Subsurface Scattering:
Skin shaders for Poser 9/Poser Pro 2012

This tutorial explains some of the pioneering techniques developed by Bagginsbill for using subsurface scattering (SSS) to set up effective skin shaders in P9/PP2012. Because SSS was introduced with this generation of Poser, it is not suitable for earlier versions.

I've intentionally pitched this tutorial at beginner/intermediate Poser users who may have little or no experience of working in the material room.

Let me stress again: these techniques were developed by Bagginsbill, not me. I'm merely trying to explain them for people who may not be comfortable in the material room, or who have difficulty following BB's posts across several Poser forums. Mistakes, however, are mine, not BB's.

General principles

SSS is a realism tool. If you're interested in increasing realism in your renders, setting it up and using it effectively is a must. If you're interested in illustrative renders it may work for some images but not for others.

If you're going to use SSS effectively, you need to learn how to set it up yourself, because very few commercially available skin materials will work effectively out of the box. For a start, SSS is new and

not many vendors use it yet. And as they come onto the market, be aware that vendors try to maximize the appeal of their products by appealing to a wide range of users – that's just good business sense – but that may mean the shaders they're using aren't optimized for you.

And if you've used Poser for a while, you probably have older textures in your runtimes which you can bring to new life by learning how to set up the shaders.

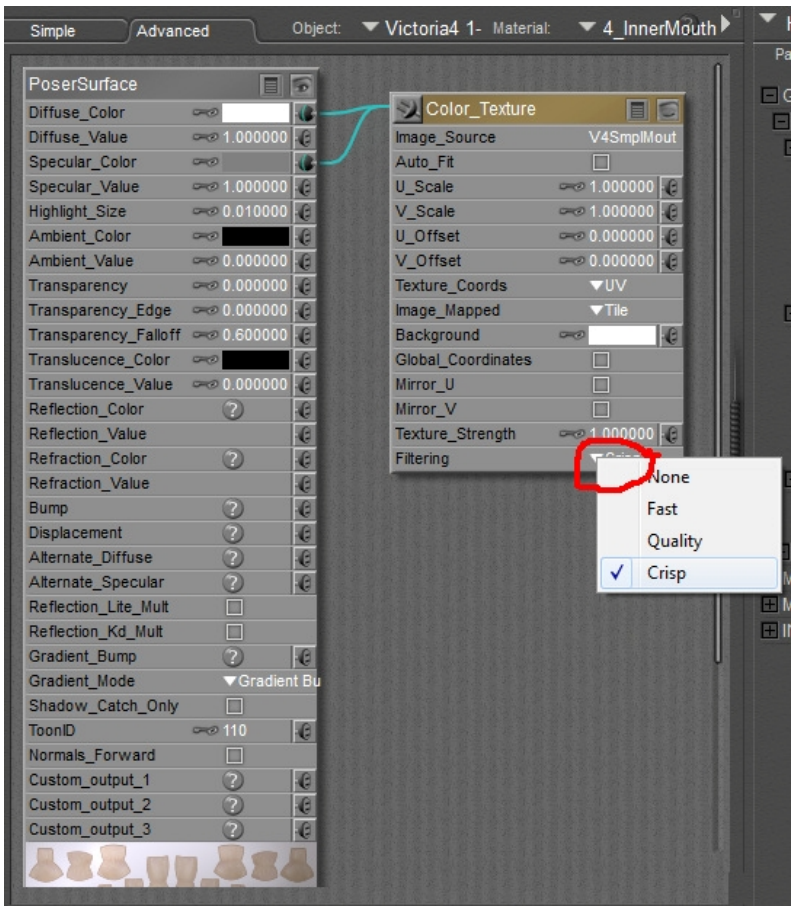
Fortunately, it isn't hard. Poser's new scatter nodes simplify the process of making attractive, effective skin. The incredibly complex shader systems of Poser 6, 7 & 8 may not disappear entirely, but they aren't necessary any more.

Lighting and textures

To render SSS you must also use indirect lighting (IDL). If you don't use IDL the SSS will automatically disable. I generally use just one light for this kind of work – a white infinite or spot light.

SSS is most effective when used with raytraced lighting. You can increase shadow map size and add some shadow blur if you like. Using depth-mapped shadows may produce unexpected effects such as nostril glow.

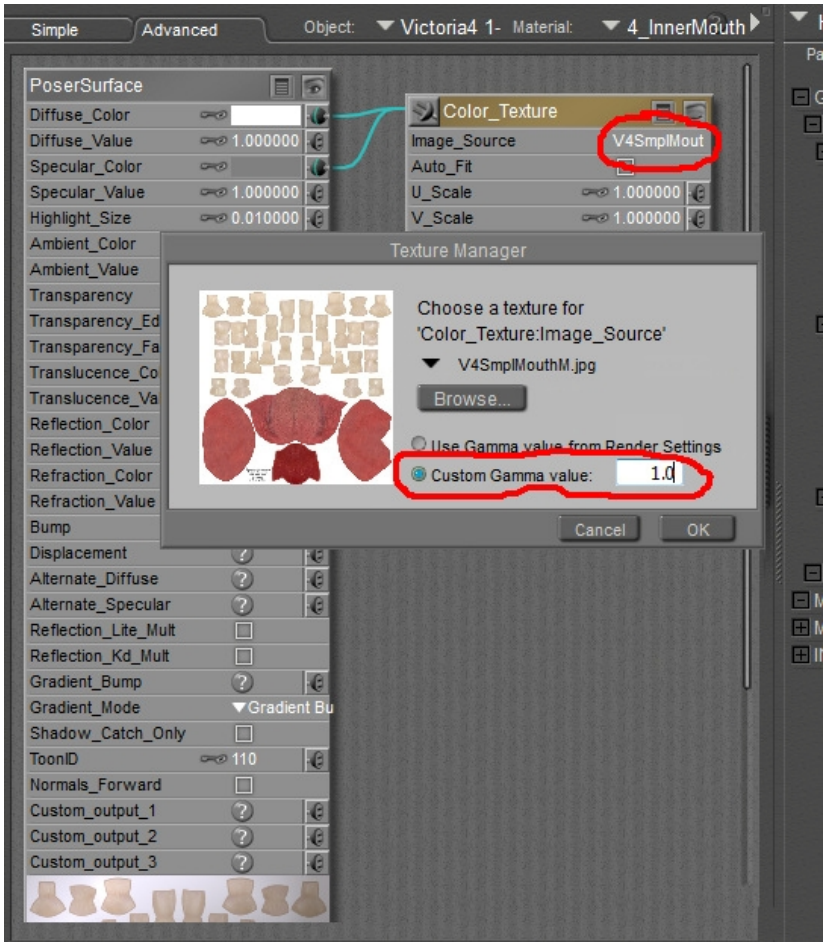
For close-ups, you'll get the best effects if you change texture filtering of all image-based textures to Crisp. That includes the diffuse map, bump maps, transparency maps – everything. DO it on clothes and close-up props as well. The only textures you don't want to do this with is transmapped hair (try it and see: transmapped hair looks rubbish with crisp filtering).



