

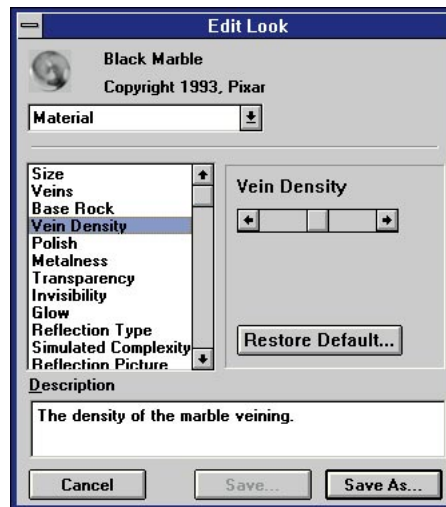
editing Looks



Ever wish you could change things like the swirl in the grain of your Oak Look, or the brightness of your Sky, or the color of your Marble's veins? That's why we provided a Look editor. This allows you to save customized variations of your favorite Looks, called *Instances*. Let's make one now.

Make a new Look variation now!

1. In Typestry, get some text in the project window and apply the Black Marble Look to it. Black Marble is in the directory sequence *Looks\Starter\Material\Stone\Marble*.
2. Click on the Edit Look button in the Looks tab in the Details window. The Look Editor dialog will appear.
3. Click on the Veins item. A black color swatch will appear.
4. Click on the black color swatch. This brings up the Color Picker.



make a new look variation now!

editing looks



5. Select a color to replace the black marble veins and click on OK. (Very very dark green could be nice...)
6. Now click on the Vein Density item. The Vein Density slider will appear.
7. Drag the Vein Density slider about halfway to the left end. As you might suspect, this will produce wispier, less dramatic veining when you render.
8. Click on the Save As button. When the Save As dialog comes up, open the *User Instances* directory (in the *Look Instances* directory).
9. Type in a name for the new variation you just created and click on Save. Now, in the Looks tab the Look image has a red “corner” turned down, indicating that it’s been edited.
10. Since you’ve made a change to the object, click on the Modify button to apply the change.
11. Now render using Excellent & Slow, and watch the difference!

Now. If you want to cut to the chase and get to more serious Look editing, turn directly to “Basic surface controls” later in this chapter. However, the more circumspect among us will simply continue on, thank you.

What the heck is this thing for, anyway?

Well, as you just found out, the Look Editor is for making variations of a Look, called *Instances*. It doesn’t make new Looks, just new Instances. One way to think about an Instance is to make an analogy between a Look and a font. Take the Times font for example. There’s Times Roman, Times Bold, Times Italic, and so on. There’s no plain Times: a font only exists as one of its typefaces. And the typefaces are just closely-related variants. Similarly, a Look only exists as one or more variants — the Instances. These are variants of a Look “Master.” If fonts used this terminology, Times Bold would be an Instance of the Times Master.

Each Master provides you with controls over certain aspects of a Look, called parameters. The Look Editor allows you to change the parameter settings and save the result as a new Instance. When your application uses a Look, the thing that actually makes the picture appear is the renderer. For a given object, the renderer needs to know generally what sort of surface the object should have, and it also needs to know the specific details about the surface. A Look Master supplies the general surface information, and the Look Instance provides the specifics.

Look Masters are referenced by Look Instances, and are never seen directly by the Look Editor. When you open a Look, you are actually opening an Instance, which contains a preset variation of its Master, and likewise when you save a Look, you are saving an Instance.

There are four kinds of Look Masters:

Materials. A Material controls everything about a surface except its bumpiness. So it provides color and shininess, determines how it reacts to light, and so on.



Reliefs. A Relief can create pits or cracks, waves, protrusions, bumps — it controls the shape of different areas of a surface. Obviously, a surface as a whole must have some shape to exist in the first place; a relief simply alters the basic underlying shape.



Lights. A Light, of course, controls how light from a light source behaves: whether its rays spread out or stay parallel, whether it can be used as a slide projector, whether it casts shadows, etc.



Environments. An environment determines what gets reflected by an object — what the environment around the object is.



