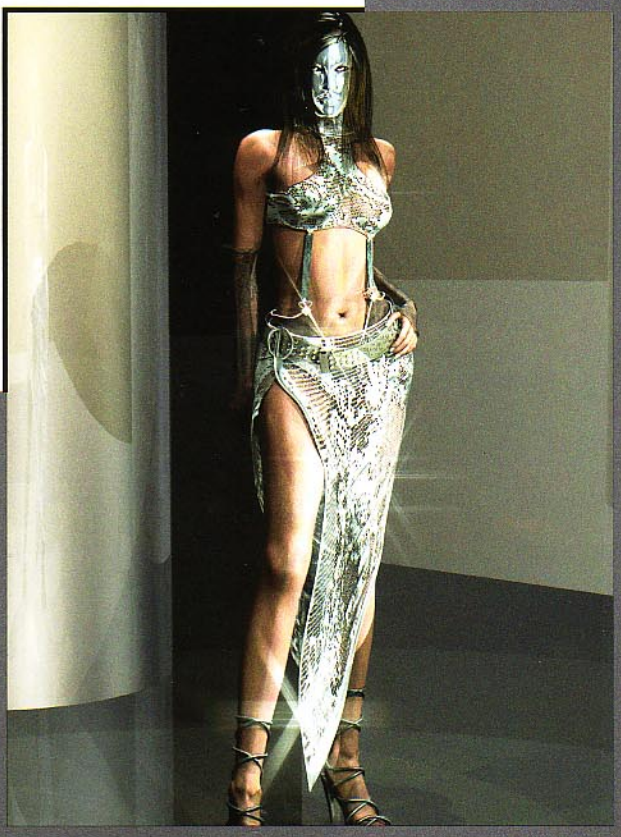
**Software used:** Maxon Cinema XL 7.3
**Special techniques:** Photoshop
**Hardware:** Athlon 1GHz, 320 Ram, Geforce 3
**Render time:** 1 hour
**More process:** 2 hours - Photoshop
**Complete process, design and modeling**: 1 Week

## *Ryan Church*

**Workplace:** conceptual art supervisor for Star Wars EP3
**Software used:** Corel Painter

## *Pavel*

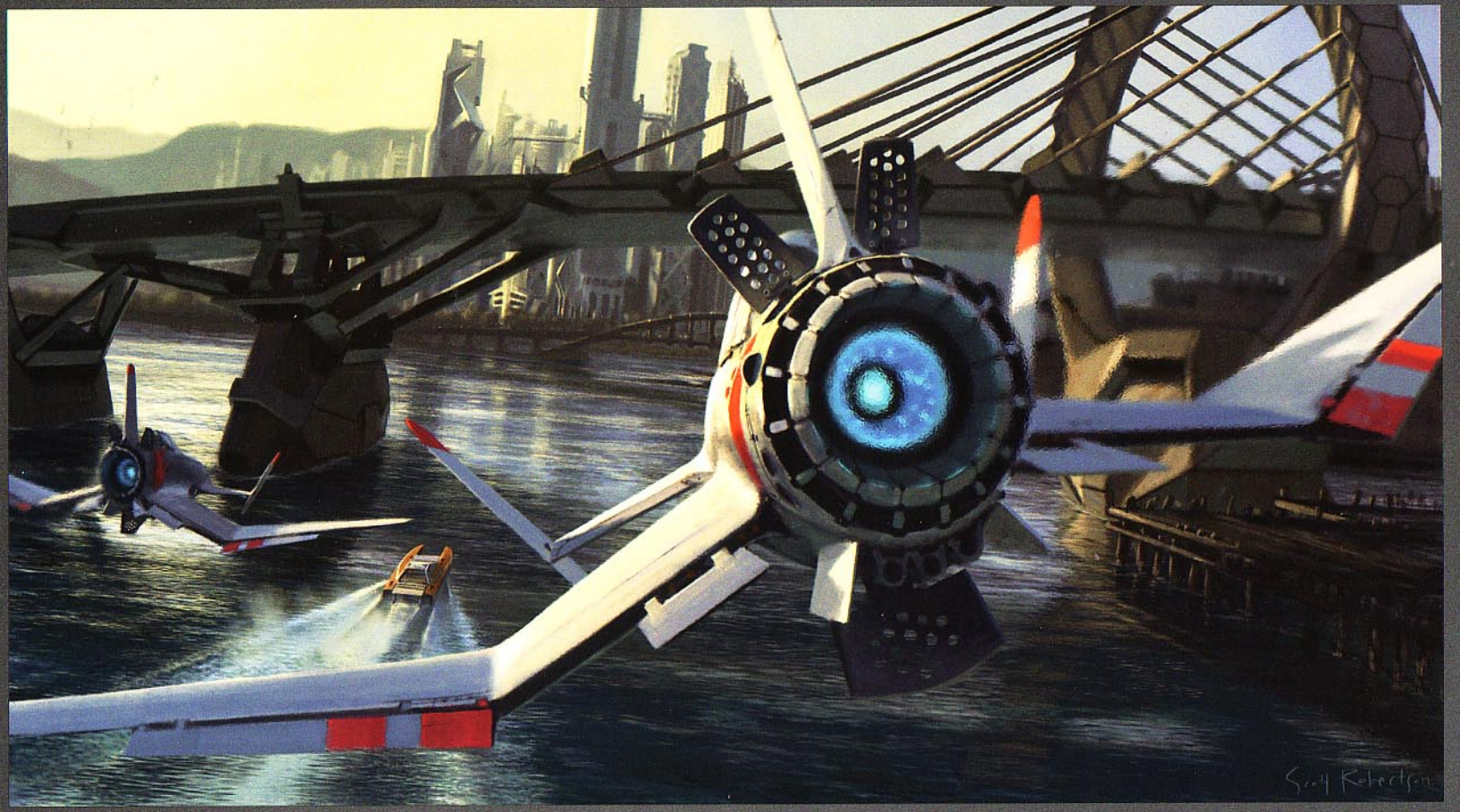
**Workplace:** Student
**E-mail:** mpavlos@mail.ru
**Highend Username:** MPavlos
**Software used:** 3D Max5.0, Rhino2.0
**Special techniques:** NURBS modeling in Rhino, GI, HDRI envir., Motion blur
**Hardware:** Cel 2.2Gh/ 512 DDR/ ATI radeon 64(7500)/ MuthB Asus P4B/
**Render time:** about 1.5 hrs with Brazil 1.0.3