Gamma correction

If you're using Poser Pro 2012, you should use gamma correction at 2.2 for diffuse maps. You can either set this up for each material through the material room, or for the scene as a whole in render settings. I recommend setting it up through render settings; it's easier and quicker. If you apply it to individual materials you'll have to set it up for every material in your scene. Gamma correction is an all-or-nothing operation: do it for everything or do it for nothing.

However, greyscale maps such as bump, displacement or specular maps should NOT be gamma corrected. Set the custom gamma value of greyscale maps to 1.0 through the material room: click on the image name, select custom gamma value and type 1.0 in the box. (Yes, in the image below, I'm demonstrating on a colour image. I'm being lazy. Sue me.)



The reason for this is because gamma adjusts luminescence in a non-linear fashion (unlike Brightness, which adjusts it linearly). In the simplest terms, it makes dark areas lighter. Using it on bump, displacement, etc. will affect the depth of the bump in an undesired way.

If you're using Poser 9, you have four options. You can either ignore gamma altogether, you can compensate for the lack of gamma by adding extra diffuse-only lights to illuminate shadowed areas, you can manually create the gamma equations through maths nodes (if you can do this, you're already an advanced user, so why are you reading this tutorial?) or you can gamma-correct your

render in postwork, using an image editor such as PaintShop Pro or Photoshop.

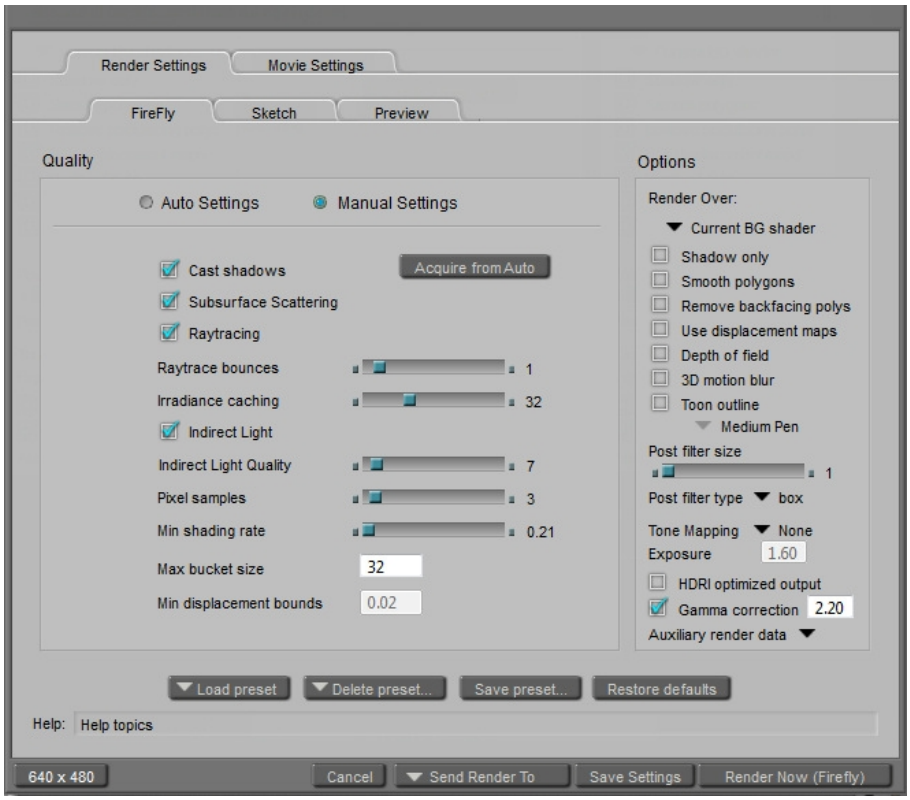
I highly recommend doing the gamma correction in postwork. Ignoring gamma will not make nice images. Using lights to compensate is what we used to do before gamma was introduced with Poser Pro; it requires a high degree of lighting skill to do properly, and it is an illustrative rather than a realism technique. And manually coding the gamma equations is probably not an option for you anyway (I won't pretend: I can't do it. If you want to try, you're on your own.)

Render settings

When I first started trying to improve my renders beyond the plug-and-play level, I was convinced there was some secret to render settings, and that if I found the secret everything I rendered would somehow look more professional. I cranked settings up as high as they would go. I spent hours rendering simple images. Nothing worked.

That's because there isn't a secret to render settings. The secret to great renders is in the material room and in the lighting you use. Get those right and you'll produce great renders with surprisingly low render settings.

You can, however, improve your render efficiency if you understand something about what the settings do. Don't increase them unless you need to – it just increases render time.



These are my default/preview settings for realism-based renders. Depending on the image, I sometimes even use these for a final render.

Raytrace Bounces: Keep these as low as you can for efficiency. You can reduce this to 0 if you have nothing reflective in your scene. Reflection requires 1-2 bounces. A reflection of a reflection (eyes

reflected in a mirror, for example) requires 2-3 bounces. Translucency or refraction (realistic water) requires a minimum of three bounces.

Irradiance Caching: This dictates how much the render engine estimates light, and how much it calculates it accurately. Lower values are faster but less accurate. Keep it as low as you can. I generally find a value of c. 30 is fine for previews. For final images I may increase it to around 50. For showcase images I may go up to 90, but that will really slow the render down, especially if combined with a high number of raytrace bounces.

Indirect Lighting Quality: Does what it says on the tin. I rarely increase this above 7-10 if I have even a single real Poser light in the scene. If I'm relying on light emitters I may move it up to around 30.

Pixel Samples: This controls the radius of pixels which affect each other in the render engine's calculations. For most images, 3 is fine for a preview and 5 for a final render, but if you're using depth of field you must increase it – I often use 10 or 12 for a final render using DoF. If using DoF leads to jaggy backgrounds rather than blurred ones , increase the pixel samples.

Minimum Shading Rate: Determines how much a polygon will be subdivided by the renderer. A value of 1 means no subdivision. 0.95 is fine for a preview or distance shot. For higher-quality renders you should lower it. I use 0.2 for final renders of close-ups. There's not much point in going below 0.2 except in very unusual circumstances.

Bucket Size: Controls the size of the pixel block Firefly renders at a time. Larger blocks need more computer resources and hence more time per block, though larger blocks may decrease overall render time. How many blocks Firefly will render at once depends on the number of processors in your computer. I'm currently using 4 processors and usually use a bucket size of 32; for many images 64 would probably be more efficient, but I rarely bother changing it – 32 is quick enough when the rest of my render settings are correct.