Many Looks have parameters that allow for the most extreme variations. Believe it or not, a bumpy glass Look can be made to look something like granite, or like wavy water. But while you may be able to bend a Look to your will and make it look like something totally different, the Look Editor is most profitably used to “tweak” a Look: to change one of its colors, to make it reflect what you want it to, or to vary its smoothness, for example. However, the power and flexibility of the Look Editor cut both ways. As easily as you can transform a surface at

your whim, you can transform it into one that could never exist in the real world. This is, after all, computer graphics.

If you have a particular surface in mind that you'd like to create, you'll need to find a Look that shares some key characteristics that you can vary to approximate the new surface. However there is a virtual infinitude of surfaces, and you may find that none of the Looks you have will transform into exactly what you have in mind — that you simply “can't get there from here.” If you find yourself in this situation, take heart. You can scan in a picture of a surface, get the Pixar One Twenty Eight collection of seamlessly-tiling real-world textures, or create one with a paint program, and have the Look Editor apply it to an object, effectively giving the object a new appearance.

Be forewarned though — until you've gained some experience with the Look Editor, some controls may have unexpected results if you just play with them. But that's OK, go ahead and play — you might discover a new way to make an asphalt Look or something!

If you'd like a little more nerdy discussion of RenderMan issues like shaders and shading models, turn to the “RenderMan Expert Parameter Information for the Adventurous” at the end of this chapter.

How to make a new Instance

The process is pretty simple, really. First, create something *with a Look on it*. Then:

1. Select the object.
2. Make sure the Looks tab in the Details window is showing (select Looks from the Windows menu), and click on the Edit Look button. This brings up the Look Editor dialog. This dialog will have the parameters for the Look. You can also get this dialog from the Score window by clicking and holding on the space between the object's name and its Perfs icon. This displays a popup from which you can select Edit Look.



3. In the Edit Look dialog, if the Look is a combination of two Looks you can select which to view (the Material or the Relief) by using the pull-down menu at the top.
4. Clicking on a parameter displays that parameter's control, and you'll see a description of the parameter as well.



5. Adjust the parameters to change the Look. You won't see the results until you rerender.
6. When you're done, click on Save or Save As, and render.

Danger, Will Robinson! If you're using a Look with a Relief, the Relief will affect the faces and sides of characters differently. You can minimize the differences by keeping the height of any bumps low.

More Danger, Will Robinson! Some parameters may have an override icon next to them (a yellow triangle with an exclamation mark in it). The Object Info dialog allows you to override some of a Look's parameters (Opacity, Color, and Scale, for example). Any parameter changes you make in the Object Info dialog overrides any settings you make within the Look Editor.

- Note: In the Object Info dialog, changing the Color Override or Opacity Override controls doesn't change any settings in the Look. This is so you don't have to save a new Instance of a Look every time you want to see it with a new color or opacity. We made them available in the Object Info dialog because they may be among the controls you use the most. If you want to save these settings with the Look you must make the same changes to the Look's Color and Opacity parameters.

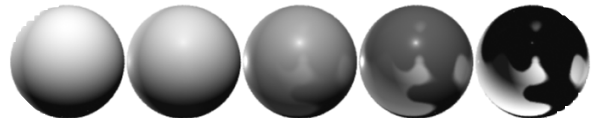
- Warning: The Color Override control overrides whatever is the *first* color parameter in the Look Editor. So for Black Marble, for example, it would override the Veins color, not the Base Rock color.

Basic surface controls

Believe it or not, with no more than three parameters, you can create reflective properties that mimic surfaces as diverse as matte, plastic, shiny metal, and glass.

Shininess

The shinier a surface gets, the more it will take on the color of the light shining on it, both from light sources, and from any reflected objects as well. It takes on this color at the expense of the color of the object itself, which will eventually dwindle away to almost nothing. In fact, if the slider were all the way to the right, you would see areas that are mostly whitish (from the reflections of fake objects) and blackish (from the black space between the objects).

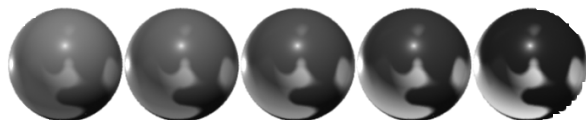


Increasing Shininess:

- Increases the degree to which the surface takes on the color of the light and any reflected objects — you'll see less of the Color parameter color.
- Increases the visibility of reflections.
- Decreases the size of highlights.

Metalness

When you look at a metal, you're actually seeing a rich combination of the color of the surface and the color of the light striking the surface. The Metalness parameter mixes surface color back into the mix of surface and light colors set by Shininess.



