

Color Management

written by **Jeff Mottle**

technical review by **Pierre-Felix Breton**



Introduction

What Is Color Management?

Simply put, color management is the controlled process of converting color between different devices with the end goal of obtaining repeatable and as closely matched color reproductions as possible on each of the various devices. In other words, we want to set up a process where all of our devices speak the same language and interpret color the same way. Practical examples include printing an inkjet output that closely matches your display, or scanning a physical material swatch that looks the same on screen as it does in real life. Although this is certainly not an exhaustive list of applications for color management, this should give you a general idea of the subject matter of this chapter. I suspect many of you reading this chapter have spent a great deal more time than you might like to admit trying to get your printouts looking as you expected or passing files along to clients with the hopes they see the colors you saw on your own display.

Color management is a subject that is fraught with much confusion and misconceptions and, in many cases, is not used at all. In fact, it is a subject that is widely misunderstood within many graphics fields including printing, photography, and graphic arts. Less than half of the architectural visualization industry currently uses a color managed workflow on a daily basis (based on various industry surveys conducted by CGArchitect.com). Considering the importance of achieving accurate and repeatable color in our work, it stands to reason that more importance should be placed on adopting a color managed workflow. However, most do not use color management. The reasons for this are numerous, but I think the number one problem lies in the complexity of the subject and the vague documentation that accompanies many of the applications and devices we all use in production. Color management deals with how we as humans see color and how that understanding can be turned into mathematical equations that can be understood by computers and devices. This sounds a bit daunting, but the good news is it doesn't require a degree in advanced applied mathematics to

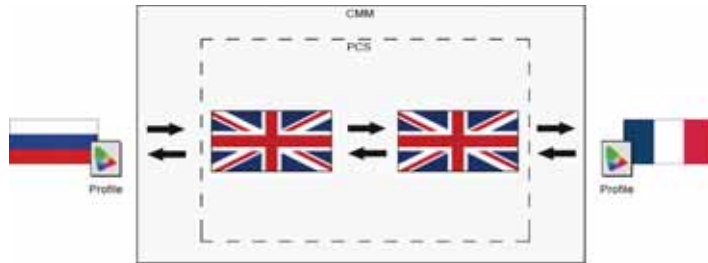


Figure 1-4. The color management workflow

Color Management Theory

How We See Color

This next section of the chapter is going to seem like a review of high school physics, but bear with me as I do a quick refresher on some of the core concepts involving how light works and how we perceive color. The core of many aspects of color management and color science are designed around how we perceive light and color; hence, it is important to have at least a cursory knowledge of the basic principles.

Three factors affect how we perceive color--the **light source**, the **object**, and the **observer**.

The Light Source

Light is most commonly referred to as an electromagnetic wave that is described by its length, usually in nanometers (nm). Light is the visible part of the electromagnetic spectrum (Figure 1-5). The full spectrum of waves includes radio and microwaves at the larger end of the spectrum to x-rays and gamma rays at the smallest end. Visible light falls somewhere in the middle where the wave's measure is between 380 and 750nm. These are the only waves we are interested in as this is the generally accepted range of waves we can perceive. When waves of light are in the 400-500nm range, we perceive the color as being blue. When they are in the 500-600nm range, we perceive them as being green and, finally, when they are in the 600-700nm range, we perceive them as being red. The entire spectrum of visible light is made up of light consisting of red, orange, yellow, green, blue, indigo, and violet. When our eyes receive amounts of colors from the entire spectrum, we see the color white, and, when there are no waves of light, we see black.

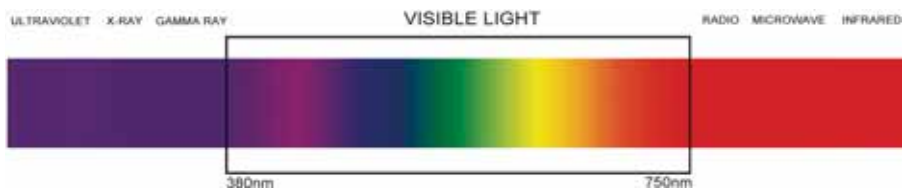


Figure 1-5. Electromagnetic spectrum

Light sources come in many forms and can include the sun and artificial sources like fluorescent lighting, incandescent light bulbs, tungsten bulbs, and candles as well as others. When the spectral curve of a light source is viewed, you will see that it is comprised of a mixture of wavelengths (Figure 1-6). Most light is a mixture of many wavelengths with the exception of pure light as emitted from lasers. The perceived color of any particular light source depends on the predominant

Composition

written by Ernest Burden
technical review by Michele Bousquet



ARCHITECTURAL RENDERINGS OFTEN FALL SOMEWHERE in between informational reporting and subjective instruments for sales. At its simplest, a rendering can be a visualizing document to help people understand an architectural proposal. It can be quite accurate in scale, material, and lighting. It can be informative and clarifying, even exciting and persuasive. It can be beautiful. It can also be muddled, confusing, and uninspiring. The difference comes from the artist having an understanding of the project and the best ways to present it. The artist must recognize the various aspects of the work as visual tools he or she employs to describe, simplify, and, especially, clarify the subject. Composition is the art of arranging the scene or image to give the best visual presentation possible for the needed purpose.

A rendering is not the real building, of course. It is a translation, an interpretation. A rendering transforms a work of architecture—a three-dimensional object—into a flat illustration, a medium that has always been about telling a story. The best works catch the viewer’s attention and draw them in. But what are the tools and techniques architectural artists can employ to do that? That is the essence of good composition, and what this chapter is about.

Point-of-View

Visualization is not newspaper reporting; do not be afraid to be biased. Have a point of view. There are many ways to present a project. Think about the things you need to say, to whom you need to connect, and what choices will present the project best. What does your client need you to portray? Renderings and animations that try to be “all things to all people” will likely be dull. In partnership with your client, it is your responsibility to design a visual communication that captures the attention of your target audience and opens it to the project in a positive way. Taken further, the artist should make the project exciting and welcoming. He should strive to tell a story featuring the subject.



Figure 2-21. An angled view of the swimming pool leaves ample space on the right for wandering.

Image by Allen + Philp Architects/Interiors, RVT Custom Content at Revit Market, TurboSquid.com. Scene created in Revit.

In graphic terms, there are commonly held ideas about placement of focal points within a picture. The most common one is the rule of thirds. Simply put, draw a grid over your image dividing it into thirds, horizontally and vertically. The grid creates four intersecting points that are good locations for areas of interest, higher contrast, or important elements. The lines are used to place the boundaries between large shapes. An example of that would be the horizon. The rule of thirds says place it either up from the center or below it at about the one-third level. The same holds true for vertical things—around the one-third line, left or right. This produces a pictorial form that expresses another major/minor relationship. The proportions created mimic the Golden Section, or Golden Rectangle, which is an ancient yet still widely used device for proportioning works of art and architecture. There is no reason to always follow these proportions, but their use can help you keep your images interesting and dynamic.

Training yourself to think of your visual art in terms of its two-dimensional, framed forms takes time but is worth the effort. Producing pieces is a transformative process from idea to the tangible product. Having a feeling for the end state will guide you through that process. Working in a virtual environment separates us from the world we are trying to depict. Practice drawing to stay connected to the physical result. Go outside and observe a scene, making notes to yourself about it, what it is like and how you feel when you are there. Then take a photo of that scene. Compare the photo with your memories and impressions of the subject and look for what is different between them. Does the photo capture the feeling you remember? Identify what is missing from the photo that you could add to help it match your memory; also look for aspects of the photo that enhance your understanding through detail or a different sense of scope.

You have probably seen the clichéd depiction of a movie director holding up his hands to frame a scene. He makes an “L” with each thumb and first finger, turning his hands to form a rectangle with the posed fingers. He uses that opening to frame his view. Try it yourself; it works. That simple act of framing allows you to see what is before you differently, to see it as a 2D projection within a frame. Understanding how imagery is perceived depending on how it is presented is a key to producing effective imagery.

Advanced mental ray Shaders

written by **Joep van der Steen**

technical review by **Jeff Patton**



Copyright www.belly.be

3DS MAX HAS A HISTORY of keeping old features in future releases so that features used in older scenes continue to work properly in the newer releases of the program. This is good but also a bit confusing, especially when the old functionality is still there but has actually been overtaken by new and improved features. For example, the newest optimizing modifier is **ProOptimizer**, but you can still find the **Optimize** modifier in the modifier list as well.

A new user might struggle to figure out which one to use since some do basically the same things. Most of the time the newest version of a feature is preferred by veteran users as it uses the latest algorithms and technology and does an overall better job than the older ones. In other words, it's helpful to know some of the feature history of 3ds Max in order to know what NOT to use anymore because it has become rather redundant.

Mental ray has been available as a native render engine since 3ds Max 6, but at that time it was somewhat confusing to users because the way the materials and mapping worked was completely different than what users were used to. Now we are at 3ds Max 2010 and, of course, mental ray has gone through some enormous changes and improvements.

In this chapter, I begin by pointing out some of the older material and giving you advice on what not to use anymore for both materials and shaders. This will help you understand what you should be focusing on. After this short introduction, I will provide a number of exercises that are based on real world support questions I received over the years from users of all skill levels.

Keep in mind that mental ray is still making big steps forward with every release as rendering technology is very much a dynamic technology. Please don't assume you know it all from previous releases; there might be fundamental differences in how and what to use as mental ray moves to the next release in 3ds Max. Things such as modeling to real life scale, physically correct lights, and materials are still important, but elements such as Exposure Control and Gamma have gained in importance as the rendering technology used by mental ray has evolved. Working with mental ray is an ongoing learning process, and that is what makes it challenging and fun.