Scene set-up

IDL works best when it has something to bounce off. If you're not loading a scene around your figure, you should load some kind of skydome and a backdrop to hide the skydome from your render. I usually use the IDL Dome and IDL Cove from Colm's IDL Studio (available from Runtime DNA). You can also use Bagginsbill's free Environment Dome, or the geosphere and studio background which ships with Poser (you'll find them in the Props/Primitives folder of your main Poser runtime).

Display Units

Some settings on the PoserSurface node (the main block you plug everything else into) are based on your display units – notably the bump and displacement values. This tutorial uses inches; you will need to adjust these settings if you use a different display unit. Check in your preferences: Crtl+K->Interface tab->Units. If you set up nodes in inches and later change to a new display unit, Poser will remember and change the settings for you; you don't need to set up new shaders.

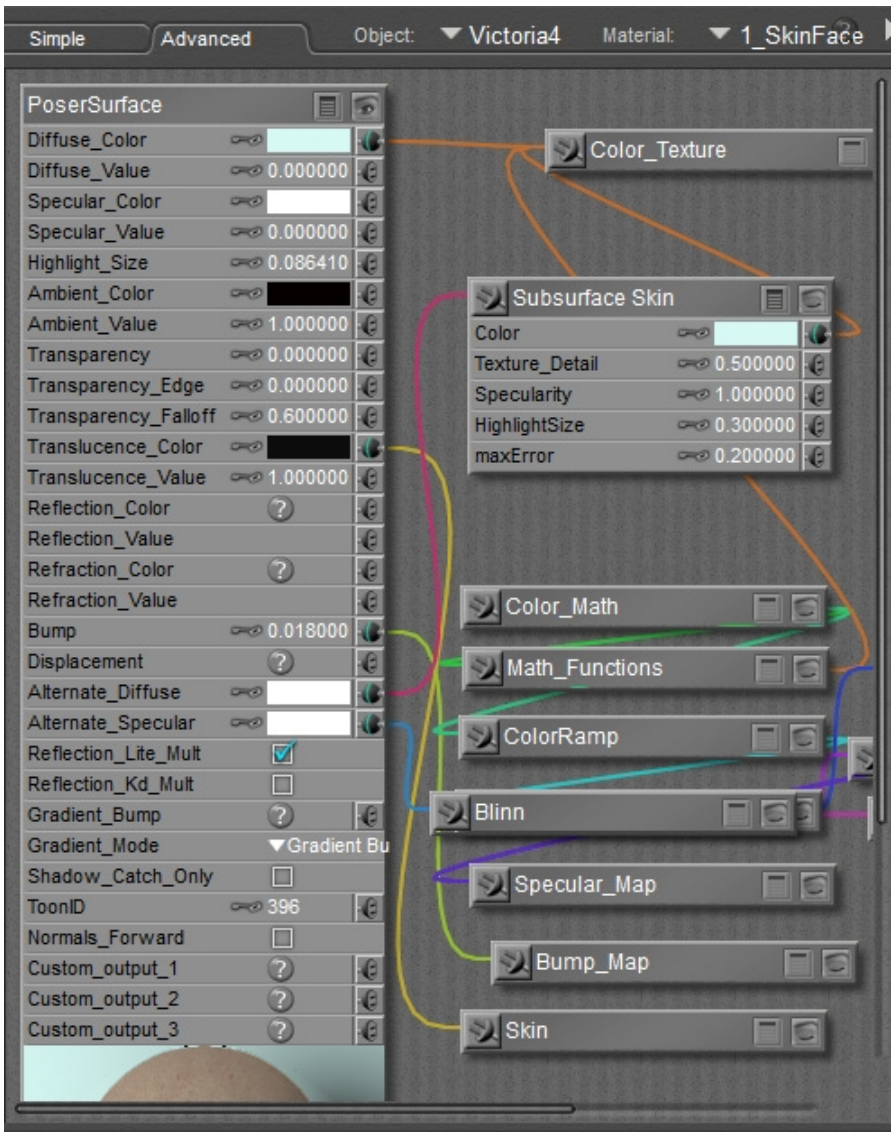
Ho-kay. We know what we're doing with our lighting, scenes, gamma and render settings. We know what display units we're using. Are we all sitting comfortably? Then we'll begin.

The Subsurface Skin node

The easiest way of setting up SSS on your characters' skin is by using the SSS wacro in the material room. It automatically adds the Subsurface Skin node to the selected material. You'll need to do this for all the skin materials in your figure.

The wacro does two things: it inserts the subsurface skin node between the texture map and the alternate diffuse channel, and it turns the standard diffuse and specular nodes to 0.

Here's what the wacro nodes look like in the material room:



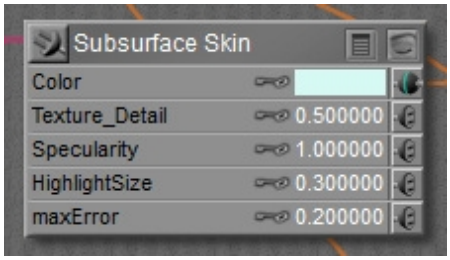
Note: I'm using the V4 Amy textures from DAZ3D here. I've hidden the Amy texture settings, as they're a commercial shader. See how the Diffuse_Value and the Specular_Value are set to 0. This means they do not contribute the final render, which is based entirely on the Alternate_Diffuse and Alternate_Specular settings. The alternate channels allow the use of complex nodes (such as subsurface scattering or Blinn specular) which do not work in the standard diffuse and specular channels.

OK, that's the set-up. Let's see how they render:

V4 Amy without SSS

V4 Amy with wacro SSS

The difference is marginal. With SSS the skin is a little softer and a little paler. It's noticeable but extremely subtle. And both sets are rather dull and lifeless.



Now let's look at the Subsurface Skin node and see what we can adjust.