Increasing Metalness:

- Increases the amount of surface color (often set by a Color parameter) that appears in the surface.

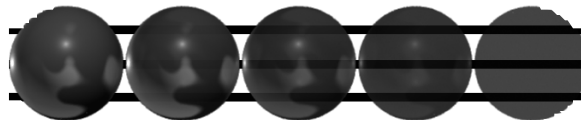
If you have no Shininess, Metalness will have no effect. Remember that Shininess trades surface color for light color. If no light color is present, you'll just see all the surface color, so adding more with

Metalness won't do anything.

Transparency

With transparency, the important thing is to remember that something that's completely transparent can still change the color of things you see through it. So don't be surprised when you make something completely transparent (the slider would be all the way to the right) and you still see some color.

Glass, however, is not completely transparent. By definition, something that's completely transparent allows *all* the light through. And that doesn't leave any light left over to reflect off a surface. Weird, huh? Only in computer graphics can you get a surface that's 100% transparent...



Increasing Transparency:

- Allows more background to show through, colored by the surface color.
- Decreases the visibility of reflections and highlights.



Don't get this confused with Opacity! Transparency controls how much light can get through an object. Opacity controls how "ghostly" an object is, how much it "exists." An object with no opacity may as well not exist — it has no effect on light, and light has no effect on it.

Using your own images in a Look: the Picture/Background Instance

Got an image you'd like to use in a Look? You can use your own image in any Instance that uses a "Color Picture" parameter.

- The Picture/Background Instance in the *useimage* directory (in the *simple* directory) is designed specifically as a starting point for making Instances that contain your own images.

If you have a drawing program or paint program that saves files in BMP or TIFF format, you can create your own library of Instances that use your images.

What can you do with your own pictures?

Parameters generally affect a whole surface equally. If you change a plastic Look's Color parameter, the surface will be all that color. Change Transparency and the whole surface's transparency changes. But if you'd like the plastic to have colored dodecagons on it, or to have some partially-transparent areas, you'd

need a way to control these parameters at different places on the surface. Well, you can use a picture to do this. You can use this as a way to control area by area a parameter that would otherwise apply to a whole surface equally.

You can supply two kinds of pictures: color or gray scale. Color pictures can do the job of any parameter that is a color. In practice, you'll probably be using color pictures just to "paint" an image on a surface using the Color Picture parameter, or to use as a slide or reflection in a Light Look. Gray scale pictures can be used to control all sorts of things: bumps, metalness, transparency . . . just about anything you might have a slider for. See "Gray scale pictures" later in this chapter for more on this.

Picture considerations

Image type. You can use as a picture any file you create with a draw or paint program, or even an image you've rendered to a file. When your file is ready, you must save it in BMP or TIFF format. Look parameters will work only with these formats.

Size. When you create a picture to be used in a Look, you'll get the best results if you follow a simple rule:

- Make the picture at least as big (in pixels) as it will appear in your final rendered image.

Let's say you have some 3D letters in an image, and you want to put your own TIFF image on one character. If the character will take up a portion of the final image that's only about 100x100 pixels, that's how big your TIFF image should be. If you made it much smaller the renderer would have to blow up your image to make it fit. And you know what happens when you scale up a bitmap image — that pesky blurriness starts creepin' in! If you made it much bigger, it would still look the same, but might take slightly longer to render.

- Using any picture in an Instance will increase the size of the Instance, sometimes dramatically. Practically speaking, using a color picture whose largest dimension is 128 pixels or less can add around 90K to the Instance. If that dimension is 129–256, add about 300K. If it's 257–512, add about 1000K. Gray scale pictures take up less space: 40K for a picture 128 pixels on a side, 100K for one that's 256-square, and 450K for one that's 512-square.

Shape (aspect ratio). You may want to create a picture whose shape approximates that of the bounding box of the surface on which you'll be using it. For example, if you want to use your image as if it were a picture in a frame you'll need to match the shape of your image to that of the frame — tall and skinny, square, short and wide, etc.

- When rendered, *any* picture used inside a Look

is one “unit” high, (about the default size of a letter) until you change the scale, either in the Object Info dialog or in the Look Editor! The smaller dimension will depend on the shape of the picture.