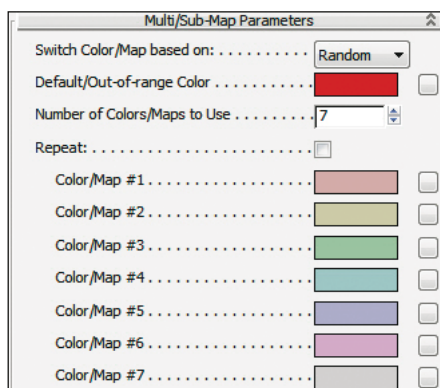
3ds Max 2010 ships with two new maps/shaders that can be quite useful, **Multi/Sub-Map** and **Object Color**. Neither of these shaders is too complicated. The Multi/Sub-Map shader is especially useful. The following exercise demonstrates how the Multi/Sub-Map can be used in an efficient way when using Bitmaps.

Using the Multi/Sub-Map and RGB Multiply shaders

1. Open the file **ch03-01.max**. This is a simple interior scene consisting of a basic living room with an ordinary tiled floor. We want to apply some color to the floor but give it some color variation as well. This is exactly what this new shader allows you to do: assign one material to multiple objects and introduce color variation within the objects that receive this material.



2. Open the **Material Editor** and within the **Color** channel of the **Floor** material, apply the **Multi/Sub-Map**.
3. Set the **Switch Color/Map based on** setting to **Random**, as we want to apply random colors to our geometry.



Advanced Unwrapping

written by Lukas Dubeda

technical review by Taylor Leach



UVW MAPPING IS ONE AREA of 3D content creation that artists tend to enjoy the least. This is understandable because UV mapping can get extremely time-consuming. It is mostly unavoidable, rather technical, and stresses the poor individual who has to perform the unwrapping task. In this chapter, I will try to cover techniques that should save you time and effort when unwrapping your meshes, whether inorganic surfaces or organic, fluid forms. Further on, I will discuss more advanced concepts of UVW mapping and try to explain what those coordinates really are, how they are seen by the 3D software, and how you should approach them for best results. We will also take a look at some areas that are indirectly related to UV mapping, such as **Normal** mapping and techniques connected with them. At the end of this chapter, I will discuss third-party tools and plugins that were designed and developed to save time, effort, and generally improve your daily workflow as they relate to UV mapping.

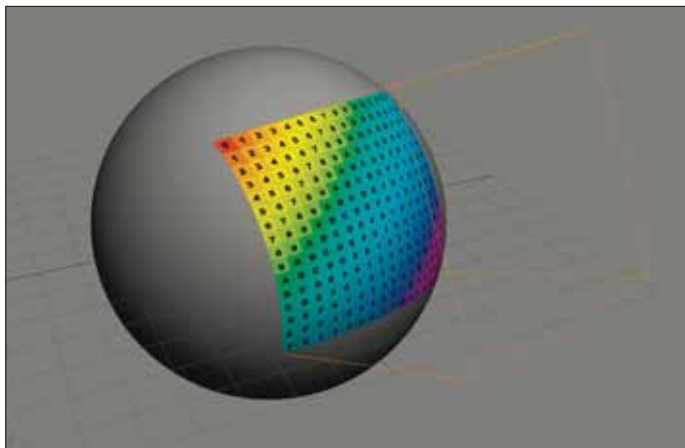
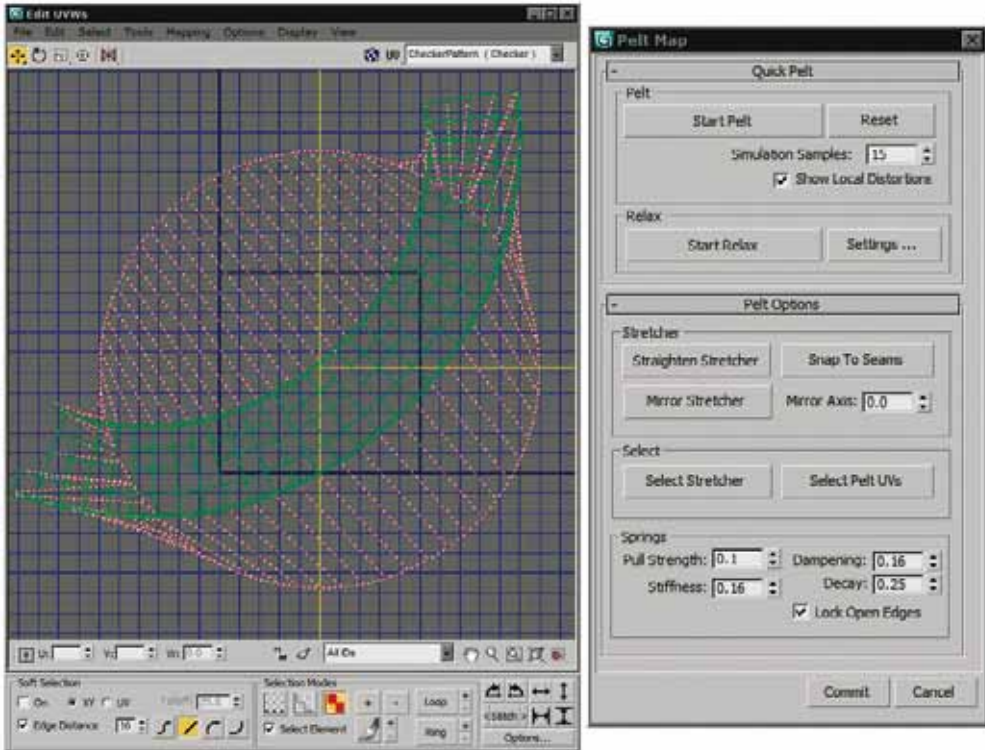
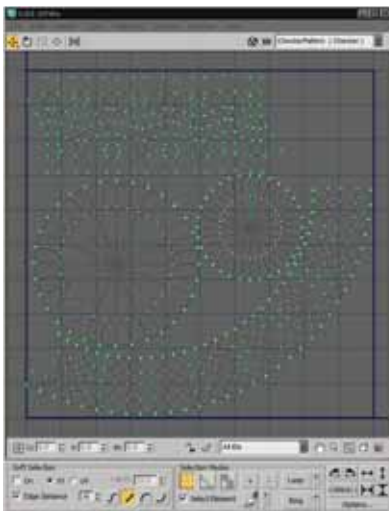


Figure 4-1. A UVW mapping projection visual demonstration



- The most difficult part is done! That is the beauty of Pelt Mapping. All you have to do now is organize the UV chunk so that it fits within the 0-1 UV space and, of course, finish Pelt Mapping the rest of the model: which is essentially the same procedure as we have just done.
- The following illustration shows a finished UV mapping for the entire model, all completed nicely and cleanly in a few minutes.



Texture Painting

written by Leigh van der Byl

technical review by Laszlo Molnar



FOR MANY, TEXTURE PAINTING IS a somewhat confusing and complex process best dealt with as simply and quickly as possible. Sadly, sticking a couple of random texture swatches that you downloaded off the internet doesn't generally suffice for work intended as high quality professional imagery. Using Photoshop as the primary technical tool, this chapter covers various fundamentals to equip artists with the necessary theory and practical tips to work efficiently and confidently with their texturing requirements.

Texturing is a combination of both applied theory, as well as artistic skill. If your painting skills are a tad lacking, don't worry as they're likely to develop quickly with practice. While Photoshop can seem quite an intimidating and huge beast to tame, this chapter shares a number of tips for improving your Photoshop workflow and technique.

The texture process works hand-in-hand with the shading and lighting processes. In a production environment, it's important for texture artists to work closely with "look-development" artists who set up the shaders as well as the lighters to devise an appropriate texturing strategy based on the requirements of the scene.

Ultimately, texturing can be an extremely enjoyable and rewarding creative process. Once you've mastered the fundamental theory that drives it, you're free to devise your own painting and editing techniques and tricks to build up your textures, which is where the fun really starts. Texturing is all about bringing color and life to otherwise plain, gray surfaces, giving you a chance to truly indulge your artistic side and bring out the most in your CG models.

Getting Started – Understanding Real World Surfaces

As a texture artist, it is vital to observe your environment in a way that enables you to understand exactly what you need to create within the computer generated environment in which you work. Merely observing the world on a superficial level is not sufficient.



Figure 5-8. A basic reflection map, on the right, creates the illusion of a wet floor in this render

Bump and Displacement

Probably the most commonly used material channel along with the color map is the bump map. Bump mapping is a method whereby you create the illusion of irregularities, topical abrasions, or damage, such as scratches, dents, or any kind of textural grain to a surface, without adding to or altering the geometry of the model in any way.

Because no surface in this world is perfectly smooth, no matter how smooth it may appear to the naked eye, all surfaces you create require some kind of bump mapping in order to create a sense of actual tangible texture or roughness to the object. Clever use of bump mapping conveys an idea of what the surface would feel like to the touch.

It is important to reiterate, however, that bump mapping does not affect the actual geometry of the object itself as it is solely an illusion. This illusion is usually betrayed by the edges of an object when rendered, appearing smooth even if a very rough bump map is applied to it as shown in Figure 5-9.