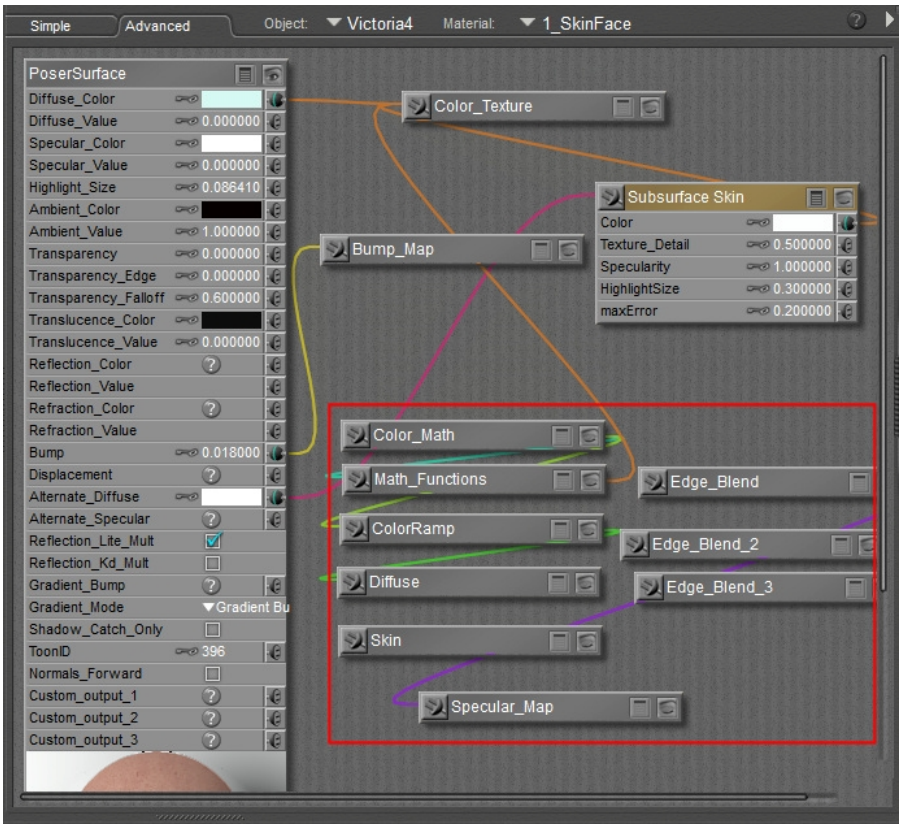
The first thing I notice is that the colour is set to a pale blue. That means any light I shine on the material will pick up a slightly blue cast. That's an illustrative effect, not a reality one, so I'm going to make it white.

Next, I see the node includes settings for specularity and highlight size. But Amy already has a Blinn specular built into her shader. I'll disable the Blinn (right click it and delete). I know – because I've read up on it – that the Subsurface Skin node includes its own specular as part of the render. We'll leave the Alternate_Specular channel empty.

The other thing I want to change is the Translucence_Color and Translucence_Value channel. Since subsurface scattering is a form of translucence, I'm going to disconnect the node and turn the value to 0.

The results are definitely better.

However, now we've disconnected the old shader's specular node as well as the diffuse, we're left with a bunch of nodes which aren't doing anything.



The nodes I've highlighted within the box are no longer connected to anything on the PoserSurface node. They can be deleted.

In addition there is a node called skin which is plugged into the Translucence_Color channel, and the Translucence_Value is set to 1.0. Not only is this unnecessary (SSS is a form of translucency), it may actually be interfering with the shader's output. We should delete the node and set the Translucence_Value to 0.

You'll have to do this for each and every skin material. Yes, it's a pain, but it does help you learn your way around the material zones of the figures you use, and around the material room.

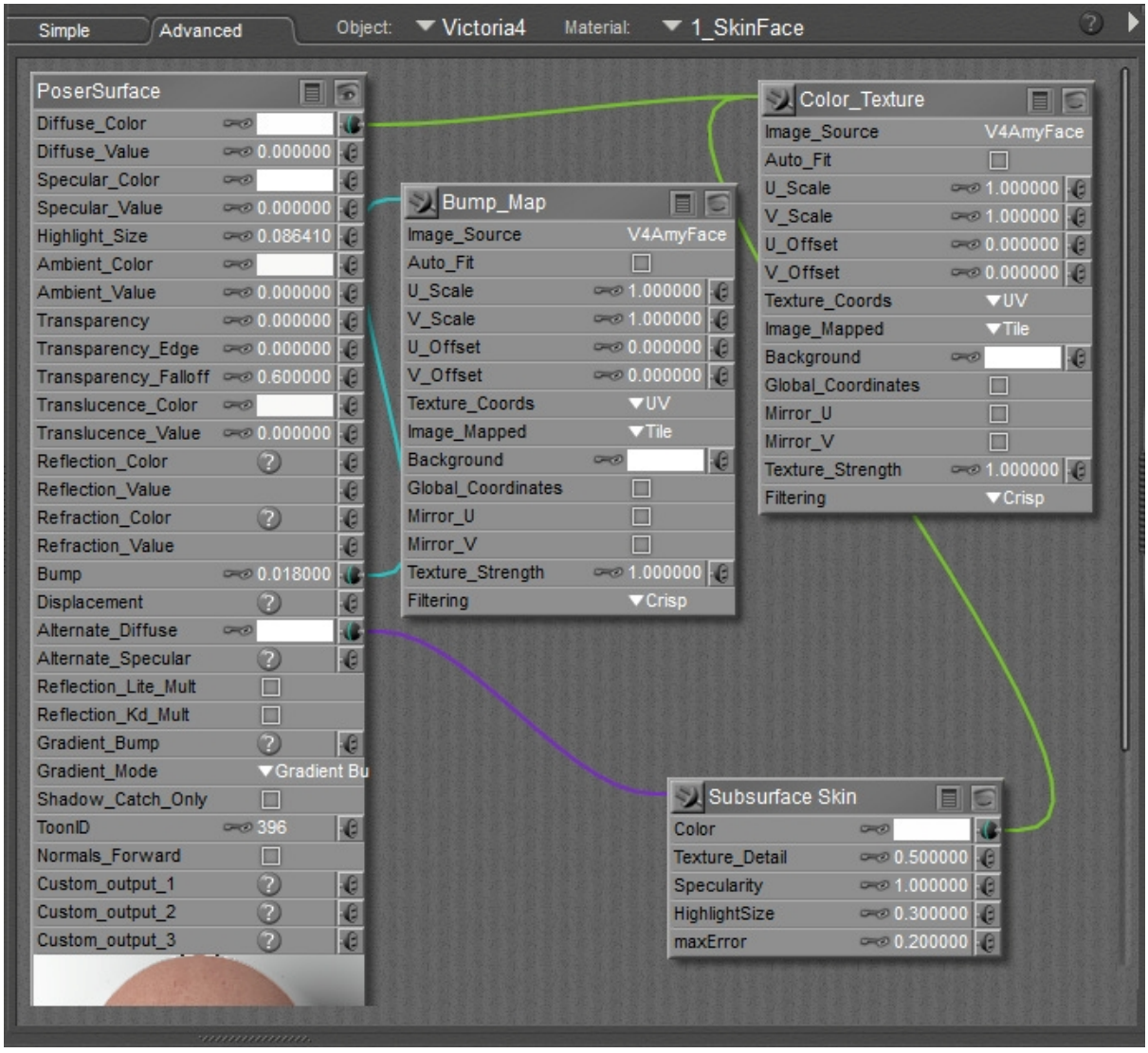
Once we've done that, there isn't a great deal of the original shader left – just the Color_Texture node and the Bump_Map node, and both of those are texture maps. The only procedural node we have is the Subsurface Skin node.

