VinceBagna

VB SuperShaders

for Poser11 SuperFly

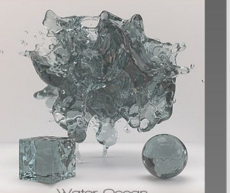
Volume 1

VB SuperShaders for Poser11 - V1

Glass Magic

Painted Wood 05

Glass Mirror


Water Ocean

Glass Broken

Plastic Translucent

Ice 02

Concrete Polished 03

Leaf

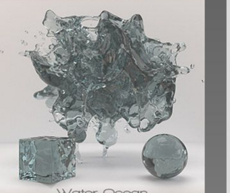
Plastic Scratched 02

Satin Bump

Glass Magic

Painted Wood 05

Glass Mirror


Water Ocean

Glass Broken

Plastic Translucent

Ice 02

Concrete Polished 03

Leaf

Plastic Scratched 02

Satin Bump

Introduction

First of all, I will try to explain a little bit how the new Poser render engine SuperFly works. Hey, don't be scared, it will be veerrryy simple, you'll see ;)

The last iteration of Poser has added (among other exciting features) a new render engine called **SuperFly**, and based upon a renderer called **Cycles**. Cycles is used as an add-on renderer in Blender. It's been a long time since the addition of the last render engine in Poser, FireFly, and it was really time to upgrade that.

You have perhaps already heard about this new hype everywhere, **PBR**, no? Keep calm, no secret name here, **PBR** is used for **P**hysically **B**ased **R**endering. What is this? Basically, it's a new way to treat lighting and surfaces by changing how light bounces off. In short, no need to trick now to make your renders look realistic. The physical behavior of the light and the shaders (energy conservation, Fresnel...) is already computed by the render engine, so the result is instantly realistic. There is way more to say about that, but it's the most important thing you should know. I said I was explaining in short, no? ;)

If you really want the full story about how PBR works, then you could go there to have more infos ---> Basic Theory of Physically Based Rendering.

Cycles is not a PBR, so SuperFly is not either... Wow! Don't look at me like that! I explained what SuperFly is not, so you can understand easier what it is. FireFly is not a PBR, it's better than that! It's a probabilistic ray-tracing engine that adds randomness when it traces the paths of rays of light at slightly different angles. For that, it uses BSDF shaders (Bidirectional Scattering Distribution Function) which determine in which direction the ray will bounce.

In one simple sentence: **you can have realistic results, and also unrealistic ones!** Isn't it great?!

You are probably beginning to think I've lost my mind, we could do unrealistic renders before SuperFly, and realistic ones too. But we were very limited, in a way or in another. Now it's supra easy to get realistic results, and not less easy to have unrealistic ones. By unrealistic I mean: we can get ARTISTIC results, If we could render with real light behavior ONLY, with shaders that work like they should do in real life ONLY (think metal, glass and all), it would be very depressing, right? If you want a vampire not to be seen in a mirror (everybody knows that real vampires are invisible in the mirrors, right?) but your render engine says that mirrors reflect everything, vampires included, you'll be forced to use post production to make your effect. Now SuperFly will make your mirror shader acts like real mirrors do indeed, BUT you can get around that and make your mirror not reflect something, or reflects it with

another color, or without shadow, or... excuse me, I'm getting excited, SuperFly is way more than a true PBR, it lets you stay Artists.

Finally my explanation has been longer than expected, but if you get till this point, it was not that terrible, huh?

I will try in the following pages to explain some of the shaders I made. They are real easy to understand, trust me. Basically, one node with several settings, labeled in a way to catch it fast! Some I created while playing with Poser 11 during the Beta phase or after, others I was inspired by the guys at BLENDER ARTISTS or BLENDER GURU or elsewhere, doing some changes and improving it while keeping it simple and intuitive. Seriously guys, try to play, with the SuperFly nodes or my shaders, mix them, make some experiments, and most important, **have fun**!! It's the best way to understand and to do great materials!

Now the best way to understand SuperFly is to use it. So let's go!

General Settings

You'll find several settings you can change in these shaders. They are globally the same you can find with the Cycles nodes in the Poser 11 material room.

Diffuse Color

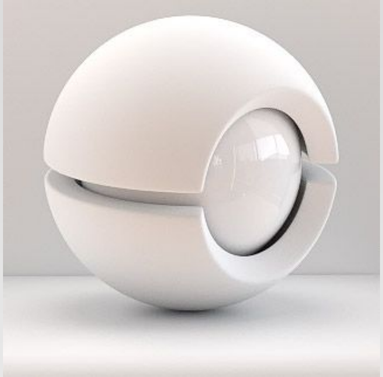
Controls the overall color of the shader. Generally an image map can be plugged here.

Glossy Color

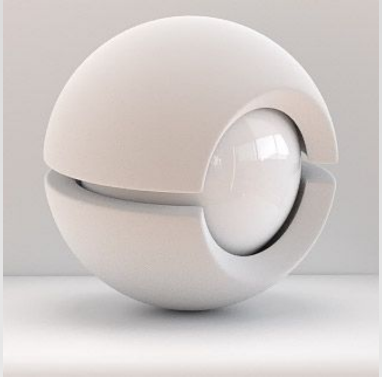
Controls the glossy/reflection color (in SuperFly it's the same). Generally white, can be colored to create an effect.