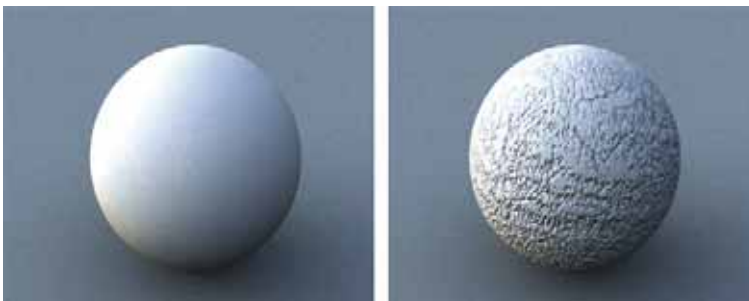


Figure 5-9. The illusion of a bump map is betrayed by the lack of detail along the edges of the object

Physically Based Materials

written by Pierre-Felix Breton

technical review by Zap Andersen



REPRESENTING MATERIALS AND COLORS ACCURATELY in computer graphics has always been a challenge for anyone who desires to represent architectural finishes accurately. This chapter does not instruct you about what button to press in V-Ray or mental ray to create nicely looking wood or concrete, but rather focuses on strategies of how to estimate characteristics of material samples with affordable hardware, such as digital cameras and light metering tools. You will learn how to correlate them to user interface parameters of commonly used, physically based renderers.

The Typical Problem

What is the correct RGB color to use for a given material sample?

As a 3D artist, you most likely face situations where clients ask you to illustrate a project by using a specific brick, paint color, or carpet sample (Figure 6-1). Although you can scan or photograph the samples you have in hand, it is possible that you are unclear about how closely they will end up looking in the finished rendering. To achieve this with success, you should think about color management and color measurement.



Figure 6-4. A dark floor vs. a pale floor under identical lighting conditions - nothing else has changed; the amount of reflected light (reflectance) has a direct relation with the material color (and vice versa).

You have now seen how a material color correlates to its reflectance, but there is more. Before we go further, it is important to understand basic physical principles as applied to materials. Understanding these rules will be essential when it comes time to measure optical characteristics of materials.

Diffuse Reflectance vs. Specular Reflectance

The **Energy Conservation** rule states that the total sum of outgoing rays never exceeds the sum of incoming rays.

In other words, a material cannot reflect 80% of diffuse light while reflecting 80% of specular light. It would result in creating energy that is physically incorrect. In 3ds Max, this can translate roughly as the following: diffuse lighting + specular highlights + reflections ≤ 1.0 (Figure 6-5).



Figure 6-5. The energy conservation rule states that the sum of outgoing rays cannot exceed the sum of incoming rays.

Exposure Lighting Analysis Tools

written by **Tod Stephens**

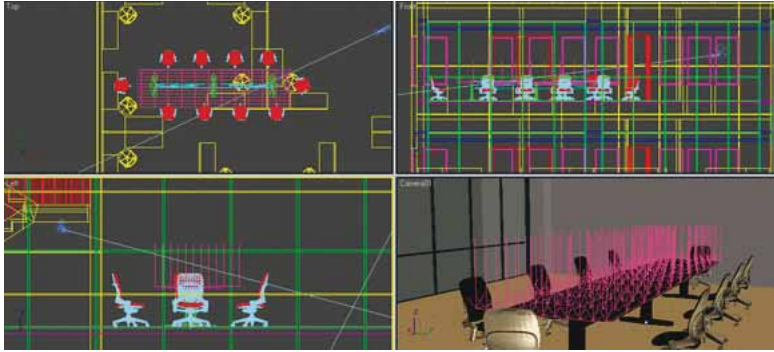
technical review by **Steve Sanderson**



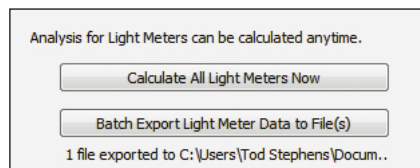
UNIQUE TO 3DS MAX DESIGN, **Exposure Lighting Analysis** tools can be used for simulating and analyzing physically based Sun, Sky, and artificial lighting in a 3D scene. Ideal for Architectural projects, these tools can help evaluate light intensity in your designs. The Lighting Analysis tools can help facilitate the evaluation of indoor lighting quality for **LEED Certification**. Besides using the Exposure Lighting Analysis tool itself, designers can use the **Light Meter** helper object to verify lighting designs and to quickly view alternative lighting schemes.

This chapter will discuss the requirements of the 3ds Max Design Exposure Lighting Analysis tools. For accurate lighting simulations in 3ds Max Design, it is essential that the model geometry be clean, airtight, and have a ground plane or site topography. The mental ray rendering engine with **Final Gather** must be used; specific settings will be discussed later in this chapter. Exposure control settings will be discussed in detail. When using a **Daylight System** for lighting analysis, specific settings are required to achieve physically accurate light levels, and a proper **Sky Model** and **Weather Data** file must be used to drive the system. Lighting Analysis requires the use of physically accurate materials such as 3ds Max **ProMaterials** or **Arch & Design** materials. Additionally, all artificial lighting must be photometric. Once physically accurate materials, photometric lighting, and mental ray settings are in place, a **Light Meter Helper** object can be placed at any location in your model where the light levels are required to be measured. To display the light intensity values calculated by the Lighting Analysis tool in the final rendered image of a scene, an **Image Overlay Render Effect** is required.

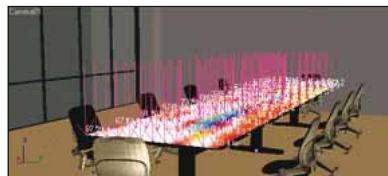
The **Lighting Analysis Assistant** is a tool in 3ds Max Design that gives a simple, step-by-step approach to setting up all the requirements for lighting analysis. This chapter will discuss each step of the process and give suggestions for settings and procedures that will simplify the use of the Assistant. A model and all necessary files are included on the book's website, allowing you to work through the entire process.



18. To display the lighting units in foot-candles, make sure the **Lighting Units** setting in the file is set to **American**. Go to **Customize > Units Setup** to verify the setting.
19. On the **Lighting Analysis Assistant** dialog, click on the **Calculate all Light Meters Now** button (shown in the next illustration), and the color-coded lighting level values (in fc) will display in the active viewport. Click on the **Batch Export Light Meter Data to File** button to save the data to an . CSV file. The default location for this file is your 3ds Max Design project folder. You can also select the **Light Meter** and click on the **Export to CSV File** button under the **Modify Panel** to save the data.



20. Go back to the **General** tab of the **Lighting Analysis Assistant**, and change the **Max** value of the **Analysis Value Color Coding** to **300 fc**. This will result in a suitable color code distribution on the **Light Meter** in the viewports, as shown in the next illustration.



21. This would be a good time to perform a test render. You have not set up the **Lighting Analysis Image Overlay** yet, so this will be a render of the scene that does not show any lighting values. You can open the 3ds Max Scene file **ch07-04.max** to work with the file with all the settings up to this point. For this test render, set **Final Gather Diffuse Bounces** to **2**, as shown in the first of the next series of illustrations, and, in the **mr Photographic Exposure Control** settings, use an **Exposure Value** of **12.5**, as shown in the right image of the following illustration. Render time for the 640x480 pixel rendering, shown in the next illustration using a Quad-core processor, was 17 minutes.

Render to Texture

written by Louis Marcoux

technical review by Mike McCarthy



IF YOU HAVE WORKED IN 3ds Max with only rendering in mind, you have probably spent many hours waiting for your renderings to happen. You have tried many tricks to save on rendering time but rarely have you been able to go under a few minutes per rendering, especially when all direct and indirect illumination settings are turned on. You have obtained amazing results at the cost of render time. That is the reality of producing pretty pictures.

Then one day you got some free time (while rendering maybe!), and you sat down in front of your favorite game console to have some fun. At this moment, you noticed how many frames per second were rendered on screen as you played. To give you full interactivity with the game, a game engine needs to produce more than 30 renderings per second. All these rendered frames per second have beautiful lighting and amazing ambiance with soft shading and complex light casting. Interactivity and pretty pictures are the key to give the gamers a great experience as they are moving inside the world of a game.

Game developers have been using tricks to produce high quality images while reducing the amount of rendering time to a fraction of a second per frame. That is true for any industry that needed to develop real time rendering content such as military simulations or virtual reality.

One of the main techniques used to reduce rendering calculation time is to remove all light calculations from a scene. In real time 3D, it is done by pre-processing all lighting solutions prior to loading the 3D content in the interactive game engine. All these light calculations, complex or not, are put directly into the texture maps of the models. This way the main texture of a model includes not only the color of a surface but also the shading of all lights in the scene on that surface. This process is called **render to texture** or **texture baking**.

The main idea behind rendering to texture is to take existing materials on a model, calculate all lighting in a scene, and bake it on top of the original material. This way the new material contains light shading and other effects in the scene. Since you can bake a great deal of information from a scene inside a material, you will also see that the same technique can be used to take high polygon count

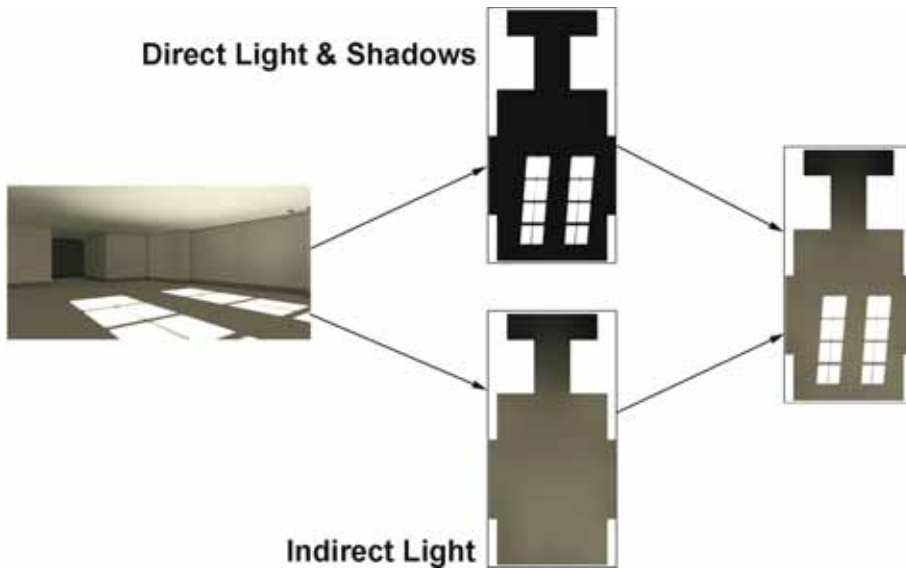


Figure 8-15. Light maps can be separated into direct light & shadows and indirect light then re-combined together. It allows much more flexibility when combining them back.