You'll also notice that the wacro didn't even do half the job we needed it to. That's an important lesson: if you want something done properly, do it yourself. There is no "make art" button.

Right, let's clean this shader up so that we have only the nodes we need. Select all the nodes you don't need (CRTL+left click them in Windows) and delete them (Right click>Delete in Windows).

While we're cleaning up, let's uncheck the Reflection_Lite_Mult box. This function makes reflections darker when they're in shadows. Our skin isn't reflecting anything, so it's unnecessary, and the way the function operates doesn't correspond to real-world light. It's therefore disliked by renderers looking to increase the realism of their images.

You're left with something like this:



Pretty simple, isn't it? All we've kept of the original shader is the texture map and the bump map. All of the other nodes have been replaced by one Subsurface Skin node.

So why were all those other nodes there if they aren't necessary? Well, once upon a time they were necessary. They were designed to make textures look better without IDL and without SSS (and without gamma correction). In other words, they're designed to look good in Poser 6 and Poser 7.

And we'll continue to see more like them. Not every Poser user keeps up to date with the latest versions. There are many out there still using Poser 6. Vendors want to maximise their market, and making Poser 6-compatible shaders does that. At the time of writing there is, to my knowledge, only one commercially available texture which uses SSS, and a couple of others shipping free with Antonia Weight-Mapped and Victoria 4 Weight-Mapped.

Fortunately, all you really need is a decent texture map. You can build your own shaders.

For many users, this Subsurface Skin shader will be enough. But not for others. We can get better.

What's wrong with Subsurface Skin?

For many purposes, absolutely nothing. It's an all-in-one solution – which is exactly what it was designed to be. It's very simple and gives results that are perfectly acceptable to many users.

Other users are more demanding.

The Subsurface Skin node is a black box. Aside from the limited values covering its specular function, we can't play with any of its parameters. That means it's impossible to adjust the amount of subsurface scattering; you get what you're given.

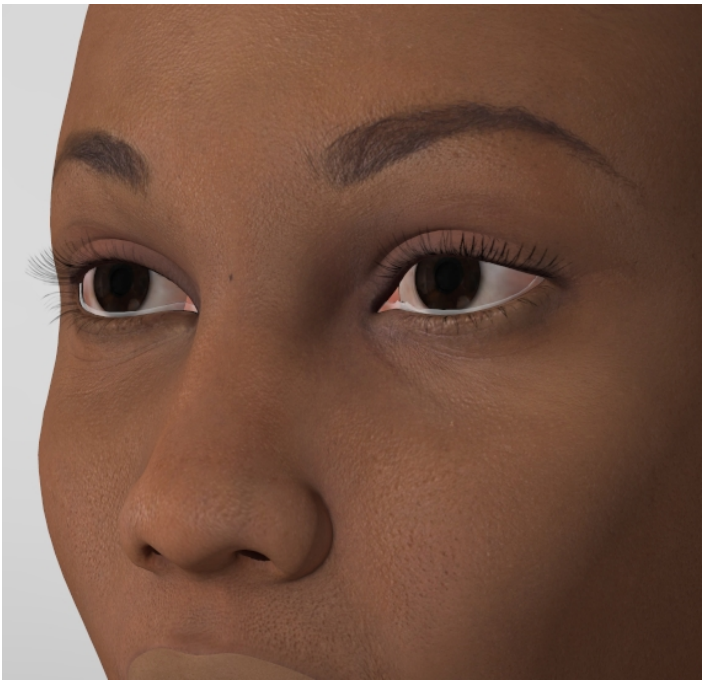
Fortunately, the nice people at SmithMicro have provided a couple of alternatives. The Scatter node has several presets for subsurface, including two for skin, and has more parameters we can play with. The Custom Scatter node has no presets, but offers the most options to adjust parameters. At the time of writing, I'm not aware of any skin shaders which require the Custom Scatter node; all shaders which eschew the Subsurface Skin node rely on the scatter node.

And some dislike the built-in specular function, which is built around a KS MicroFacet node. The only way to use a different specular algorithm is to avoid the Subsurface Skin node.

Bagginsbill ran a series of tests on the Runtime DNA forums comparing the use of the Subsurface Skin node and the Scatter node combined with a Blinn specular node: most people who responded preferred the results of the Scatter + Blinn. That's what we'll look at next. But before we do, let's consider what makes a good texture.

Interlude: good texture maps

A good texture map for realism renders has no burnt-in specular and has no hair painted on the skin. Unfortunately, most textures created from photographs have both these flaws. It seems counterintuitive, but some of the best textures for realism are handpainted.



Burnt-in specular is bad because it doesn't react to your lighting. Specular is shininess – light bouncing off a surface. It should change depending on the colour and angle of the light and the physical properties of the surface.

Check your textures by opening them in an image browser. If you can see white areas, you have burnt-in specular. It's most noticeable in darker skin.

Licence terms forbid me from showing a flat texture image, but here is V4 Marie in a diffuse-only render. With a diffuse-only light there should be no shinyness

at all, yet there is, most notably around the eyebrows, under the eyes, on the cheeks and on the tip of the nose. We have built-in specular.

We also have painted-on eyebrows. The problem with painted-on hair is that hair and skin react to light differently. It's most noticeable with specularity – the shine shouldn't move seamlessly from skin over hair and back to skin, but applying a specular node to the Skin_Face zone will do just that.

Properly speaking, the best way of correcting these issues is to open up the texture in an image editor such as Photoshop or PaintShop Pro and remove them. However, this takes a high degree of skill and a great deal of time – so much time, that many 3D professionals consider it not worth the effort and prefer to paint textures from scratch.

Bagginsbill has developed techniques to counter both these flaws within Poser itself. They aren't perfect, but they do improve matters considerably. We'll look at both these later.

Before we move on, I'd like to make it clear that I do not dislike the V4 Marie texture I've used as an example. Just the opposite – it's one of my favourite textures. The issues of burnt-in specular and painted hair are extremely common in Poser textures. For illustrative purposes they aren't especially problematic; only in realism renders does it really matter.

Bump maps

A common technique vendors use is to use the greyscale version of their colour texture as the bump map.

One method runs the texture through nodes to turn it grey and plugging it into the bump channel. That at least has the advantage of efficiency – we only have to load one texture map. Another common method is to turn the colour texture grey in an image editor, and load it separately into the bump channel.

The problem with these methods is that colours have little relation to the level of bump in a surface. Bagginsbill illustrates this by using M&Ms as an example: we have a coloured surface with a white M on it. If we make this grey and plug it into the bump channel, it will look as if the M is elevated from the surface, but it shouldn't be. M&Ms are smooth all over.