## *Shinya*

**Workplace:** JAPAN
**E-mail:** arrow@kcn.ne.jp
**Highend Username:** shinya
**Software used:** 3ds max 4
**Special techniques:** polymodeling in 3ds max 4.
Hair is made by Shag:Hair.
**Hardware:** Windows2000, Dual 1GHz, 756MB

**Software used:** Shade 6 >
**Special techniques:** none
**Hardware:** Windows2000, Dual 1GHz, 756MB




***Scott Robertson***

**Workplace:** leading concept/industrial designer in LA
**Software:** Photoshop

# CHARACTER DESIGN – DETAIL MANAGEMENT: PART III

BY PAUL HOURMOUZIS

**IN MY PREVIOUS ARTICLE, I PRESENTED SOME THEORY ON HOW TO MANAGE HIGHLY DETAILED CHARACTERS THROUGH A PRODUCTION PIPELINE.**

**THE PREMISE WAS THAT A LOW-RESOLUTION CHARACTER IS EASIER TO SKIN AND ANIMATE THAN A HIGHER RESOLUTION COUNTERPART. THEN, BY USING THE APPROPRIATE TEXTURE MAPS, YOUR LOW-RESOLUTION CHARACTER WOULD RENDER WITH THE DETAIL OF A HIGH-RESOLUTION VERSION.**

**To recap, there were two methods presented:**

The first method was based upon the use of laser-scanned characters, where, using a product called CySlice, you could reduce the mesh density and collect the inherent detail in the form of texture (normal, displacement and color) maps.

The second method began at the other end of the spectrum whereby you started your character with simple primitives, sculpted then detailed it in a package such as ZBrush. You then take low and high-resolution versions of your character, and, using freely available utilities, or CySlice, you are able to obtain the same maps as produced in the first method.

The last step in each method was to apply the maps to low-resolution versions of a character, one that is easy to skin and which exhibits the desired detail when rendered – the difference indiscernible to that of your high-resolution laser scan or digitally sculpted object.

**That was all the theory.**

This article is about practice – a more detailed, tutorial-style look at the workflow. We'll look at some of the recent developments made regarding this technique; consider the use of these maps in shaders in various rendering systems such as a Renderman compliant renderer, mental ray and even Maya's default renderer.

## 1. Recent Advancements

As indicated in Issue 3, Pixologic, the makers of ZBrush, are currently developing some new tools that will greatly accelerate the rate at which this technique is used in real world scenarios.

For modelers and texture artists not already familiar with ZBrush, I highly recommend downloading the free evaluation version and investing the time in getting to know the capabilities of this magnificent package.

Attempting to summarize what ZBrush is in a few paragraphs would never do it justice. Instead, I will focus on the newly available tools that assist with the advancement of this technique and those which help modelers in doing their job with increased speed and creativity.

The most significant new tools in ZBrush are Projection Master, multi-resolution subdivision modeling, integrated displacement and normal map generation and excellent polygon management tools.

ZBrush now supports polygon objects with several million faces, far exceeding the 250K polygon limit in version 1.55b.

Projection Master, a logical extension to the Texture Master ZScript (ZBrush's 3D texturing tool), allows you to paint detail such as muscles, veins, crevices - almost any imaginable detail onto a character. The full range of ZBrush tools, brushes, strokes and alphas can be used when painting detail with Projection Master, giving enormous creative depth.

Multi-resolution modeling is a subdivision modeling system on steroids. This feature allows you to subdivide your model, keeping the ability to traverse the different levels of subdivision and make changes that propagate through to higher levels.

For example, at a low level, you can make changes to a character's head, changing the general shape. You can then switch to higher levels of subdivision and sculpt the detail using the standard toolset and/or Projection Master. You can at any time switch back to a lower level and make sweeping changes, keeping the detail at the higher levels - the history of changes you make at each level is kept.

Multi-resolution subdivision controls take your standard modeling workflow to a new level, dramatically increasing the speed and flexibility to make changes well after you have added detail to a character.