The **Shadow map** is a representation of the shadows on an object. The Shadow map is created from shades of gray. The map is pure white where there is no shadow and darker where there are shadows. It can be used as a mask later for adding shadows to a light map. If you are using a Lighting map as opposed to a Shadow map to capture shadows, you will capture the color of the shadows. Therefore, if a light of a certain color fills the shadow of another colored light, the Light map will render those colors. The Shadow map gives only the intensity of the shadows.

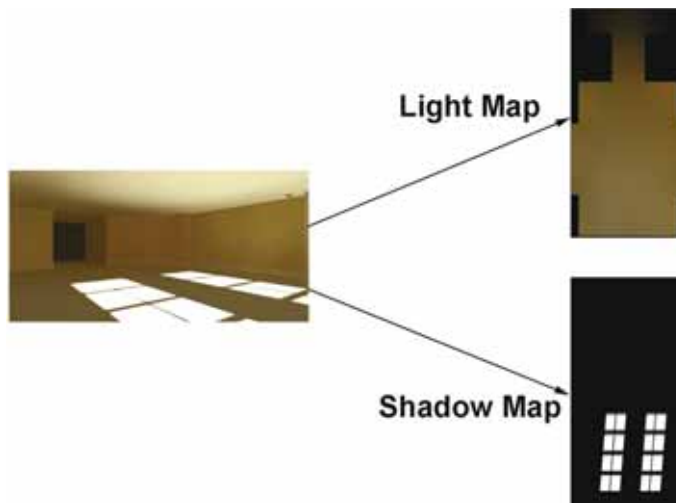


Figure 8-16. Shadow maps are masks used to define where the shadows are on the object.

Another map that is quite important is the **Ambient occlusion map**. Ambient occlusion is a special effect based on the level of exposure of a surface to the environment. This level of exposure is based on the angle of view of the environment. The larger this angle the brighter the surface.

Advanced mental ray Lighting

written by Darren Brooker

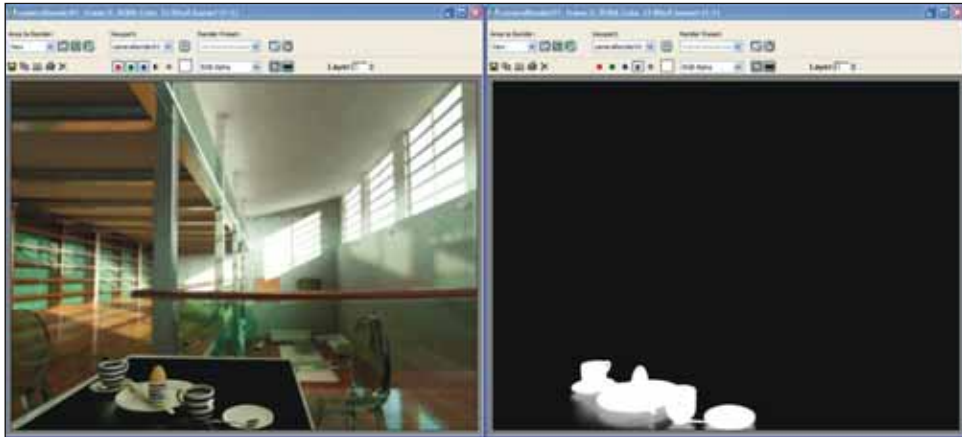
technical review by Jason Addy



WHY MENTAL RAY? SIX YEARS ago, mental images, the creators of mental ray, won an Academy Award for Technical Achievement, recognizing the contribution its advanced renderer had made to motion pictures. Despite this fact, not so many releases ago, the answer to the question “why mental ray?” was not an easy one. Several of the third-party renderers for 3ds Max offered functionality that in many areas was either more advanced than mental ray or was easier to use. Add to this that its documentation was poor, its licensing restrictive, and its workflow much less well integrated, and you can see why renderers like V-Ray won a good market share and a loyal user base to boot.

However, a decade since it was first integrated with 3ds Max, mental ray is now so tightly integrated into 3ds Max that it has replaced the scanline as the default renderer within the package’s design visualization flavor and is fast becoming the standard rendering platform across the entire Autodesk portfolio. It has been steadily stealing a march on its rivals for a number of years now, and, if any answer was required today to the question “why mental ray?” then surely the 2009 Visual Effects Oscar for the incredible work on **The Curious Case of Benjamin Button** provided it, neatly book-ending the Technical Achievement recognized by mental images’ own Academy Award.

14. Finally, you should set your output resolution back to 800 x 600 and press the **Render** button. This will return an image with your spherical image in the background and a black table surface roughly matching the table in your backplate. If you hit the Display Alpha Channel button to examine the alpha channel, you can see that the alpha only includes the foreground objects and their interaction with the table surface; the background is just acting as a reference. You might note that this background area of the image is not of great quality, and you could improve it by changing the **Filtering** from **Pyramidal** to **Summed Area** in the **Bitmap Parameters** rollout in the **Material Editor**.



15. When saved, it is necessary to save this image in a format that will save its alpha channel, like .tif. Brought into Photoshop and copied into a new layer as your final image from the last tutorial—**ch09-01g.jpg**—with the alpha channel copied into a mask for this layer, you can see that these foreground elements sit comfortably within your backplate, and their lighting matches the original scene elements.



reactor

written by Mir Vadim

technical review by Brian Kitts



REACTOR IS AN INTERACTIVE PHYSICS engine from **Havok** that was integrated into 3ds Max 6 and has since become a robust and efficient way for creating all kinds of dynamic simulations in 3ds Max. Thanks to reactor, many animators and visual FX artists from all over the world were finally able to create believable and realistic dynamic animations in an extremely short amount of time. Since its creation, reactor has been widely used for film effects, commercials, CGI trailers, and games. Though its use in the visualization industry has not been so ubiquitous, many advanced users have come to depend on it for their most sophisticated animations.

reactor allows us to create dynamic simulations with both rigid and soft bodies, as well as both cloth and rope. It supports different forces, like gravity, water surface, and wind, and constraints like motor, car wheel, dashpot, spring, toy car, hinge, and even rag doll constraint. Since reactor is a pretty big and complex tool set, one small chapter is not enough to cover all reactor's capabilities. However, after reading this chapter, you should have the basic skills needed to do many of the things veteran reactor users do in their most sophisticated animations. Furthermore, you should be much better equipped to experiment with additional reactor features in an efficient manner.

Overview of the reactor Interfaces

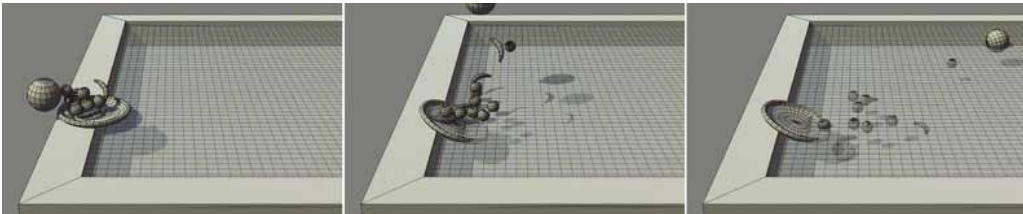
reactor's user interface is very well integrated into 3ds Max. You can quickly set up simulation in different ways using toolbars, menus, floaters, and quad menus. Some of them duplicate each other, so you can always choose which ways you like to use most in your workflow. This section illustrates the most widely used interfaces for simulation setup.

Utilities Panel

To begin using reactor, go to the **Utilities** panel and click on the reactor button. When you do, several rollouts become available, as shown in Figure 10-1. Here you can set the most important simulation

4. Turn off **Auto Key** mode.
5. We also need to decrease the strength of the ball's bounce, so set its **Elasticity** to **0.4** (using the **Open Property Editor** icon).
6. Click on the **Preview Animation** icon and start simulation, pushing the **P** key.
7. Now, if you did everything exactly as described, the ball should bounce off the ground and hit the dish. Most of the fruit should fall into the water.
8. Close **Real-Time Preview** window.
9. Click on the **Create Animation** icon in the reactor toolbar to create the final dynamic simulation and bake all the objects' animation in keys.
10. Click **OK** to continue.

Congratulations! We just created a basic rigid body simulation. We used an animated ball that hits the ground, bounces up, hits a dish with assorted fruit that falls into the pool and floats over the water's surface.



Summary

Hopefully, it was not as difficult as you might have first thought to create a dynamic simulation using reactor. Step by step you learned the reactor user interface as well as the most important simulation properties. You prepared geometry, set up simulation and physical properties, analyzed the world, tested and interacted with a simulation using Real-Time Preview mode, and, finally, baked the animation with keys. Using these same procedures, you can create dynamic simulations as complex as you want. Just be patient and take care in preparing your simulation.