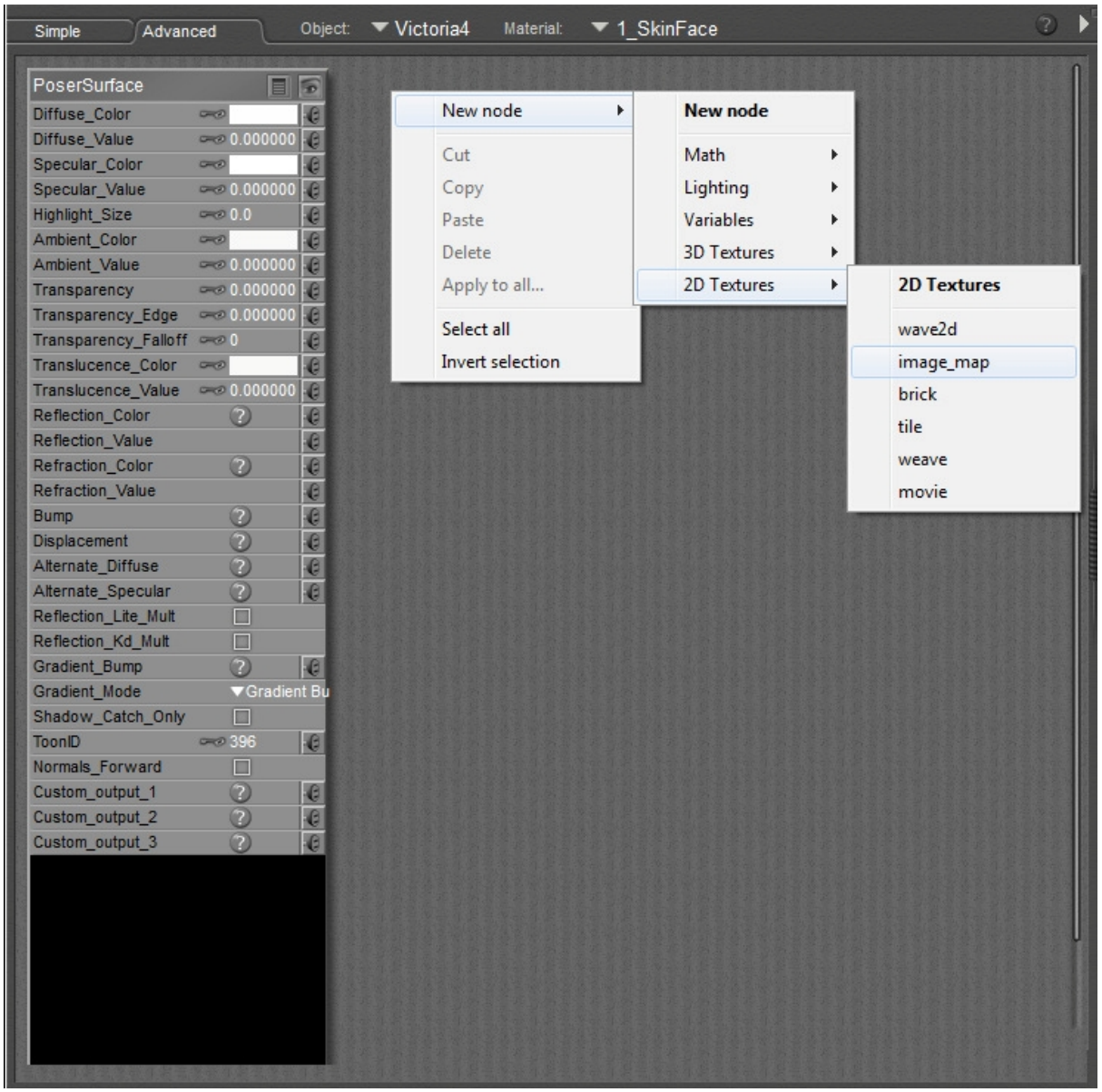
The ideal bump map would be a handpainted greyscale map. Unfortunately, no one's made one for Poser figures so far. Bagginsbill's solution is to create a procedural shader, which doesn't rely on a texture map at all.

Simple Scatter + Blinn

One of Bagginsbill's most famous skin shaders is called Scatter + Blinn, also known as the James shader. This is a simplified version.

We're going to build this one from scratch, so start with a blank PoserSurface node. Make sure all of the colour values are white and all of the values are set to zero. Uncheck the Reflection Lite Mult box.

Right-click a blank area and select New Node->2D Textures->Image Map and select the texture of your choice. In this case, I'm going to continue with V4 Amy.



Remember to adjust the texture filtering to Crisp. Connect the image node to the Diffuse_Color node. Leave the Diffuse_Value at 0. This will let the preview display the map, but the 0 value means this channel will play no part in the final render.

Now we'll set up the path to the Alternate_Diffuse channel.

Create three nodes: a Scatter node (New Node->Lighting->Special->Scatter) an hsv node (New Node->Math->hsv) and a Blender node (New Node->Math->Blender).

Change the Scatter node material to Skin 1. Change the colour of Input_2 on the Blender node to black. Uncheck Use Material Color. Change the Scale to 1.75.

Connect the Image Map node to the Color input of the Scatter node. Connect the Scatter node to the Color input of the hsv node. Connect the hsv node to Input_1 of the Blender node.

Before we go further, think about what each of these nodes is doing.

The Scatter node is our main driver for subsurface scattering. It has a number of parameters we can

adjust, but we'll leave them alone for now.

The hsv node allows us to adjust the colour and intensity of the texture map(hsv stands for Hue, Saturation, Value). With the Color parameter set to white and the other values set to 1.0 it isn't adjusting the texture map at all. It's there just in case we want to change them later (for example, you could make lips redder – or any other colour – to represent lipstick, turn skin green to represent an orc or goblin, or boost the intensity of the texture if it looks a little dull when rendered).

The Blender node mixes to paths into one. At the moment, we aren't mixing it with anything. We will later.

Now we can set up the bump shader. V4 Amy has a dedicated bump map but it's basically a greyscale version of her colour map. We'll use Bagginsbill's procedural bump instead.

First, create a Turbulence node (New Node->3D Textures->Turbulence). Lower the x,y,z scales to 0.05 and the Gain to 0.15. You can leave the Octaves, Bottom and Gain at their default values. Leave the Noise Type as Original.

Next create a Math_Functions node. Select Subract from the drop-down and set Value_1 to 1.0 and Value_2 to 0.5. Plug the Turbulence node into the Value_1 input. This takes the output of the Turbulence node and normalises it around 0; instead of the range of the turbulence going from 0 to 1, it goes from -0.5 to 0.5. In other words, the surface of the skin is the middle.

Plug the Maths_Function node into the Bump channel, and set the Bump value. If you're using inches as your display units, the bump value should be 0.02. If you're using feet, it should be 0.001667; if metres, use 0.000508, if centimetres use 0.0508 and if millimetres use 0.508. And if you're using Poser Native Units, it should be 0.000194.