Another new feature of interest is the integrated capability to create both displacement and normal maps within ZBrush. This capability is closely coupled with Project Master and multi-resolution subdivision, as you would typically detail a character for the sole purpose of obtaining these maps. ZBrush allows

Image 1

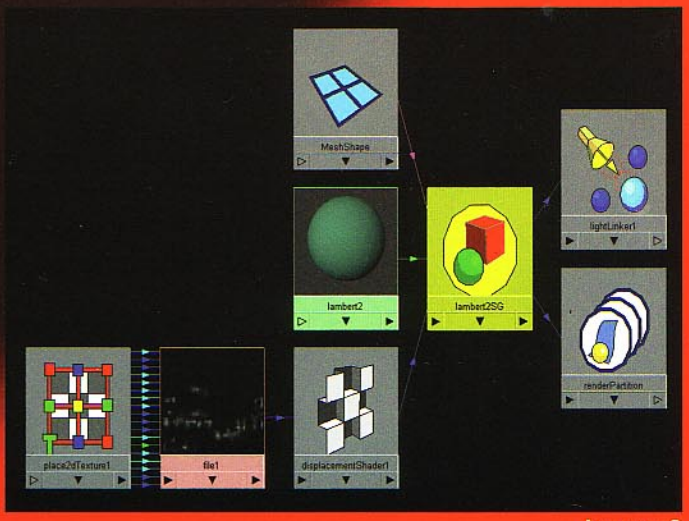



Image 2

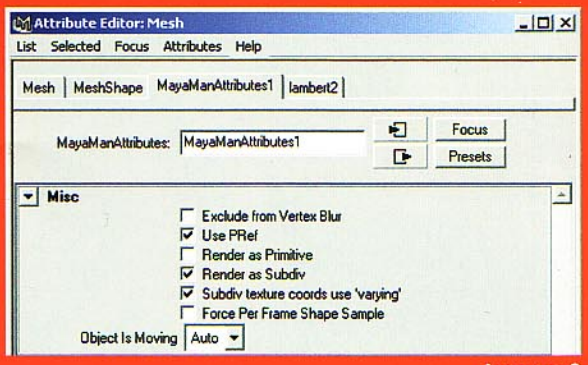



Image 3

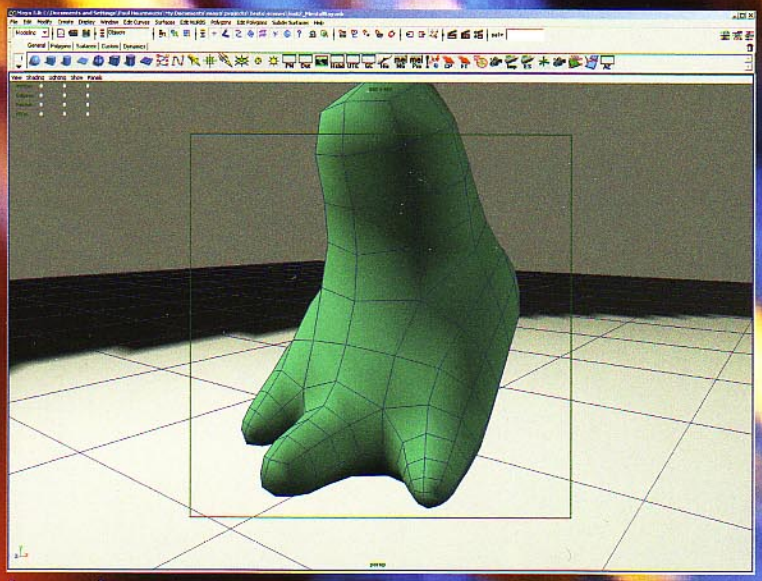


Image 4

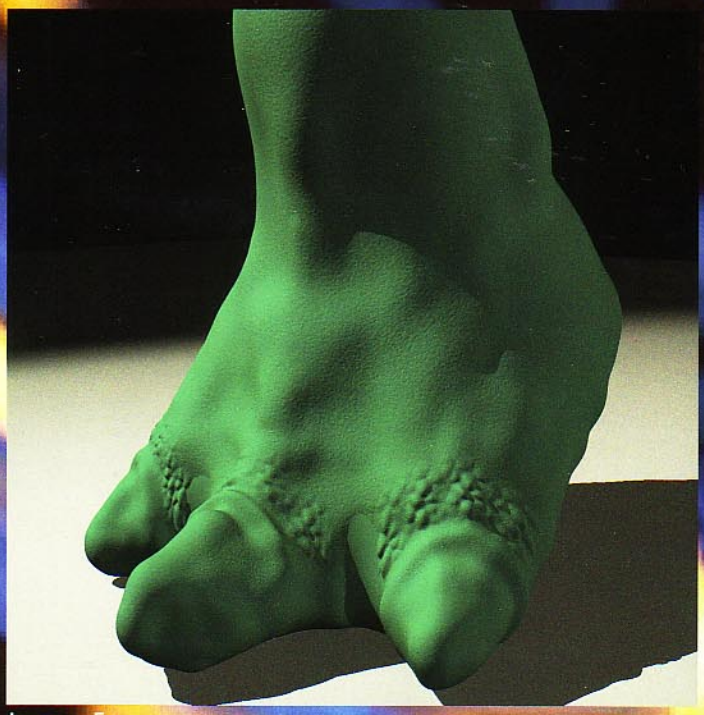


Image 5

Image 6

you to extract 16-bit displacement and normal maps once the detail has been added to your character.