Particle Systems

written by Pete Draper

technical review by David Duberman

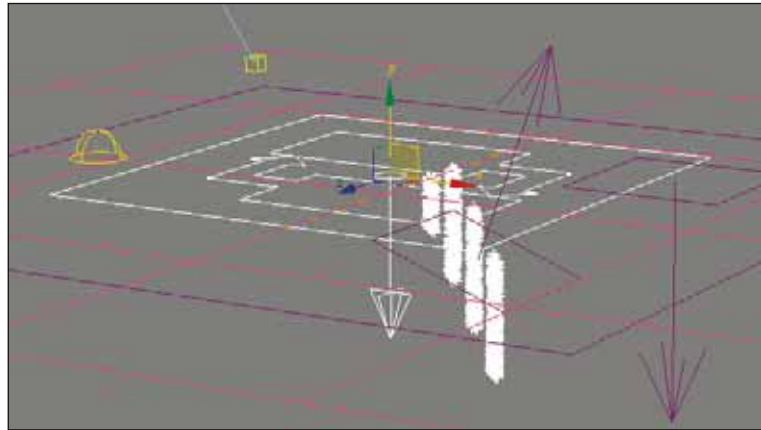
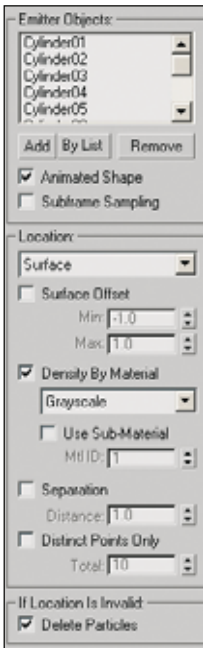


IN THIS CHAPTER, WE WILL cover the basics of Particle Flow and its associated space warps that are relevant to the system. We will break down the base fundamentals of how the (standard) system works and how to incorporate the aforementioned space warps into the particle system before taking steps to the next level by building three main systems; one to create a tree planting system, one to create a timed fountain by animating materials, and one to create a flock of animated birds with basic behavioral characteristics.

An Introduction to Particle Flow and Space Warps

Particle Flow is a node-based, event-driven particle system that allows virtually unlimited combinations of operators, tests, and scripts to (pretty much) create any type of particle system you desire. The system is constructed by adding in particle display and/or behavior properties in the form of **operators** within a main collated **event**. You can also perform **tests** to see if the end condition results are true or false and pass the particle to the next event with the same or different behavior and/or shape properties. As with anything, there are some limitations as to what you can produce with the base kit, but, all-in-all, it is one of the most advanced systems available. However, to get the most from it, you really need to understand its structure and “flow” of data. You will come to that shortly when you start analyzing the structure of the main **Particle View Event Display** canvas.

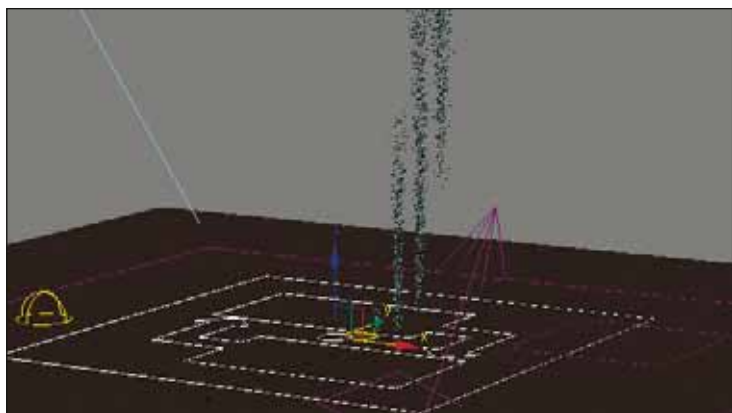
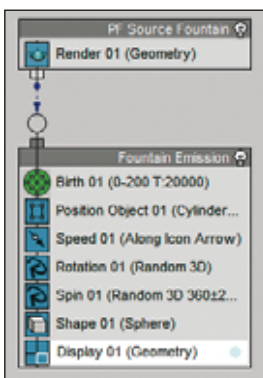
Unlike the “legacy” systems such as **PArray** and **Spray**, Particle Flow is not limited to a set number of properties—you can add and build up the system as desired, creating event loops, spawned particles, different particle shapes, scene geometry referencing, animation retiming, and so on, all within one single system. The beauty of it really comes when you feed a system into one you have already constructed, in essence sharing particle properties and saving valuable time. However, after all of this “bigging up” of the system, it is not without its quirks, not covered in the standard manual that ships with the product, which you should be aware of in order that you do not fall foul to the perils and pitfalls that some of us encountered when we first started using the system. Assigning forces



In the **Speed** operator, set the **Speed** value to **800** with a **Variation** of **200**. In the **Direction** group, enable **Reverse** to get the particles to travel in the opposite direction and set the **Divergence** to **2** to create a slight spread. Scrub through the animation to see the particles emitting as desired, animating across the length of the cylinders, and travelling vertically upwards (with a slight spread).

To get the particles to spin, right click on the **Rotation** operator and navigate to **Append** ➤ **Operator** ➤ **Spin**, and then reposition the added **Spin** operator beneath the Rotation operator. Set its **Variation** value to **200** to break up the spin animation so they are not all spinning at the same rate.

Select the **Shape** operator. Set the **Shape** to **Sphere** and set the **Size** to **3**. You currently cannot see the size of the particles in the viewport as the **Display** operator is set to **Ticks**; select the **Display** operator and set its **Type** to **Geometry**. While you are here, change the color switch to a light blue, for example **RGB 215,255,253**. Ideally, you should assign a material to the particle system by using a Material Static or Material Dynamic operator, but this color will do in this example.



Rigging

written by Michele Bousquet

technical review by Beau Perschall



A **RIG** IS A COMBINATION of tools and objects for animating a jointed model, such as a robot arm or a character skeleton. The term comes from the film industry, where a special effects artist makes a mechanical hand or body (a “rig”) and operates it with electronic or mechanical controls rather than moving each piece by hand.

In 3D, a rig is designed to move or rotate several objects at once to simulate real-life movement. All animation keyframes are held by rig controls rather than individual pieces of the rig, reducing the number of keyframes overall and making it much easier to control and edit the animation.

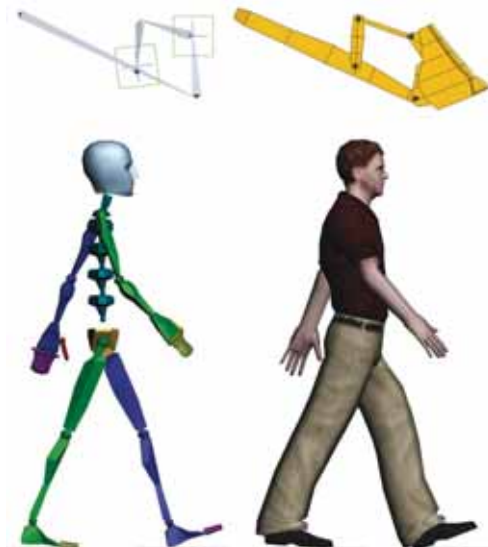
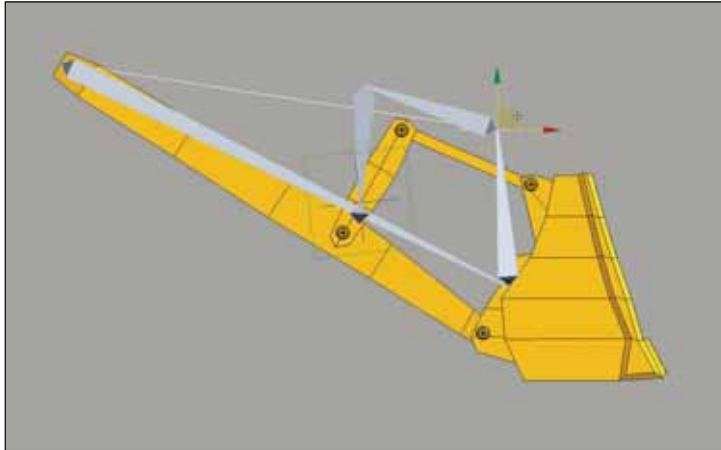
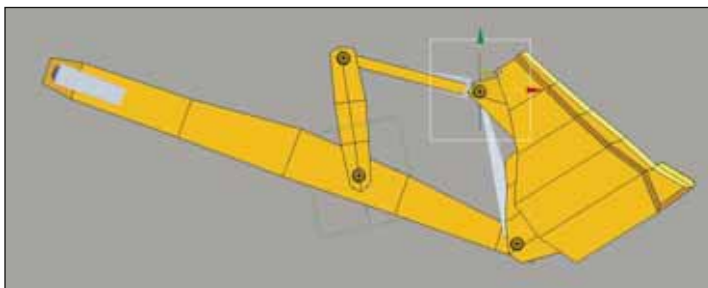


Figure 12-1. An example of an object suitable for rigging

6. Link **IK Chain02** to **Dummy01**.
7. Link **Dummy01** to the **first bone**. Now when you move the first IK chain around, the entire assembly works as expected.



8. Create a **Dummy** and align it to the crosshairs of **IK Chain01**.
9. Link the **IK Chain01** to the **Dummy02**.
10. Move the bones, IK chains, and dummy objects so that they fit inside the arms and digger.
11. Link each arm to its corresponding bone and link the digger to its bone. Now you can control the assembly just by moving the upper dummy object. You can find this completed rig in the file **ch12-04.max**.



12. This concludes the exercise.

Mechanical Rig in Practice

To use this rig in practice, you'll need to attach, link, or group objects in the scene to work correctly together. You might also need additional bone sequences and IK chains. For example, the bulldozer shown in Figure 12-3 and 12-4 has additional pieces attached to its arms, and the arms need attaching to the truck body in such a way that the rig still continues to work. You can find this bulldozer in the file **ch12-05.max**.