If you subsequently change your display units, Poser will know and will automatically adjust the bump setting to suit. Setting the right bump value only matters when you create the shader.

We need to set up one more path: the specular.

Our main driver will be a Blinn specular node. Create it now (New Node->Lighting->Specular->Blinn).

However, we want it to apply only to the skin, not the painted-on eyebrows (or head hair or facial hair, for textures that have those painted on as well).

To do this, we use one of the neat little Bagginsbill tricks I mentioned earlier. BB realised that skin contains more red pigmentation than hair. If we can detect the areas with more red, we can separate the areas of skin from the areas of hair.

Fortunately, Poser has a node that does just that. It's called Component (Comp for short), and it has only one purpose: to detect either the Red, Blue or Green channels of a texture. If you want it to detect red, set the Component parameter to 0.0. To detect Blue, use 1.0. To detect Green, use 2.0.

Create your Comp node (New Node->Math->Component). Set the Component parameter to 0 (so it detects red), turn the Color parameter white, and connect the texture map you're using to the Color

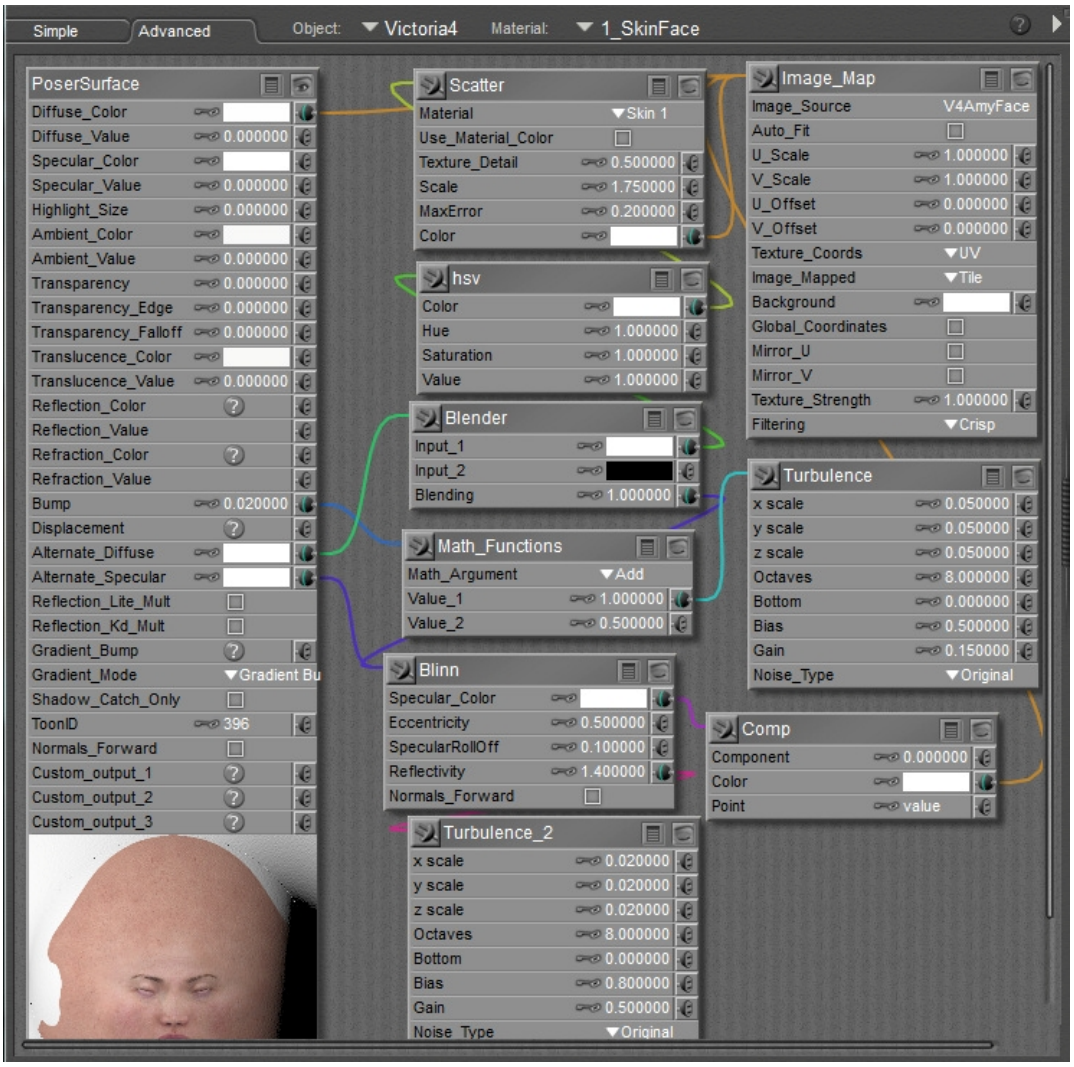
input. Now plug the Comp Node into the Specular_Color input of the Blinn node.

Now set the Blinn parameters: Eccentricity should be 0.25, SpecularRollOff 0.4 and Reflectivity 1.4.

We can increase the realism a little further by varying the level of the reflectivity because skin isn't uniform. We do this with another turbulence node. Create one (Turbulence is in 3D Textures, remember) and set its x,y,z scales to 0.02, its Bias to 0.8 and its Gain to 0.5. Now connect the turbulence node to the Reflectivity input of the Blinn node and connect the Blinn node to the Alternate_Specular input.

There's one more connection to make. Remember that Blender node we created along the diffuse path? We're going to plug the Blinn into the Blending input with a Value of 1.0. This suppresses the diffuse channel where there is specular, ensuring the total light reflected from the surface does not exceed the light falling on it. It's a classic Bagginsbill trick, designed to fool the Firefly render engine into operating more like real-world physics.

Your shader should now be looking something like this:



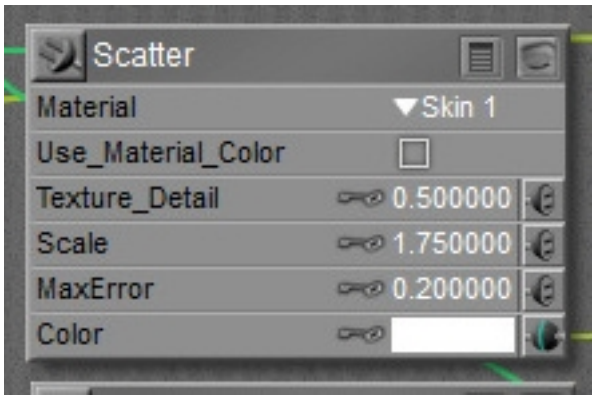
Now you've finished the shader, save it into your library as a single material. You can then apply it to the other skin materials of your figure, changing the texture map when necessary (the face, nostrils, lips and eye socket of V4 and M4 use the Face map, for instance, while the ears, back of the head, neck and nipples share the torso map. Arms and legs use the Limbs map.) This is much quicker than creating custom creating the nodes for each material zone.