Getting a picture into an Instance (making labels)

This process actually applies to pictures of any sort — color or gray scale pictures, or reflection or slide projector pictures. If you're wondering how to make something like a decal, with invisible areas, turn to the section on “Single-color decals” or “Multi-color decals” later in this chapter.

Once you have a BMP or TIFF image, apply the Look in which you want to use the picture. For color pictures a good place to start is the Picture/Background Instance in the *simple* directory (in *useimage*). Then:

1. Click on the Edit Look button. This brings up the Look Editor dialog.
2. Click on the parameter that uses the picture (usually Color Picture), then on the Browse button. This brings up the Select An Image dialog.
3. Click on the name of the image (in BMP or TIFF format) you'd like to use.



4. Set Wrapping On if you'd like the image to repeat vertically and horizontally across the surface.
5. Click on Ok. The image will be copied into the Instance, and converted into a format directly usable by Typestry. The larger the image, the longer this will take. When you render, you will see the picture repeat across the surface of the object, both horizontally and vertically, like rows of tiles.

Gray scale pictures

You already know about applying your own color pictures. However, gray scale pictures have their uses as well.

Using pictures instead of sliders

Theoretically, a gray scale picture could do the job of any parameter that is controlled by a slider. While a slider sets a value for all of the surface, a picture can effectively set different values at different places on the surface — it controls a parameter in two dimensions. Practically speaking, you would rarely want such a thing for most sliders. So there are a few “all-purpose” Looks that allow you to use a picture for the most important parameters. You'll find out about these later...

The two ends of a slider represent two extremes. In

a gray scale picture these extremes are represented by black and white:

- For gray scale pictures, black is the left end of the slider, and white is the right end.

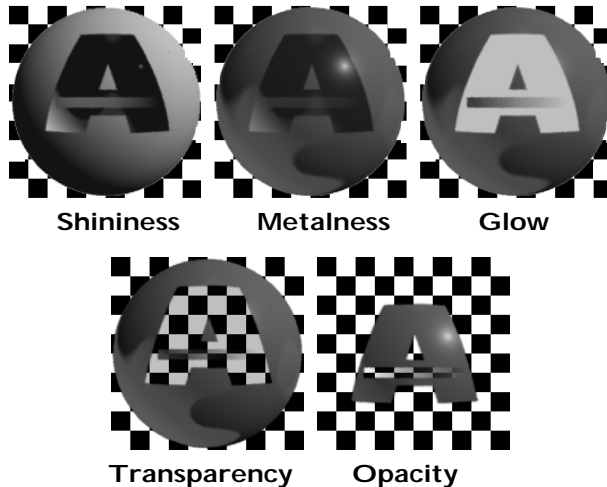
Take the Transparency parameter, for example. When the slider is all the way up, an object is completely transparent. But by using a Transparency picture you could control the transparency anywhere on the surface. You could make some areas completely transparent, some partially transparent, and others completely solid. All you would need is a black image with gray circles blending into white in the center. Or you could punch holes in an object by simply painting white circles on a black background and using this to control Opacity.

- While it's best to design a gray scale picture to be used in Typestry in gray scale, you can use any color image. It will automatically get converted to gray scale in the conversion process.

Let's say you're using a gray scale picture like the one below on a plain plastic-like surface:



Here are some examples showing what might happen using the image to control various parameters:



When you use a gray scale picture, its associated slider works slightly differently: it exaggerates or diminishes the effect created by the picture. The slider affects the whole picture, effectively making it brighter or darker.

If you need the utmost flexibility in using pictures to control things, use the Pictures as Anything Instance in the *expert/useimage* directory. In addition to using pictures for many of the most common parameters, this Look uses a picture for Opacity, which is ordi-

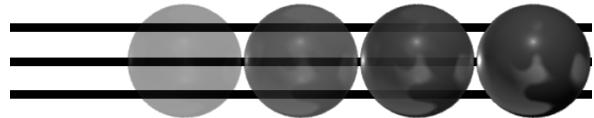
narily set within the Object Info dialog.

Using Opacity

Opacity determines how much a surface “exists.” Don’t get this confused with Transparency! An invisible object may as well not exist — it has no effect on light, and light has no effect on it. A transparent object can still color the light passing through it, and can still reflect, to a greater or lesser degree.

Decreasing Opacity:

- Increases the visibility of the background; turning Opacity all the way off makes a surface completely invisible.
- Decreases the “existence” of a surface, making it more “ghostly.”