Revit Integration

written by **Scott Rosenbloom**

technical review by **Tod Stephens**



SLOWLY BUT SURELY, **REVIT ARCHITECTURE** is becoming the standard for architectural design. A major difference between software like Revit (known as **BIM** or **Building Information Modeling**) and AutoCAD is that in traditional CAD you draw with lines, arcs, and circles, whereas in BIM, you create objects such as walls, doors, and windows. With the release of Revit Architecture 2009, the rendering engine **mental ray** was introduced. Although you can achieve photorealistic results using it in Revit, 3ds Max provides additional advanced tools for modeling, rendering, materials, lighting, etc. In previous versions of Revit, the only way to bring data into 3ds Max was to export it as a DWG and then use the **File Link Manager** (the **File > Import** command could also be used but would cause a loss of Revit-based 3D data). There were inherent problems using this method, including the inability to bring over material image maps with accurate mapping. Autodesk has introduced the **FBX** file format as a go-between for its multiple 3D design products. The file format contains much more data than the DWG format, including the new **ProMaterials**, **daylight systems**, **photometric lighting**, and **daylight portals**.

In this chapter, we will first take a look at what Revit is and why it is known as BIM or Building Information Modeling. Next, we will get into the heart of this chapter, Revit integration with 3ds Max. We will begin by stepping backward by going through the DWG method of working between the two pieces of software. There are still several features of the FBX file format that need further research; therefore, it is worth knowing how the original method works. Next, we will talk about the main point of this chapter: the FBX Method. We will first create a house in Revit and include within it materials and lighting. Then we will bring it into 3ds Max using the FBX file format. At this time, we will step aside and take a look at the files that were created at the moment of import. Within 3ds Max, we will explore how the imported objects are organized and how to modify their shape and materials. Next, we will go through the basics of preparing and rendering the building. As I mentioned earlier, several features still exist that the FBX method lacks that hopefully will be included in future versions. One of these features is the ability to link the imported FBX file back to Revit (as you can with the DWG format). We will go through a work-around that will allow you to use this linking feature with the FBX file format. Learning how to work between Revit and 3ds Max will expand your toolset, giving you a more robust design process.

Imported Revit Object Organization

Let's take a look at what was actually brought in from Revit by clicking the **Tools** pull-down menu and choosing **New Scene Explorer** (Figure 13-17). Here you will see how the objects have been named and organized. Notice that the name of an object is exactly as it was in Revit. Additionally, at the end of the name, you see a number in brackets. This is the **ID** number and will become important later on when we update the design in Revit. The Scene Explorer also contains several other Revit-related columns that give information about the imported objects, including the **Revit Category** (wall, windows, etc.), the **Revit Family** (Basic Wall, Single-flush – referring to a window, Double-glass – referring to a door, etc.), the level the object exists on in Revit, and the specific **Revit Type** (Generic – 8" Wood, 36" x 48" – referring to a window, etc.). Also, notice at the very top of the list, there is an object called **3D View: {3D}**. This is the name of the view that you exported from Revit and has been converted into a camera in 3ds Max. At the bottom of the list, you'll see an object called **Compass**. If you click the + symbol to the left, you'll see another object below it called **SunandSky-002**. The angle, time of day, and geographic location of it was determined by adjusting the settings in the **Sun and Shadow Settings** dialog box in Revit (In Revit, **Settings** > **Sun and Shadow Settings**).



Figure 13-17. Scene Explorer listing imported Revit objects

Importing the Interior Model FBX File into 3ds Max

Now we will follow the same process to import our interior model into 3ds Max.

1. Click the 3ds Max symbol at the top-left, and click **Reset**. If you are asked, save the file and then click **Yes** when asked if you really want to reset.
2. Once again, click the **3ds Max symbol** at the top left and click **Import**.
3. Open the file called **3DView-FBXExport-Interior.fbx**.
4. When the **FBX Import** dialog box appears, notice that the preset is already set to **Autodesk Architectural (Revit)**. Simply click the **OK** button.
5. When the warning message appears, click **OK**. Now you will find that only the objects that were cropped in the view in Revit appear in 3ds Max.

Advanced Poly-Modeling

written by Todd Daniele

technical review by John Stetzer



THE MODELING OF COMPLEX OBJECTS is something that every 3D professional is bound to be required to do at one time or another. While many of the modeled elements encountered in architectural visualization are structural forms that are often square and linear, there are times when buildings, walls, furniture, props, and architectural details will require either a bit more detail, or that they are non-linear by nature. In many cases, the way to create this detail is through the use of subdivision modeling. Building a low polygon *base mesh* that will later be subdivided through the use of the **TurboSmooth** or **MeshSmooth** algorithms is a powerful way to create complex models with high levels of detail and smooth-flowing surfaces. I am sure everyone reading this has modeled and used TurboSmooth many times. I am also confident that there have been many occasions when your model has turned into something completely unexpected when adding the TurboSmooth modifier. The ability to create intricate models that will subdivide properly and create consistent and predictable results is what makes a modeler's skill level *advanced*. With this in mind, I am going to outline some of the key factors that make poly-modeling and subdivision surfaces more predictable, while also improving the quality and speed by which you produce your models.

Poly-modeling Theory

The **quad** polygon is the basic building block for any well-built model that is going to be subdivided. A **quad** polygon does not necessarily need to be square in shape, although a majority of the time it will be. Simply creating a 4-sided polygon when building a model is not going to guarantee that your model will look clean when TurboSmooth is added. There are other factors that need to be considered: polygon size, spacing, and polygon flow all impact the look of the final product when subdivided. Having long quad polygons next to short quad polygons can quickly turn a model into something that doesn't react as expected when subdivided. An even quad polygon size and spacing is not only important structurally but also will make the process of texturing your model a much more predictable process. As a general rule, using all quads for subdivided models is a good habit to get into;

Modeling an Ornate Writing Desk



Figure 14-2. Reference photo for desk modeling exercise

It's time to implement some of what I have shown you up to this point in a practical exercise. I am going to take you through the process of modeling a writing desk. What makes this a good example is that the legs and hardware are not simple linear shapes. The leg has compound curves and a cut-away detail that make it a great candidate for using some of the techniques covered in conjunction with TurboSmooth. The handles and hardware have intricate details that are also quite challenging to model. When you are finished, you will have a working knowledge of techniques that can be applied to any highly detailed element. While the example you are modeling is a desk, the techniques could easily be applied to carved wood and stone elements as well as interior and exterior moldings.

Modeling the Desk Leg

1. **Reset 3ds Max.**
2. In the **Top** viewport, create a **Box** primitive with a length and width of **2"** and a height of **33"**. Give it **2** length and **2** width segments and **8** height segments. The next illustration (left) shows the box primitive in the perspective viewport.
3. Convert the box to an **editable poly** object.

Managing Large-Scale Projects

written by Spine 3D

technical review by Nils Norgren



MANAGING LARGE-SCALE PROJECTS — WHAT exactly does this mean? Large-scale projects typically require significant involvement from stakeholders, engross large budgets, and/or the workload is too overwhelming for one person or small group to handle on their own. Personally, I have dealt with two different scenarios: One was working on a project with multiple phases of a large development concurrently being designed, e.g., **Project City Center (MGM)** shown in the image above, and the other was working on **Pier 27 (Cityzen)**, an average-sized project with many deliverables due in an extremely short time frame.

The first scenario, Project City Center involved producing 10 minutes of HD animation and 40 still renderings in approximately 100 days for a design that involved 6 high-rise towers including outdoor amenities, a mega retail center, intermodal transportation hubs, and a casino, all from different world-renowned architects, that included additional modeling of the Las Vegas strip from reference materials. A project like this would be considered large-scale because of the sheer magnitude of 3D modeling required and all the coordination involved in gathering files, comments, and approvals from the different parties involved.

The other scenario, Pier 27 (Figure 15-1) consisted of one luxury resort/condo tower with amenities, and the scope included 3 minutes of HD animation and 15 still renderings to be completed in 30 days. As you can see, even though the project itself is not large, the expected deliverables and the compressed timeline made it a managing nightmare that would required as much, if not more, attention and personnel than the first case. No matter how large or small the project, basic guidelines are to be followed to properly execute any project regardless of size or time constraints. In this chapter, we will go into detail with procedures that have been tried and tested personally and with my production team on many projects successfully completed.

be necessary to have different FTP sites for different stakeholders, especially if they are competing against each other for future components of the job. This method is much faster and more effective than requesting files via a shipping carrier.

Internal Communication

The project manager is the person who will communicate all of the client's requests, concerns, and ideas directly to the production team, in a way becoming the client's representative for your production team. With this responsibility, it is important that the PM is knowledgeable in all aspects of the project. Weekly update meetings should be a standard practice so that the PM can keep a close eye on the different aspects of the project as well as follow the progression of the timelines. Often artists are not the best judges of time, and trusting your team to follow a schedule without PM supervision can be a costly mistake. The PM's role is tricky because he has to juggle the best interests of the client and your company, and it is likely that your team will sometimes see him in a negative way when he acts on the client's behalf. You cannot let this affect you because at the end of the day making sure the projects progress smoothly is also in the best interest of the company and is only achieved if the client is happy. Sometimes artists do not realize this, and it should be communicated as diplomatically as possible.

Communicating client comments and requests to other team members: During the course of a project, a client and his team of consultants will constantly feed information to the PM. As mentioned earlier, it will be the PM's responsibility to organize and to distribute this information to the appropriate departments.

Another responsibility for the PM is to transfer the client's requests in a format that will be easier to understand for the production team. When a request is submitted, the PM will coordinate to whom to forward it and also the best way of communicating this request.