Roughness

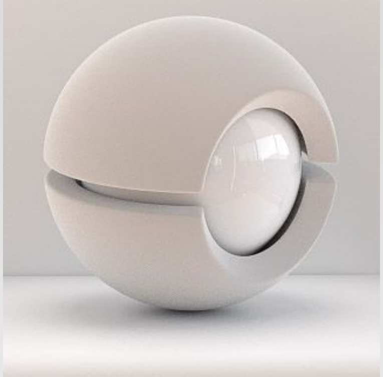
It can refer to the diffuse, reflection or transparency properties. Roughness can turn diffuse surfaces more light absorbing, reflections more blurry, and glass more foggy.



Diffuse Roughness 0

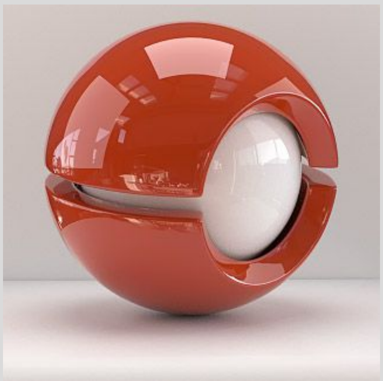


Diffuse Roughness 0,5

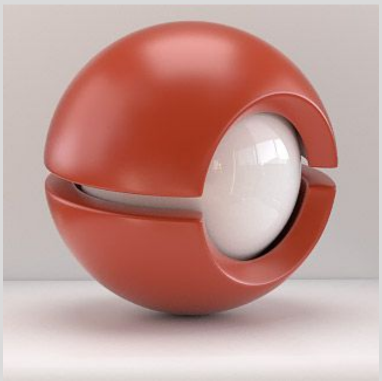


Diffuse Roughness 1

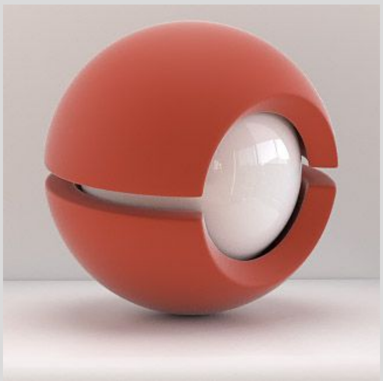
(also known as Oren-Nayar shading)



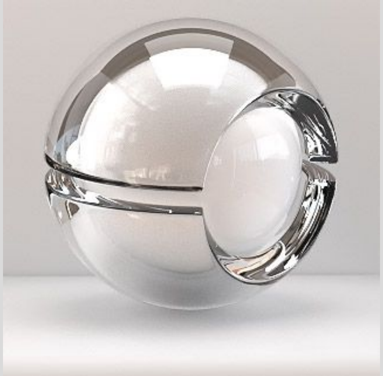
Glossy Roughness 0



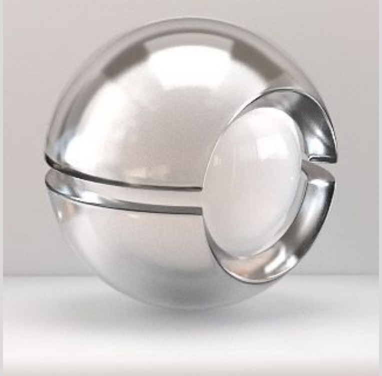
Glossy Roughness 0,1



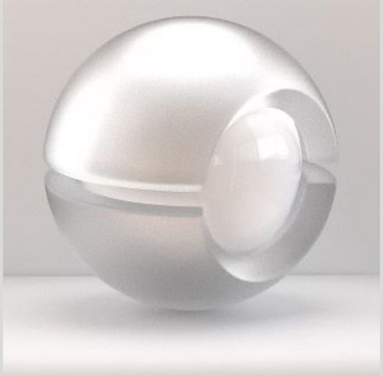
Glossy Roughness 0,5



Glass shader Roughness 0



Glass shader Roughness 0,02



Glass shader Roughness 0,2

Coat Color

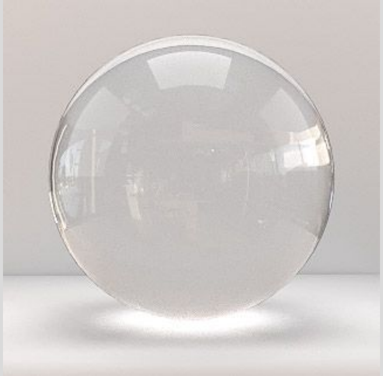
Controls the color of the extra layer of coating (i.e. secondary glossy). Can sometimes go with a Coat Roughness setting that works the same as Roughness.