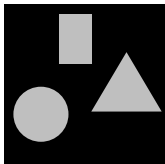
Using two or more pictures together

If you’re using two or more pictures in conjunction with each other in the same Look, you’ll need to bear a few points in mind.

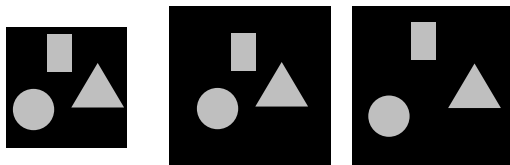


- It's best to make all the pictures exactly the same size. This way they'll all match up at every point, and you won't be confused about what might be overlapping what.
- There's only one Size control, and it scales all the pictures together! If you want to change the scale of one picture independently of the rest, you'll have to do it in a paint package. But remember — reducing the size of the picture itself is different from reducing the size of the individual elements of the picture. In the latter case there will be more “empty space.”

Consider the shapes in this image:



There are three ways of reducing their size:



In the first image, the whole original image has simply been scaled down 75%. However, remember

that it will still cover the same area as the original image, since within Typestry all images are one unit wide. This effectively provides fewer pixels to try to achieve the same effect as the original. As a result, it will be blurry where edges line up.

In the middle image the overall size is the same, but the shapes have shrunk 75% as a group and moved toward the center, relative to their original positions. So they're out of alignment with their counterparts in the original image.

In the right image the overall size is the same, and the shapes haven't moved, they're just 75% smaller. Their centers will match those of the original image, but they'll just be smaller.

So be sure to consider what effect you're trying to achieve when you resize one of a group of gray scale pictures.

Relief pictures

Sometimes, instead of using a Relief calculated by an Instance you might want to customize the bumping of a surface. If the Combination Look you're using doesn't supply enough control in this regard, use the Pictures As Anything Look in the *expert\useimage* directory. This allows you to use a gray scale picture in the Relief Picture parameter.

- For a Relief picture, white means “bump out,” black means “bump in,” and 50% gray means “no bump” will appear on the surface.
- The Relief Height slider sets the distance (in “units”) to bump the surface.

Danger, Will Robinson! When you use a Relief on an object, the joint between the faces and sides may have cracks or holes. If you see this, try either lowering the Relief Height or using a bevel that doesn't make a sharp corner at the face.

Multi-color decals

One thing you can do with an Opacity picture is make a decal. This is an image with areas that are invisible. You can get interesting effects by applying a decal to a wall, or, scaled down, to letters.

You need two pictures: one for the colored areas, and one to create the invisible areas where there is no color. But the good news is that the second picture is a simple variation of the first. Ordinarily, you would create your color image first. Then you'd need to alter it in a program like Photoshop to create the Opacity picture. All you need to do is make any colored areas white, and the rest of the image black.

Here are a few considerations to bear in mind as you convert your image to an Opacity picture:

- If the colored areas are antialiased, be sure to try to preserve this when you convert the image to gray scale.
- Do not resize the image at all — the color and gray scale versions must match each other precisely, except for color.
- If you want the colored areas to be only partially visible, be sure to use a shade of gray instead of black. The darker the gray, the less visible things will be.

To make a multi-colored decal:

1. In Typestry, apply the Decal - Multicolored Instance in the *simple* directory (in the *useimage* directory) to the selected object(s).
2. Make sure the Looks tab is showing (select Details from the Windows menu). Click on the Edit Look button. This brings up the Look Editor.
3. Click on the “Color picture” parameter and click on the Browse button. This brings up the Select An Image dialog.
4. Click on the name of the image (in BMP or TIFF format) you'd like to use.
5. Set Wrapping On if you'd like the image to repeat vertically and horizontally across the surface.
6. Click on Ok. This converts the image into a for-



mat the Look can use, which may take a few seconds.

Now you need to get the Opacity picture so the right parts of the image will go away.

7. Click on the “Opacity Mask Picture - gray scale” parameter and then on the Browse button. This brings up the Select An Image dialog.
8. Select a grayscale TIFF or BMP file to be used as the Opacity Picture.
9. Set Wrapping On if you set it for the Color Picture, and click on Ok. This converts the image into a format the Look can use.
10. Click on the Save or Save As button and save the new Look.

If you use colors other than black, white, and gray in the Opacity Picture image, they will automatically get converted to their gray scale equivalents.