Once you've applied the new shader to all the skin zones of your figure, you can save it as a material collection. Check only the skin materials when you're asked what to include – we haven't done anything with the eyes or inside the mouth.

Here's what this shader looks like rendered:

I think you'll agree these are pretty good results. I've only used the preview render settings I gave on page 4. Lighting is a single infinite light with raytraced shadows.

The scatter node



There are a number of settings we can play with. Experiment and see what works for you.

First, we can swap between the Skin 1 and Skin 2 presets.

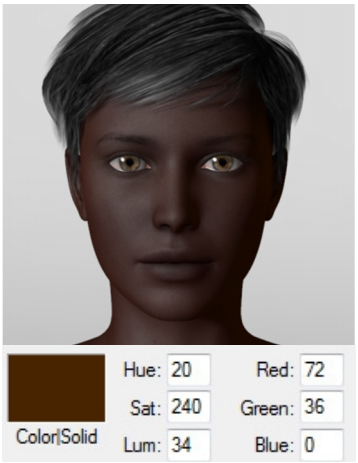
Checking the Use_Material_Color box imposes the preset's inbuilt colour on the shader.

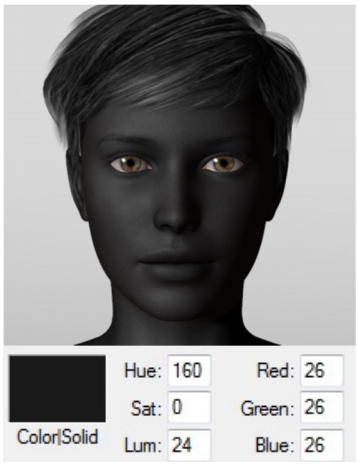
Texture_Detail controls how strongly the original texture is applied after the scattering effect is rendered. A higher value means more detail. Values run from 0.0 to 1.0. Consensus among posters to the RDNA and Renderosity forums suggests a value around 0.7 offers a good level of detail.

Scale controls the depth of the scattering effect. Lower values mean more scattering, resulting in a more wax-like skin, but run the risk of 'blueing' in areas where the mesh lies close to itself (such as the nose) or where there is underlying geometry (such as the inner mouth underneath the lips). The effect of underlying geometry is worse when that also has SSS effects. Experiments suggest this happens with values of 1.5 or below. For how to avoid this and still have waxy skin, see the Baggins Max Scatter Trick below. Note that younger skin has more translucence than adult skin, so if you want a youthful look, lowering the scale helps.

MaxError controls the quality of the scatter. Lower values mean better quality but slower rendering times. The default, 0.2, seems to work well for most people.

Color tints the incoming texture map. Small adjustments can make a significant difference in the final render. You'll mostly want it for fantasy effects, though a subtle touch may adjust a skin tone to reflect a suntan or ethnic skin.





The Baggins Max Scatter Trick

I mentioned earlier that lower Scale values in the Scatter node could create blueing. Let's have a look at that.

Here's what Scale 1.0 looks like:

The skin has a more wax-like appearance which is quite attractive, but the blueing is just starting to be visible around the nostrils and ears.

Both effects will get more pronounced if we reduce the Scale further.

Here's Scale 0.5:

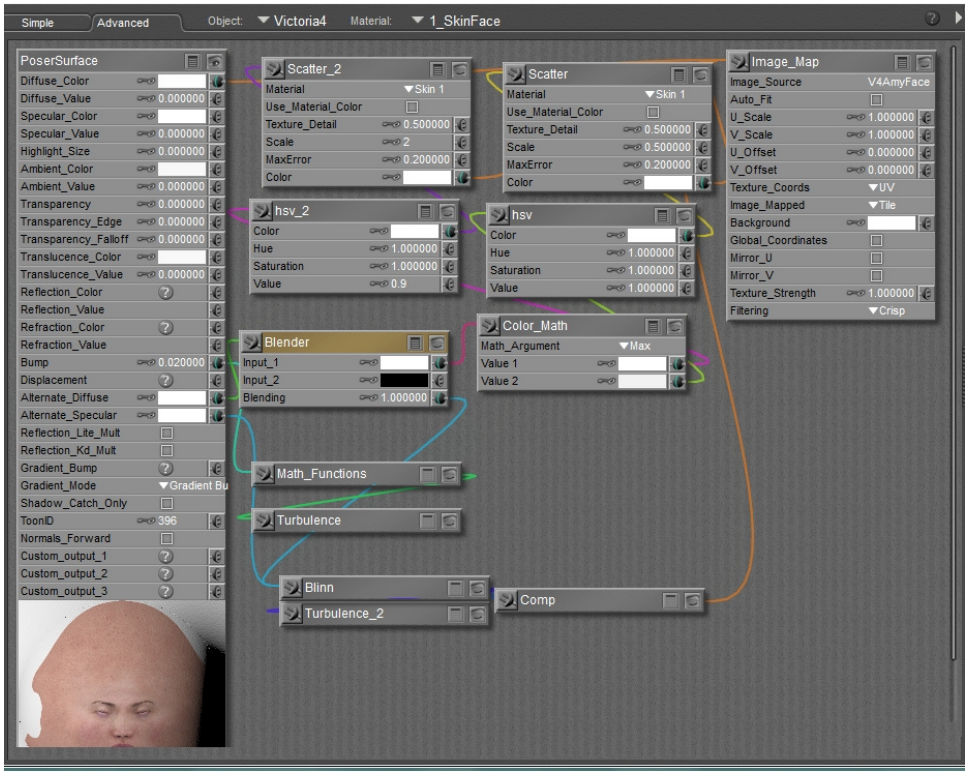
The blue cast is now highly visible as grubbiness around the mouth. The effect is very unattractive.

The strength of the artefact is affected by morphs which adjust the distance between the skin surface and the inner geometry, and by textures themselves.

So how do we increase the translucency without getting the blue cast? This is where the Baggins Max Scatter Trick comes in. Essentially, we run a duplicate scatter node, and all its inputs, use a higher scale on the second scatter node, run that node through an hsv node where we can adjust

the value, and run both scatter paths through a Color_Math: Max node. The Max node will select the most colourful of the two inputs at any given point.

Let's set this one up.



You may need to adjust the Value in the hsv_2 node to get the best results.

Here's how this set-up renders, compared with the simple Scatter + Blinn:

The Max Scatter, on the left, is a little paler (that's the effect of reducing the hsv_2 value), but it has details in the shadow that the Scatter + Blinn version lacks.