Typically, the PM will generate an "edit" or a visual mark-up of a request to forward to a team of artists, thus streamlining the process and leaving no room for error. (Figure 15-8)

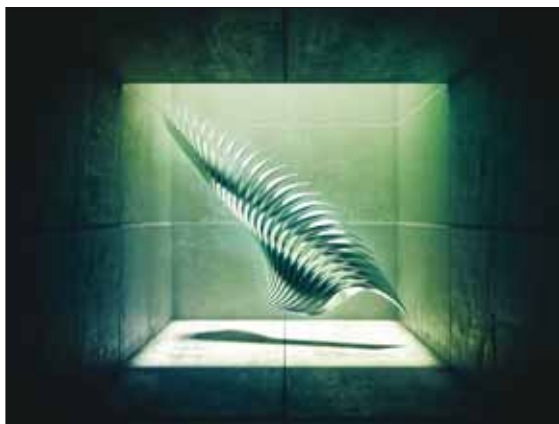


Figure 15-8. Rendering mark-up

MAXScript

written by Markus Boos

technical review by Lukas Dubeda



"Do I really need to learn to program with MAXScript?"

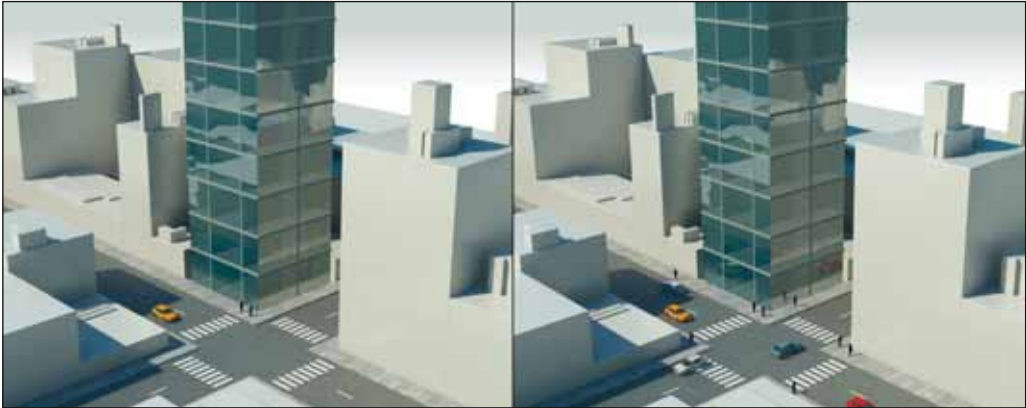
THIS IS A QUESTION MANY 3D artists ask themselves when they hear people singing the praises of 3ds Max's internal scripting language **MAXScript**. Today's software has become much more artist-friendly than it was just a few decades ago, and some consider the need for programming like taking a step back in time. However, as software becomes more complex and the 3D industry becomes more competitive, the need to streamline workflow and maximize efficiency has never been greater.

Larger shops might have a technical director or R&D team available to write special tools and extend off-the-shelf software with the functionality needed to finish a project in time and on budget. For most of us, though, it is up to individual artists to research ways of solving problems and coming up with appropriate solutions.

Quite often, 3ds Max's internal scripting language, MAXScript, can help you reach your aims in an elegant way. In the simplest case, you can "remote control" the application by feeding it written commands or playing back a procedure that you recorded using the mouse beforehand (hence, the name "scripting" language). In more advanced cases, MAXScript gives you the ability to create anything from small helper programs that solve a single problem to completely customized toolsets that are specifically adapted to your workflow or pipeline. These tools can then become integral parts in ensuring a smooth production and can give you the ability to react quickly to client changes or other unforeseen variables. Especially in large-scale projects, having scripting knowledge can be a huge plus.

The following sections will demonstrate that MAXScript programming, with its relatively straightforward design, is easy to comprehend--even for people with a non-technical background--and that it is unjustified to think of it as a "three-headed monkey" that is difficult to tame. I will introduce you to MAXScript by describing what it can do for you, after which you will start writing your first lines in the context of simple practical examples. Next, I will explain the most commonly used language features. After that, you will create your own little toolset to automate the setup of ambient occlusion and

This will provide you with a new object that you can transform to the desired position and orientation. Create one or two cars in each lane and a few people on the sidewalks as shown in the illustration.



Bonus: Assign each car a random-colored standard material. This time, use the `copy` command to assign the material to the variable `m` (there will be more about variables in the next chapter). Then assign the copied material to the new car.

```
m = copy $Car_01.material
$Car_02.material = m
```

Since the car body's material is the first inside a Multi/Sub-object material, you need to access it as follows:

```
$Car_02.material.material1.diffuse = ...
```

Set its diffuse color property using RGB color values. Save your scene when you are done. You will continue working on it in the next assignment.

Animating Objects

Now let us create some animation with MAXScript! A good way to figure out the way something is done in MAXScript is to list the steps you would take to do the task in the UI, then translate each step to MAXScript commands. The most straightforward way to animate is to do the following:

- turn on Auto Key mode (1),
- move the time slider to a certain frame (2),
- change the desired property or transformation (3)
- turn off Auto Key mode (4).

You know how to do part 3 in MAXScript; thus, you need only to research parts 1, 2, and 4. You will do this using a feature called the **MacroRecorder**. When the **MacroRecorder** is on, it records each step you do with your mouse or keyboard as MAXScript commands in the upper part of the listener. For simple operations, it can be a comfortable alternative to searching the MAXScript reference.

Camera Matching

written by Mike Merron

technical review by Jon Hey



Copyright Uniform

CAMERA MATCHING IS THE PROCESS of accurately recreating a real world camera within a 3D program to match a source photograph or moving image sequence. It is a vital step in the process of creating photomontages.

Photomontages are a combination of a rendered image (either drawn by traditional methods or created using computers) and a photographic background. They are an important part of marketing and planning as they show a product or building with real world context. Camera matching is used extensively in many areas of visual communication, not just architectural visualization.

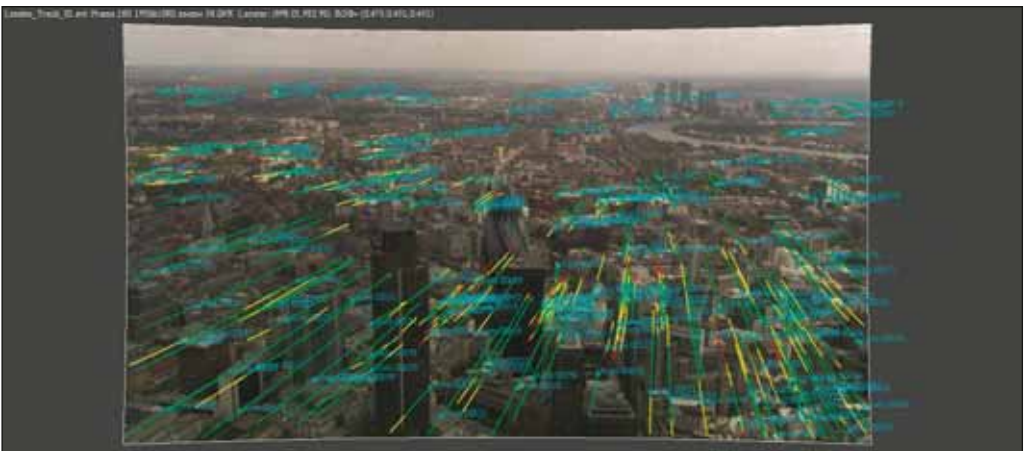
Camera matching has become an important part of architectural visualization for a number of reasons, e.g., informing the public of what a new scheme will look like. It aids the architect with the design process and also enables city planners to decide if new schemes will be approved.

As these hyper-realistic images are being used more and more, it is now essential that they are portrayed as accurately as possible. This is especially true when you are working on images that will be used in planning proposals. The methodology in this chapter is by no means the only way to achieve accurate results. As new technologies become available, this process will likely be more automated and amazingly accurate.

In the following illustration, the highlighted feature is tracking the intersection where a foreground building meets background building and, as the camera moves through the space, the parallax between the buildings is obvious to us, but for the tracker the feature is a nice, solid track. The problem is it isn't actually a definable point in space, and this point is constantly moving. Like the bus feature, it needs to be manually deleted.



It is also possible to add user-defined features to aid the camera solving; however, for a straight-forward track like this, we will achieve good results with the auto features.



Now that the Feature tracking is completed, the next illustration gives a really good indication of how the camera is moving through the shot even though this is only a 2D process so far. We can already infer the motion.

Now we are ready to solve the camera motion. The solver will attempt to estimate the path of the camera through the scene as well as estimating the 3D position of auto and user features. PFTTrack will calculate any missing details, such as the focal length, that were not added to the camera parameters. A point-based depiction of the scene will be constructed and can be seen below in the 3D viewer. We can also see the camera path that has been generated.

Green Screening

written by Smoothe
technical review by Alex Menendez



GREEN SCREEN IS A TERM used to describe the mixing of two images or frames to form one complete image. The most common scenario is the combining of a foreground element with a background image, such as 3DATS co-founder Brian Zajac conducting a presentation in a virtual environment, made possible by being filmed behind a colored desk and in front of a colored screen. The colored environment is then removed, isolating the presenter who is then composited with a virtual studio created with 3ds Max. A number of other terms are used to describe this technique, including **blue screen**, **keying**, and the **color difference system**, but another term used more commonly throughout the CG industry, and one we will focus on in this chapter, is the term **Chroma Key**.