Notice that you still have control over the material parameters, so you can make your decal shiny, or metallic, for example.

Single-color decals

Here's how you make a single-color decal:

1. In the Look Editor, open the Decal - Single

Color Instance in the *simple* directory (in *useimage*).

2. Click on the “Opacity mask picture - gray scale” parameter and click on the Browse button. This brings up the Select An Image dialog.
3. Click on the name of the image (in BMP or TIFF format) you'd like to use to control the visibility.
4. Set Wrapping On if you'd like the image to repeat vertically and horizontally across the surface.
5. Click on Ok. This will convert the picture to a format Typestry can use, which may take a few seconds.
6. Use the general material parameters to control the type of surface used for the decal.
7. Save the Instance.
8. Be sure to set the color in the Object Info dialog, available by clicking on the Object Info button in the Looks tab in the Details window.

Reflection issues

In many Looks, you'll notice some Reflection parameters:

Reflection Type. This determines whether the surface has any reflections, and if so, what type.



Simulated creates light blobby areas on the surface. These are fake reflections, computed by the Instance. They are meant to give the impression of there being some surrounding environment other than blank featureless space. Reflection Picture uses the picture selected by the Reflection Picture parameter as the reflection. (See the Reflection Picture parameter description below.)

Simulated Reflection Complexity. This controls the size and number of “blobs” in the fake reflection. It works only when Reflection Type is Simulated; otherwise it has no effect. Increasing this:

- Increases the number of blobs.
- Decreases the average size of the blobs.



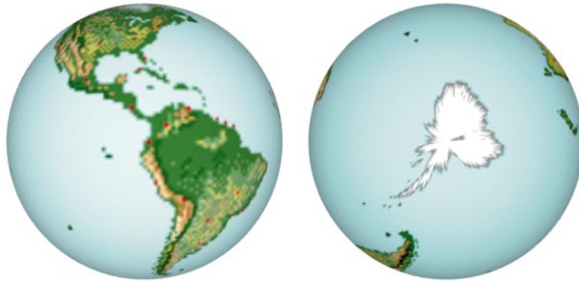
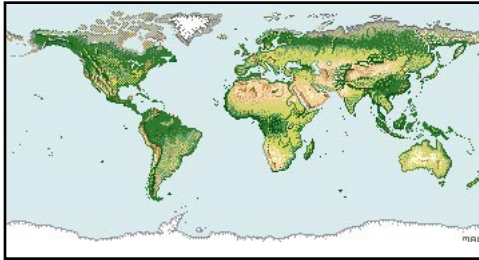
Reflection Picture. This allows you to select a color picture to get wrapped into a sphere and “shrink-wrapped” onto the surface to serve as the reflections. This is an image constructed something like a world map (which uses a Mercator projection), with distortions created so that when it gets wrapped into a sphere things look “normal.”

Danger, Will Robinson! When you use a Reflection picture you won’t see anything until you set Reflection Type to “Reflection Picture.” Also, if you use an Environment light to create a reflection, you’ll see that reflection as well as a reflection picture or simulated reflection.

- We’ve provided a few reflection pictures for you to use in your Instances. These will provide you with different sorts of scenes to reflect in your objects. You can find them all in the *reflectns* directory.

If you’re feeling adventurous, you can try your hand at making a reflection picture. This is an image that gets wrapped into a sphere in the same way as a Mercator projection of the world would. The main thing to remember is that the farther toward the top and bottom of your image you go, the more things will get scrunched. The very top line of an image’s pixels will be squeezed into a single pixel. It’s difficult to create an image and get really predictable results, but if you just need some custom blobs or shapes you’ll do just fine. Also, small shapes near the “equator” should look pretty reasonable, since they’ll suffer less from scrunching. Below you can see what happens at the top and bottom of a picture when it gets wrapped into a sphere — the view from the equator is what you’d expect, but notice what happens to Antarctica:





- The process for making any picture is described in "Getting a picture into an Instance (making labels)" in this chapter.

Environments and other reflections

Remember, you can also get reflections by using a TV Screen or Environment Look in the Lights window. These Looks create reflections that will appear on every object in the scene that's shiny enough to

reflect. Simulated reflections and reflection pictures, on the other hand, appear only on the objects that use them.

Approximating a new surface

If you need to create an altogether new surface, you have two choices, each of which has its pros and cons:

1) Use an Instance with a picture (scan in a photo, create an image in a paint program).