The workflow to create the maps in ZBrush is straightforward:

a. Create a low-resolution version of your character.

b. Detail your model within ZBrush, subdividing along the way as required. You can switch between different resolution levels using multi-resolution controls, making changes to your model as you see fit.

c. After detailing is complete, switch back to the lowest resolution (level 1).

d. Set the required parameters (displacement map size, calculation accuracy), then let ZBrush calculate the maps. You can then export these maps as 16-bit TIFF's for use in your rendering system.

e. Export the lowest level resolution version of your object in .OBJ format and import into a 3D package such as Maya or Softimage|XSI.

Image 1 shows a displacement map for a creature's foot I sculpted in ZBrush:

The final tool of note that greatly assists with this technique is ZBrush's polygon management tools. This includes group management, whereby you can group polygons within an object (default groups are created if your object is made from a ZBrush ZSphere). This allows you to easily hide sections of your character for easier modeling and greater interactive performance. You can also easily hide and unhide polygons via dragging a selection box over the required faces.

Other major new features of ZBrush include a redesigned interface offering

more screen space, extensibility via a new plug-in architecture, new strokes, improved importing and exporting between ZBrush and other 3D applications. With the new features, ZBrush is now a serious film production tool and is unique in many respects.

At the time of writing, ZBrush was in beta testing, however, it should be available for purchase by the time you read this.

## *2. Using the Displacement Maps*

After creating your displacement and normal maps, whether it be from CySlice (see issue 3), ATI's Normal Mapper or from within ZBrush, you then need to incorporate them into shaders. Here, we'll look at the steps you need to take to use these maps with Renderman, mental ray and Maya's default renderer.

### *2.1. Renderman Shaders*

For the purpose of establishing and testing a Renderman based workflow, I used Sitex Graphics AIR renderer in conjunction with Animal Logic's MayaMan. MayaMan allows you to create Renderman shaders on the fly by translating a Maya shading network.

#### *2.1.1. Maya Shader Translation using MayaMan:*

MayaMan does a great job of translating Maya shading networks to real Renderman shaders.

This makes the process somewhat easier than writing your own Renderman shaders, as creating a Maya shading network with displacement is very simple.

However, if you are going to achieve a proper rendered result, there are a few things you need to do before you even open Hypershade. ZBrush produces displacement maps that are 16-bit grayscale images.

You need to make the modifications to your maps prior to use in a Maya shading network. You must also modify attributes of both the file node and add MayaMan attributes to your object.

Here's a complete list of the things you should do:

a. Subtract 0.5 from the image using your favorite paint or compositing package to do so, ensuring you keep the map a 16-bit file.

b. Convert the grayscale image to RGB.

c. Setup a basic Lambert, Blinn or Phong shader and attach the displacement map to the shading group's displacement parameter. See image 2.

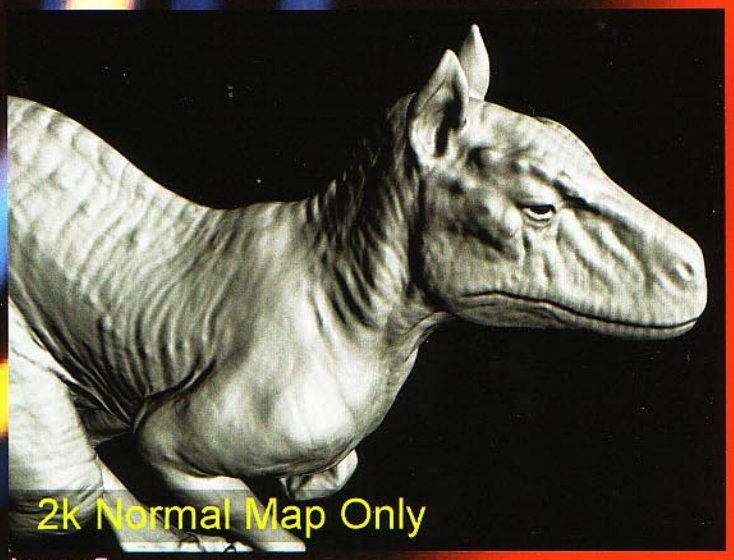



Image 7

Image 8

Image 9

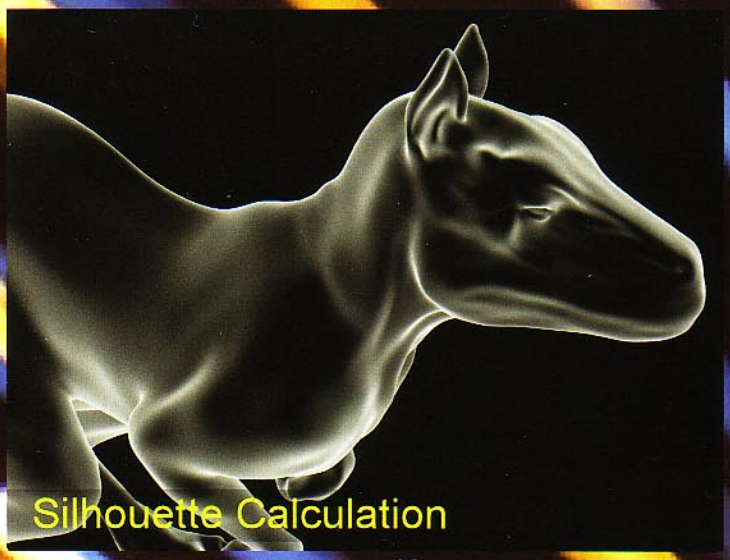



Image 10

Image 11

d. Under the Color Balance section of your texture file node, enable the "Alpha is Luminance" checkbox.

e. More than likely, you will need to adjust the Alpha Gain parameter to a lower value such as 0.1. This adjusts the displacement height of your map. Experiment to obtain the correct value.

f. Add a MayaMan Model attribute to your object (from the menu, select MayaMan -> AddModelAttributes). A new section tab will be available when you select your object and open up an Attribute Editor. Enable both the "Render as Subdiv" and "Subdiv texture coords use varying" checkboxes. See image 3.