Figure 18-1. A typical chroma key application: 3DATS co-founder Brian Zajac conducting a presentation in a virtual environment

Even though the chroma key technique is relatively quick and simple to execute, there are a number of guidelines that you should bear in mind when producing this type of work. We have known



Figure 18-14. Initial comp from pulled key

The primatte keyer is a slightly different keyer and is a much more complicated operation. The flipside of the complication though is that it is also a marvel to behold. An entire book could and probably has been written on the more complex keyers alone. In a section of one chapter, I can give you only a practical overview. Once the color has been chosen, the image will look as if the majority of the foreground has been keyed out. Fear not: the beauty with primatte is you can sample a section to fill back in, like a negative of our original pick. By using the clean foreground noise button in the primatte tab you drag your cursor with the mouse button applied over the areas you want to remain in your foreground. This can be done as many times as you want to fill in this area. I usually do this while looking at the alpha channel to give more of an idea of what areas are still missing. If you get a little too keen on this and pick some green, all you need to do is use the clean background noise button to re-pick the chroma screen. The other operations on this tab and the other parts to pulling your key in primatte are as follows: The spill sponge means the background color component in the sampled pixels (or spill) within the image window is keyed out and removed for the color region selected; also, the matte sponge makes the color sample picked 100 percent foreground. This tool is usually employed to quickly remove stray transparent pixels that have appeared during the chroma keying procedure. It is a quick and easy way to make final adjustments to a composite. Detail restore makes the completely transparent background element sampled translucent; this is effective for restoring fine detail, such as hair. The spill, matte, and detail +/- buttons are fairly self-explanatory as they gently add or remove the pick to or from your original picks in these sections. The impressive thing about the more advanced keyers is the result in all aspects --matte, spill, detail and such--with some sampling of the correct areas of the image. This creates a brilliant result that needs little to no refining.

Digital Compositing

written by Gary M. Davis

technical review by Glen Whelden



DIGITAL COMPOSITING IS THE CREATION of an image by manipulating and/or combining one or more sources using image processing software. Within most digital compositing applications, these 2d image manipulation processes, filters, and effects can be quickly and easily accomplished on a single frame and, perhaps more importantly, just as easily on moving image sequences of video or animation (batch processing). Footage types can include those captured from a real-world camera, rendered computer graphics, or any combination of the two. So, what is the advantage of adding digital compositing to your 3ds Max production pipeline?

As it relates to design visualization, the goal with digital compositing is often to combine real world photography or moving video with rendered CG buildings, vehicles, bridges, and so forth. In this scenario, the artist wants to make it appear as if the final result was a photograph or video taken with a single camera. Alternatively, there are times when compositing will focus solely on CG rendered frames. That is, no real-world photographic plates are utilized in the creation of the final piece, and every pixel is computer generated. These images' types can range from stylized realism, technical illustration, to photorealism.

Whether you want to add CG objects to real-world photography, or you are creating 100% synthetic images, there are reoccurring advantages to using digital compositing in the production pipeline. Since compositing tools are essentially a series of effects and filters applied to 2d raster image frames already rendered out of 3ds Max, the effects can be considered 2d operations and are, therefore, exponentially faster for the CPU to process and calculate. Depending on your hardware and software configuration, you can often make changes in compositing applications and see the results in real-time with no rendering lag. This means you can have many versions and variations of a final deliverable, with each of them originating from a single, "true" 3d render out of 3ds Max. When it comes right down to it, compositing can save you significant production time (money) as well as visually take your work to the next level in a matter of minutes.

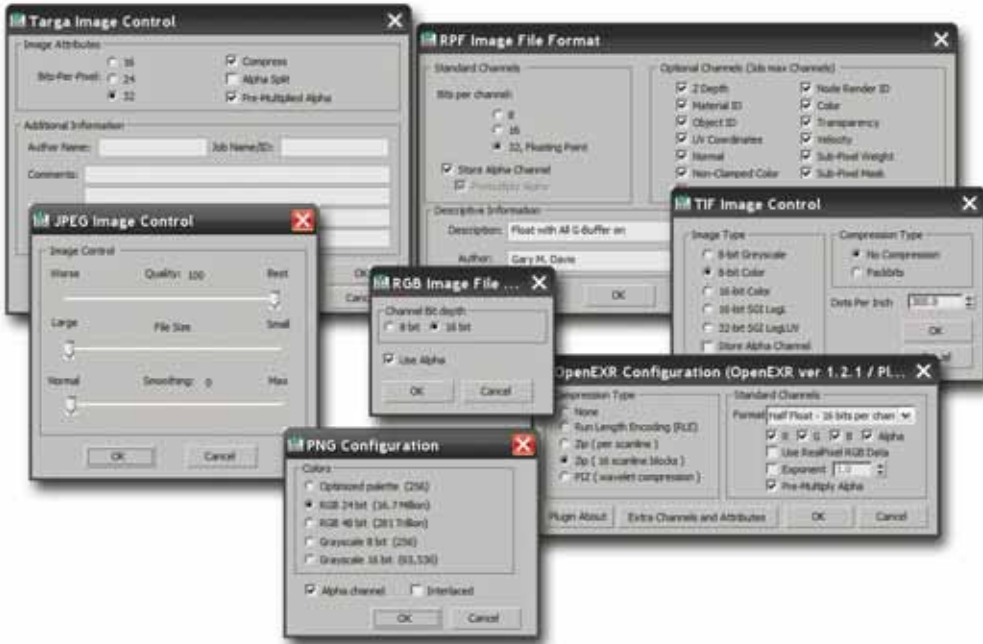


Figure 19-21. Common file output types

Alpha Channels

The importance of an alpha channel for compositing purposes should be clear by now. These are essential to a compositing toolset, and are available as an option for many different file formats upon output from 3ds Max. A few notable exceptions are JPG, BMP, HDR, CIN, and DPX. By their very nature, they are not able to contain an alpha channel and, therefore, are typically not the ideal format to save to when rendering from 3ds Max.

There is one consideration to make when outputting an alpha channel: whether or not to use what is called **premultiplied alpha channels**. A premultiplied alpha simply means that when a render pass is created in 3ds Max, the object's edges, and areas of transparency that intersect the 3ds Max background environment, will be calculated and antialiased to the colors of the environment background of your scene.

EXR and TGA are two file formats where premultiplied alpha channels are an option. Some file formats support this option and others do not. PNG files from 3ds Max, for example, are always premultiplied against the background environment color, and there is no option to enable or disable this within the file format save dialog. Ultimately, there is no "right or wrong" when deciding whether or not to premultiply, as long as you know the means of workflow you are using and understand how to identify this workflow at the raw asset/footage level within your compositing application. Performing several tests with primitives will help you understand and establish your preferred workflow.

Video Editing

written by Kim Lee
technical review by Alan Skinner



WHEN MOST PEOPLE THINK OF video or film editing, they picture someone sitting at a computer at the end of the project putting all the footage, sound, and titles together while the director and producers lounge about behind them on sofas, munching on snacks or playing video games. Occasionally, the editor will turn to the director to discuss a sequence and perhaps watch a portion of the project before returning to clicking away with the mouse. Eventually, when everything is complete, they all go home to prepare for the red carpet unveiling of their masterpiece and the inevitable success to follow.



Figure 20-1. A typical video editing workstation
(Image courtesy of Gold Crest Post)

Offline CODECS

When I refer to offline codecs, I typically mean codecs used for video files of animatics and rough cuts as well as other applications such as internet and email delivery. This short list is intended as a guide for those unfamiliar with the various codecs and, by no means represents the only codecs that can be used for offline applications.

- **H.264** – This is currently one of the most popular codecs in use today because of its high quality and small file size. It is equivalent to MPEG-4 AVC and MPEG-4 Part 10.
- **Sorenson 3** – This is a variant of H.264 commonly used for its high quality and small file size and typically used in Quicktime files.
- **Photo Jpeg** – This codec is good for getting image quality approvals when the ability to step through frame by frame is desired but not ideal for smooth playback.
- **MPEG** – This codec is a popular one for its high quality and small file size. MPEG-2 is the standard encoding for DVD. MPEG-4 is a more efficient version that is widely used today.

Online CODECS

Online codecs can be considered any codec that is capable of visually lossless quality. The following is not an exhaustive list of all possible online codecs but a sample of some popular ones in use today.

- **Animation** – This codec used in Quicktime results in high quality image at the expense of large file sizes but can be put to good use especially when moving footage back and forth between applications and between PC and Mac platforms. Depending on the footage, it can be a good alternative to using an uncompressed codec.
- **Uncompressed** – Clearly, the highest quality image can be achieved with this codec at the expense of potentially huge file sizes.
- **Apple ProRes and ProResHQ** – This is Apple's newest proprietary codec format that is, not surprisingly, Mac-centric but can result in high quality footage without the burden of uncompressed file sizes. The HQ (High Quality) variant encodes video at 10-bit as opposed to 8-bit.
- **DVCPRO50** – Created by Panasonic, this codec is widely used for electronic news gathering (ENG) applications and uses a video bitrate of 50 Mbit/s with a 4:2:2 chroma subsampling. Picture quality is comparable to that of Digibeta.
- **DVCPRO HD** – Also created by Panasonic, this codec uses a video bitrate of 100 Mbit/s with a 4:2:2 chroma subsampling. Be aware that it horizontally compresses the HD image from 1280x720 to 960x720 in the case of 720p and from 1920x1080 to 1280x1080 in the case of 1080/59.94i. For 1080/50i, it compresses the image to 1440x1080. Since it uses non-square pixels, it may affect your workflow for footage that requires VFX to be added, as covered later in the "Exporting for VFX" section of this chapter.
- **HDCAM** – Developed by SONY, it uses 8-bit 3:1:1 compression and compresses a 1080i image to 1440x1080 using non-square pixels.
- **HDCAM SR** – Also developed by SONY, it uses 10-bit 4:2:2 or 4:4:4 compressions with a video bitrate of 440 Mbit/s.
- **XDCAM** – This is another series of SONY-developed codecs based on MPEG-2 compression. There are multiple variants including XDCAM HD, XDCAM EX, and XDCAM HD422.