Pros

Great realism from scanned images

If you can paint, you can get just what you want

Cons

Limited to what you can paint or scan

Can look bad if scaled up

Always have to worry about correct wrapping

Lighting is already present in the image

Have to edit image to change the contents

Can use lots of disk space

2) Find an Instance that already shares some key characteristics that you can vary to approximate the new surface.

Pros

Control over a Look's characteristics (including colors!)

Responds to lights in scene

Scales correctly

May animate

Cons

May not be exactly what you had in mind

May not have the "right" parameters

May have to worry about correct wrapping

If you use a picture in an Instance, the process is relatively straightforward (see the section on "Getting a picture into an Instance (making labels)" in this chapter).

If you start with an existing Look, you can't escape the limitations of the parameters of the Looks you have on hand. The idea is to know your Looks and their parameters well enough to know which one shares the key characteristics of the surface you want to create. When you're experimenting with Looks, play with the parameters, explore the extremes: what's the effect of turning a wood's swirl all the way up, of increasing a glass's Metalness, of making a tiny checkerboard of white and off-white, etc.?

Get the idea? You may have to do some creative parameter tweaking to get close to what you have in mind. But remember, it just may be the case that "you can't get there from here."

RenderMan Expert parameter information for the adventurous

Ok, here's the deal. If you're reading this, we're assuming you're a bona fide hard-core computer graphics type. You've been warned...

Looks and "shaders"

A Look is actually a collection of text, bitmaps, other stuff, and a thing called a shader. The shader is the essence of a Look, the *sine qua non* for any image.

A shader is a small program that creates a synthetic surface. More accurately, it is in some ways a mathematical simulation of how a particular group of surface characteristics affects light. In the real world we usually see only light reflected from objects' surfaces. Naturally, this light is affected by the angle of the surface, the position of the light, and the position of the viewer. But what really makes a surface look the way it does are properties unique to the surface, for example its patterns of color and large- and small-scale bumps. It is aspects like these that are represented and manipulated in a shader by parameters.



Adjusting the whole set of available parameters allows literally millions of surface appearances.

The RenderMan Expert parameters

In RenderMan, the light you see coming from an object has six components:

- Ambient
- Diffuse
- Highlight Strength
- Highlight Spread
- Reflection Strength
- Reflection Spread

These are mixed together so as to mimic the reflection characteristics of a particular surface, each component contributing some percentage of the whole. (They often, but not always, add up to 100%.) These percentages are set initially in one of two ways:

- by the shader, internally;
- by higher-level controls — (Shininess and Transparency, for example)

When you move a slider for one of the six components, what you're really doing is changing the percentage of that component's contribution. In fact,

the preset percentage is multiplied by the number you see in the parameter's info window. It is most useful to think of these controls as fine-tuning values that are set elsewhere.

Ambient light in a scene gets used for one thing only: to control a surface's brightness uniformly in all the light and dark areas. The Ambient parameter multiplies the amount of *ambient* light (only!) that gets used to lighten the surface. Increasing ambient when there are 100 lights in your scene, but no ambient light, will have no effect!

Any other light can get used for two things:

- to get spread out (diffused) over the parts of a surface on which the light falls. The light gets used in proportion to the angle at which the light is striking it. Areas of the surface on which light is falling directly will be brighter than areas where the light is striking obliquely or not at all.
- to get concentrated in areas of a surface that are at a particular angle to the viewer, creating highlights.

The Diffuse parameter controls how much of the light gets spread around. The Highlight Strength parameter controls how much of the light gets used to form highlights. It often makes sense to vary these two parameters oppositely from each other, increasing one while decreasing the other. This way, when you make a surface "duller" the highlights diminish as

well. However, since this is computer graphics and not the real world you can effectively make a surface duller and shinier at the same time!

The values multiplied by the sliders basically specify a percentage. This is the contribution made by that component to the total light. The sum of all the percentages can be more than 100, which simply intensifies the effects of the light. If the sum is 0, you can have a thousand lights in your scene, but the surface won't respond to them, and may look like a black hole.

Ambient

This multiplies the contribution of the component that controls the overall brightness of a surface.

Ambient light is light that is “everywhere” in a scene, light that doesn't come from a single direction. It is all the light reflected and rereflected around the scene by all the various objects. You already know about this kind of light: it's the reason you don't need to turn on any lights in your house on a sunny day, even though the sun may not be shining directly into your room. There's enough ambient light to light the whole room adequately.