You are now in a position to render your image. All going well, your low polygon model should look just like your highly detailed version. As an example, here's a creature's foot I roughed out in Maya, imported and detailed in ZBrush, exported the displacement map, re-imported the low-resolution version of my object back into Maya and followed the above process.

Image 4 is my low polygon version as viewed within Maya, which is easy to skin and animate:

Finally, image 5 is the final rendered version rendered with Sitex Graphics AIR renderer and MayaMan:

*2.2. Maya and mental ray*

A Renderman-compliant renderer, such as Pixar's PRMan or Sitex Graphics' AIR, seems to be the renderer of choice for displacement mapping, especially when you consider the speed and flexibility these provide.

But they come at a cost that only larger production studios can afford, not to mention the added skills required to program custom Renderman shaders.

This leaves smaller studios and freelance artists with fewer options, but these too provide great results. We now find mental ray fully integrated into Maya, whilst Softimage|XSI users have enjoyed the strength of this renderer since its first release.

As with the first approach of using MayaMan with Renderman, mental ray can use the default Maya shaders, but there are a few tricks you must follow to get the same fantastic results Renderman provides:

a. Convert your 16-bit TIFF displacement map to Maya's IFF format. Use the command line utility "imgcvt.exe" to do so. This will ensure mental ray picks up the fine detail in your map.

b. Once you have imported your object into Maya, convert it from a polygon object to a true subdivision surface.

c. Attach a shader with displacement map to your object. In the file node, change the Filter Type to Gaussian. This helps to counter texture borders causing artifacts on your rendered object.

d. You then need to increase the tessellation of your SubD surface. In the Attribute Editor, change the shape node tessellation settings of your object to uniform, and increase Depth to 3 and

Image 12

Sample Count to 5. Optimal settings will depend on the level of detail in your map and will require experimentation. Note: The higher these settings, the greater the rendering time.

Image 6 shows the creature's foot rendered in mental ray.

### *3. Combining Displacement and Normal Maps*

So far we've only looked at using the displacement maps alone. An alternative method is to combine the displacement map with a normal map.

Normal maps are similar to bump maps; however, they provide a substantial amount of information when compared to their poor cousins. Instead of simply moving the normal in or out at a given surface point to simulate bumps and surface detail, normal maps can also 'deflect or angle' the normal within a hemispherical region.

This provides the ability to add further detail to a character, whilst using a lower resolution displacement map. The end result of combining displacement and normal maps appear to be quicker renders.

The tricky part is getting the maps to combine together in a shading network.

The Renderman Shading Language provides a high degree of flexibility when performing normal and displacement calculations. To work properly, normal mapping requires the pre-displaced surface normals and tangents, which Renderman provides.

This is not so in other rendering systems, such as Maya's default renderer, which typically adds, rather than replaces, the normals generated from the normal map to the displacement map-generated normals (yes, that's a real mouthful and something to get your head around).

A recently developed work-around to this problem was developed by Phil Dench of Headus in Australia, who was able to design a Maya shading network that uses the inherent detail in a normal map, and combine it with a displacement map. The displacement map value is made a factor the inverse of the facing ratio of a given sample point.

What this means is that at an object's outline or silhouette, the displacement map is providing full displacement according to the strength of the map, whereas other areas of the object are less displaced.

As an example of this approach, image 7 shows a render using a 2K normal map of "Killeroo." Notice the perfectly smooth silhouette.

Image 8 is a render using a displacement map and image 9 a render using a combined normal and displacement shader. Notice in the latter that the normals seem over-exaggerated.

Image 10 is the output from the facing ratio calculation. This shows where the varying levels of displacement occur on the model.

Image 11 is a render using the newly developed shader, which combines the normal map with a factored displacement map using the facing ratio. This renders quickly and provides a great level of detail where required including a proper silhouette.

Finally, image 12 shows the Maya shading network to achieve this effect. Thankfully, it is downloadable from the Headus website at www.headus.com.au.

The development of these shaders, continued experimentation by the community and release of new tools, such as those found in the new version of ZBrush will help to ensure this technique will continue to break new ground, and find its way into feature film production.

*With great thanks to the following creative geniuses: Animal Logic, Sitex Graphics, Pixologic and especially to Phil Dench from Headus.*

# BUILDING A SIMPLE COLOR SUPPRESSION MACRO IN SHAKE

BY MATT PLEC

THIS TIME AROUND I'M GOING TO BUILD A RELATIVELY SIMPLE MACRO THAT ILLUSTRATES TWO REALLY IMPORTANT CONCEPTS IN SHAKE: THINKING IN PARALLEL, WHICH WILL INFLUENCE THE OVERALL DESIGN OF THE MACRO, AND VARIABLE DEPENDENCIES IN MACROS, WHICH WILL COME INTO PLAY WHEN WE GET INTO THE IMPLEMENTATION DETAILS.

Many users make the mistake of thinking about what they're doing in Shake on the level of individual pixels and end up writing expressions on ColorX/LayerX/WarpX nodes in order to build up effects that vary based on input pixel values. Unfortunately, by their very nature as general-purpose tools these nodes are nowhere near as efficient as their special purpose cousins like Add or Mult. Thinking of operating in parallel, as if on the image as a whole, every pixel at once, rather than on individual pixels, can lead you to build much more efficient processing trees. Unfortunately, many of us get in the habit of thinking serially and trying to specify pixel-by-pixel what we want Shake to do. For example, to suppress the level of green in a pixel to the level of blue in it, we might use a ColorX node with the gExpr parameter set to (g > b) ? b : g which means, "if green is greater than blue, use blue, otherwise use green." This would indeed give us the result we're after: green values unchanged where they don't exceed blue, and clipped at blue otherwise. We could also implement it more simply as just min(g, b)

The output is again the blue values where green exceeds blue and green values where green is less than blue.

Unfortunately both of these require use of ColorX. The whole thing would work a bit faster if we could find a way to do it with built-in nodes optimized for their particular tasks rather than ColorX. In this case, we could reorder the channels so that the two inputs to the Min node were the green and blue channels separately and the output was copied back into the green channel of the original image, giving us exactly the same result as the expression on ColorX did but gaining the efficiency of the optimized Min node. What about all this extra work to reorder the channels around the Min? On the surface this seems counterproductive, using four nodes rather than one, but a Reorder or Copy costs next

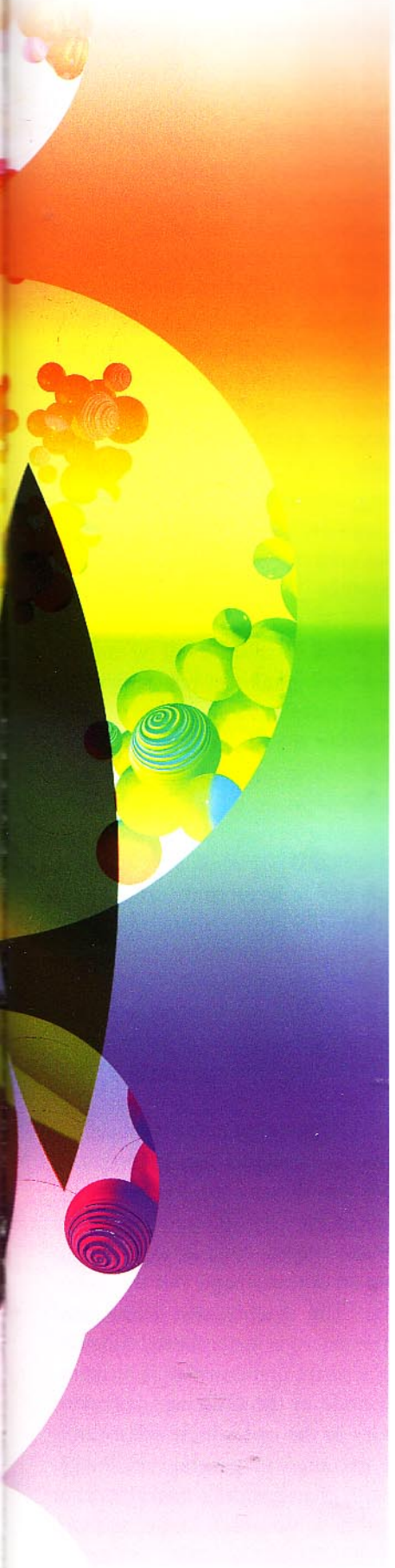
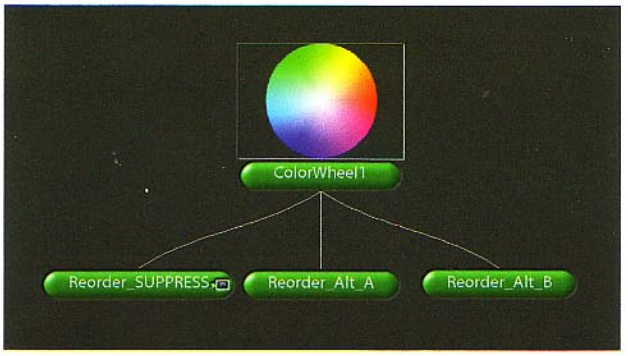


Image 001




to nothing – no image data actually has to get moved around, just pointers to channels swapped – so in the long run, and particularly for more complex operations, the result is much more efficient. For a one-off, this may not be worth the effort, but for something you might use over and over or turn into a macro, the efficiencies can add up to big improvements in overall render times.

Let's take a look at this in action by implementing a simple color suppression macro. The SimpleSuppress node will reduce the level of a given channel to the max of the other two, a basic spill-suppression technique. For example, suppress green to the max of red and blue. So this will require the Min() of the suppress channel with the Max() of the other two channels. To rough in the implementation, let's assume we're suppressing green and generalize later. So we need Min(green, Max(red, blue))

Which is easy enough to set up. First, the individual channels are separated into luminance images.

[image 001 – Input image's channels separated into luminance images]

Where Reorder_SUPPRESS has channels set to ggg, Reorder_Alt_A to rrr, and Reorder_Alt_B to bbb.