The Ambient parameter controls how much of this light the surface responds to. If you think of this as a percentage, 0 would mean the surface “sees” none

of the ambient light, 50 would mean the surface sees half of it, and so on.

If there is ambient light in a scene, increasing this number:

- makes the surface uniformly brighter;
- tends to wash the surface out somewhat — makes the colors “flatter” with less contrast.

If there is no ambient light in a scene, this parameter will have no effect!

Diffuse

This multiplies the contribution of the of the component that controls the dull, unpolished part of the surface reflection.

Diffusely-reflecting surfaces are ones like chalk, racquetballs, or flat paint. The light that hits a spot on a diffusely-reflecting surface gets scattered in many directions, instead of bouncing off in a single, predictable direction, the way a billiard ball bounces (like light reflecting off a mirror or a shiny polished surface). This has the effect of “spreading the light around” the surface.

This parameter determines how much of the available light gets used for spreading around on a surface. If the Diffuse and Specular parameters are both



present in a Look, usually as the value of Diffuse increases, Specular should decrease proportionally — things aren't usually shiny and dull at the same time.

Increasing this number:

- makes the surface brighter;
- “spreads the light around” more.

Highlight Strength

This multiplies the contribution of the component that controls the brightness of the highlights you see on shiny surfaces caused by light sources (not by simulated reflections or pictures).

Shiny surfaces look the way they do because the light hitting them gets bounced off at a very consistent angle.

This parameter determines how much of the available light gets used for creating highlights on a surface. If the Diffuse and Specular parameters are both present in a Look, usually as the value of Highlight Strength increases, Diffuse decreases proportionally — things aren't usually shiny and dull at the same time. Use the Highlight Spread parameter to control the size of the highlights.

Increasing this number:

- increases the brightness of highlights only.

Highlight Spread

This multiplies the contribution of the component that controls the size of the specular highlight. Naturally, if Highlight Strength is 0, this parameter will have no effect.

Increasing this number:

- increases the size of highlights, spreading them out across a surface,
- makes highlights' edges less sharp, more diffuse.

Reflection Strength

This multiplies the contribution of the component that controls the brightness of the reflections. This is a parameter that you'll just have to play with until the reflection looks the way you want.

Increasing this number:

- increases the visibility of reflections.

Reflection Spread

This multiplies the contribution of the component that controls the sharpness of the reflections.

Naturally, if Reflection Strength is 0, this parameter will have no effect.

Increasing this number:

- makes reflections' edges less sharp, more diffuse.

Displacement

When you use a Relief to bump a surface, you have two choices. You can actually change the shape of the surface (Displacement On), or you can use a trick of shading to make the surface just *seem* like it changes (Displacement Off). In the latter case, the surface remains flat, but areas are rendered darker or lighter to simulate changes in height. This can be very useful for prevent cracking problems at the edges of letters caused by too much displacement. Calculating new positions for points on the surface slows down rendering more than the simple shading calculations, so you might choose which method to use based on how obvious the bumping will be in your image.

For example, if you're bumping stucco on a back wall, your image might look perfectly fine if the stucco doesn't actually stick out from the wall that millimeter, but just *looks* like it does. On the other hand, if you have some bumps that need to cast shadows, be sure to turn Displacement on!

Turning this on:

- allows surfaces to change shape according to the Relief used.

Antialias

This makes surface patterns blurrier, reducing any jaggy, stair-step (aliasing) effects, if any.

Under some circumstances, some parts of a surface pattern may exhibit "aliasing." When this happens, any lines or patterns that aren't perfectly horizontal or vertical may have jaggy stair-steps at the edges. This parameter blurs the surface slightly (or lots) to make the aliasing less noticeable.

Note: Before you use this, you should try rendering with Smooth Shading turned on (in the dialog available by selecting Custom Setup from the Render menu).

Increasing this number:

- blurs the surface more.





Image: Joy Gipson

Font: Gill Sans Bold Condensed

Build Method: Extrude, custom (flat) bevel

Effects: Wall and Floor

Looks: ECShiny Sunhills; Matte, brown

Lights: #4 25%, #6 20%, #16 100%
spotlight with shadow, moved down and out

Floor made into a gradient in Photoshop