Now we simply take the Max of the alternate channels and the minimum of this with the suppress channel. Both of these nodes are operating on all the channels of the input images, but being luminance (BW) images there is really only one channel so it doesn't matter which channel we take from the output – they are all the suppressed channel. A Copy node set to "g" can be used to put the suppressed green channel back into the original image. Alternatively, we can add a control for the amount of suppression by using a Mix to blend the original image with the suppressed result, setting the Mix node's channels parameter to just "g" so that the red and blue channels don't get mixed up (so to speak) with the suppressed green channel.

[image 002 – Implementing the suppression and mixing it back with the original]

Now let's convert this into a macro so we can get into the guts of it in a text editor to generalize it to suppress any channel to the max of the other two. Select everything from the Reorder nodes down and press shift+m to bring up the macro maker. Expose the Reorder_SUPPRESS nodes' input image and the Mix node's percent parameter, choose the Mix node's output for the macro's output, and save the macro as "SimpleSuppress."

Open the macro in a plain text editor. If you're not sure where to find the macro, it's time for a trip to the "Customizing" section of the manual. In Shake 3, the quickest way to get there is the "Help -> Customizing Shake" menu item. Most likely, the macro is in your home directory's nreal/include/startup folder. It should contain something like this:

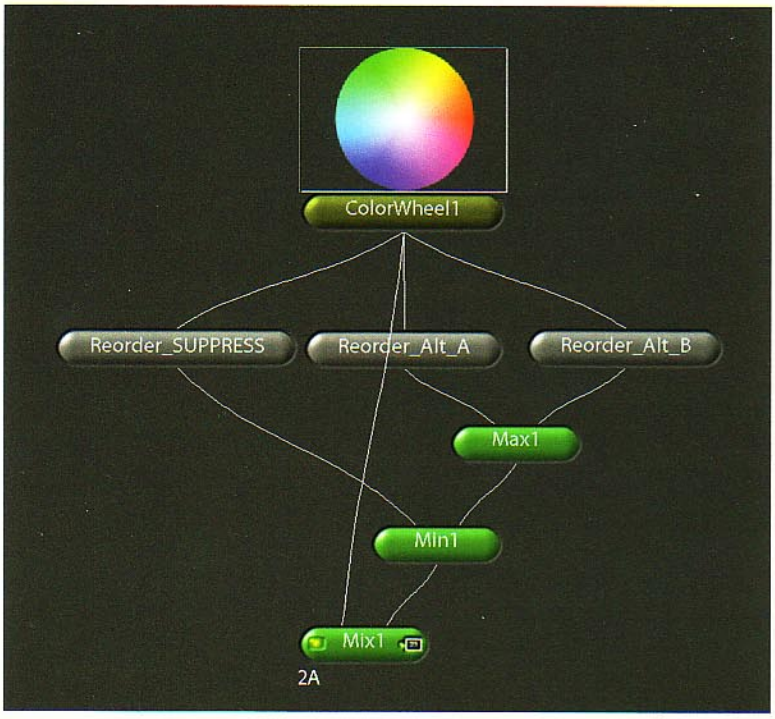



Image 002

```
image SimpleSuppress(
image In=0,
float percent=100
)
{
   Reorder_SUPPRESS = Reorder(In,
"ggg");
   Reorder_Alt_A = Reorder(0,"rrr");
   Reorder_Alt_B = Reorder(0,"bbb");
   Max1 = Max(Reorder_Alt_A,
Reorder_Alt_B, 1, 100);
   Min1 = Min(Reorder_SUPPRESS,
Max1, 1, 100);
   Mix1 = Mix(Foreground, Min1, 1,
percent,"g");

   return Mix1;
}
```

First off, we'll need to do a little cleaning up and reattach the other Reorder nodes' inputs. Next, we'll add a control for the channel to suppress; this will be a string so that we can use it directly in the nodes' channels parameters. This is the channel that will be mixed back in at the end, so we'll change the Mix() to use it now.

```
image SimpleSuppress(
  image In = 0,
  string suppress = "g",
  float percent = 100
)
{
   Reorder_SUPPRESS = Reorder(In,
"ggg");
   Reorder_Alt_A = Reorder(In,"rrr");
   Reorder_Alt_B = Reorder(In,
"bbb");

   Max1 = Max(Reorder_Alt_A,
Reorder_Alt_B, 1, 100);
   Min1 = Min(Reorder_SUPPRESS,
Max1, 1, 100);

   Mix1 = Mix(In, Min1, 1, percent,
suppress);

   return Mix1;
}
```

All that's left now is to get the Reorder nodes to take the right channels based on the suppress parameter's setting. We could set up a handful of Select nodes to feed the various luminance images into the right places, but with all the power of the macro language we should be able to find a programmatic way to just set the existing Reorder nodes to put the proper channels into luminance images. Two things will help us here. First, we can use conditional statements like if() and else within the macro. Second is the fact that we can concatenate text strings with a simple +. For example, in Shake's scripting language "about" + " " + "face" equals "about face."

So let's determine the alternate channels like this:

```
curve string alt_a;
curve string alt_b;

if (suppress == "r") {
  alt_a = "g";
  alt_b = "b";
}
else if (suppress == "g") {
  alt_a = "r";
  alt_b = "b";
}
else {
  alt_a = "r";
  alt_b = "g";
}
```

Note the "curve" statement preceding the string declarations. This is absolutely essential for this macro to behave as expected in the GUI. It's often assumed that Shake macro code is executed to generate an output image, but this is not the case. The macro code is executed when the macro is instantiated, either by loading a script that contains it or creating the node in the GUI, and it creates the nodes in the tree which will eventually be executed to generate an image. Now, with that in mind, consider what happens if a variable's value changes during the life of the node, for example by the user changing the value of

"suppress." The macro tree has already been built and the variables resolved. Nothing would change! The "curve" statement is the solution to this. When it's encountered, Shake knows that this variable could change after the tree is built and anything that uses it will need to get rebuilt if it does. In many cases you can get away without using "curve" by setting the variable to an expression that contains an input parameter when it's created. Since these are known to be changeable over the life of the node, the variable is then marked as something that needs to be rebuilt if the parameter changes. Likewise, anything that uses it will get rebuilt if it changes. However, the declaration of a variable is your first and last chance to make sure Shake knows how to treat it, so if it's not marked "curve" and not set to an expression containing another "curve"-type variable (like an input parameter) then it will get resolved when it's used. Thus, we could use the conditional statement form introduced at the beginning of this article to rewrite this as:

```
 string alt_a = (suppress == "r") ?
"g" : ( (suppress == "g") ? "r" : "r" );
 string alt_b = (suppress == "r") ?
"b" : ( (suppress == "g") ? "b" : "g");
```

And with the + string concatenation, we can set our Reorders to something like "alt_a + alt_a + alt_a" to get "rrr", or "ggg," for example. Our SimpleSuppress macro now looks like this:

```
image SimpleSuppress(
 image In = 0,
 string suppress = "g",
 float percent = 100
)
{
   string alt_a = (suppress == "r") ?
"g" : ( (suppress == "g") ? "r" : "r" );
   string alt_b = (suppress == "r") ?
"b" : ( (suppress == "g") ? "b" : "g" );
   Reorder_SUPPRESS = Reorder(In,
suppress + suppress + suppress);
   Reorder_Alt_A = Reorder(In, alt_a
+ alt_a + alt_a);
   Reorder_Alt_B = Reorder(In, alt_b
+ alt_b + alt_b);

   Max1 = Max(Reorder_Alt_A,
Reorder_Alt_B, 1, 100);
   Min1 = Min(Reorder_SUPPRESS,
Max1, 1, 100);

   Mix1 = Mix(In, Min1, 1, percent,
suppress);

   return Mix1;
}
```

To finish it off, we can add a simple radio button widget to the UI.h file to make sure the suppress channel is set to something valid. Add this to the SimpleSuppressUI.h file:

```
nuxDefRadioBtnCtrl("SimpleSuppress.
suppress", 0, 1, 1, "r|red", "g|green",
"b|blue");
```

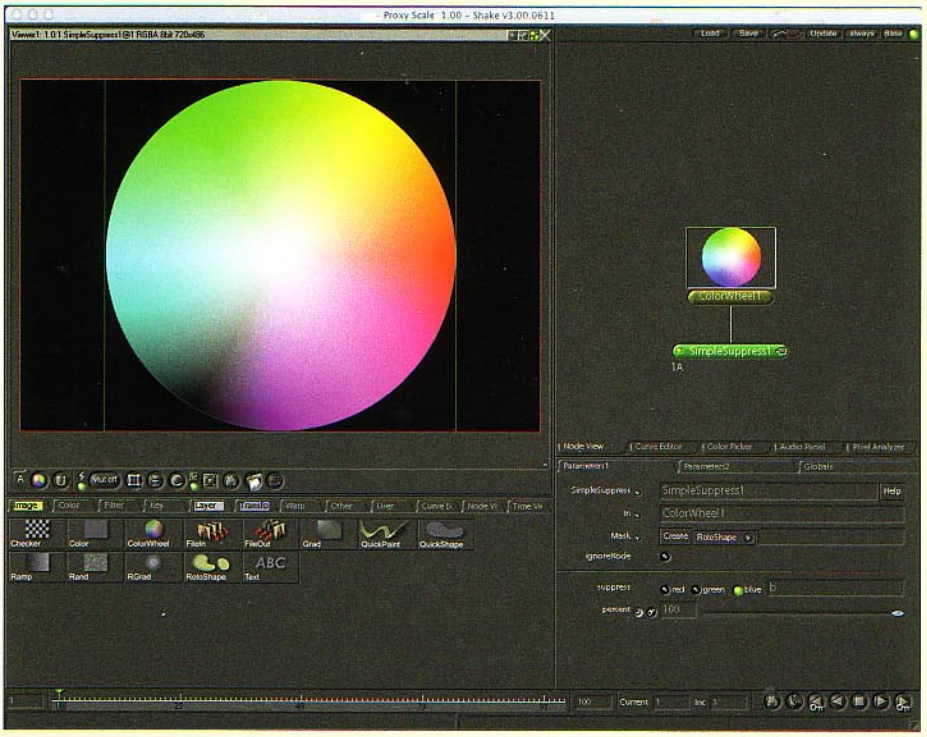



Image 003

And there you have it. Start Shake up and feed a ColorWheel into your new SimpleSuppress node to give it a test run and verify that everything's wired up correctly.

[image 003 – The SimpleSuppress macro in action]

While this is a simple case that wouldn't have suffered too much from implementation with ColorX, more complex cases can show a noticeable improvement in rendering time when implemented with special-purpose nodes rather than ColorX, even more so when used a number of times in a large film-resolution project.