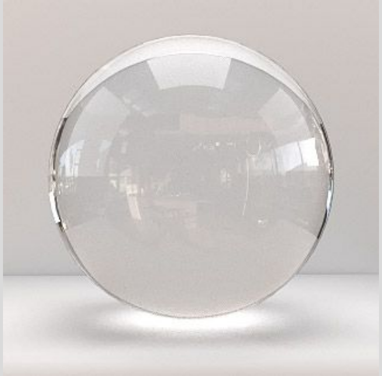
Brightness

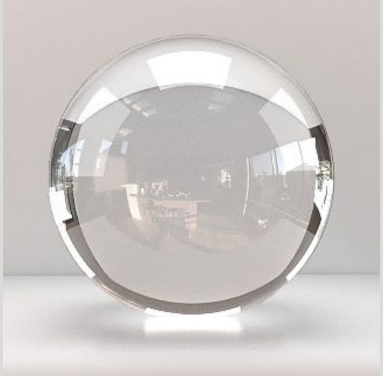
Controls how much glossy the shader is. Works like Specular_Value in FireFly root node.

IOR

Stands for Index Of Refraction. There are 2 kinds of IOR:
- If for Glass, it controls the amount of refraction to occur.
- If for other shader like plastic or metal, it is used with Fresnel to control the reflectivity change depending on the surface / viewing angle.

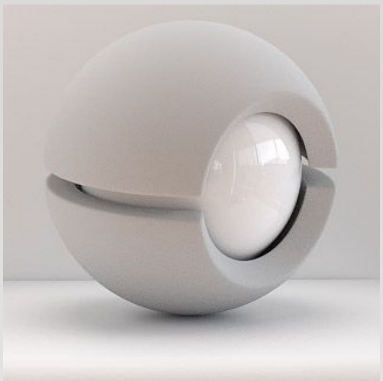

IOR 1,35

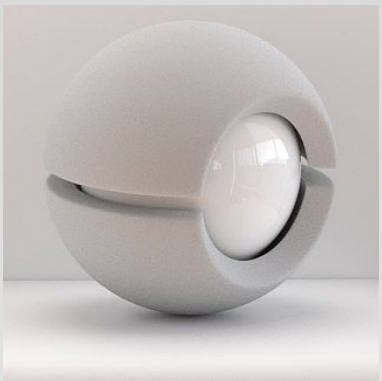

IOR 1,45

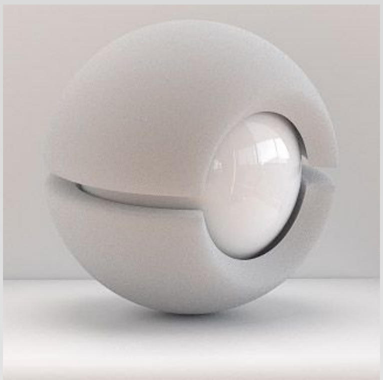

IOR 1,85

SSS

Stands for Subsurface Scattering. It simulates the scattering of light beneath the surface of an object.
- SSS Color controls the color of the effect.
- SSS Scale controls how far the light goes through the surface.


SSS Scale 0,0

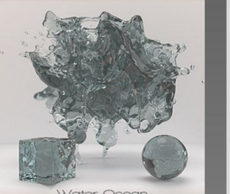

SSS Scale 0,01


SSS Scale 0,05

Glass Magic

Painted Wood 05

Glass Mirror


Water Ocean

Glass Broken

Plastic Translucent

Ice 02

Concrete Polished 03

Leaf

Plastic Scratched 02

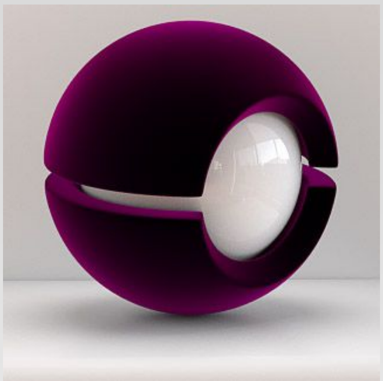
Satin Bump

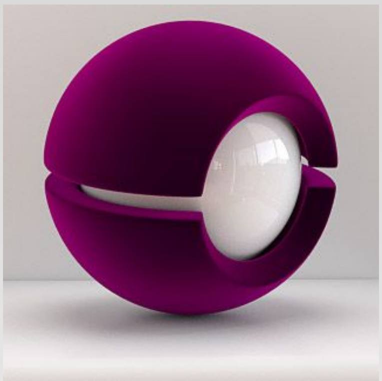
Translucent

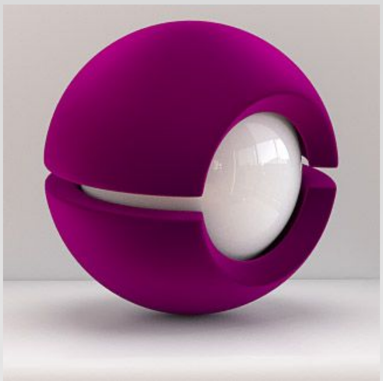
Lets light passes through it and bounces back to light up the front faces. So take care to have enough Transmission Bounces in your render settings, otherwise you will have a dark shader, or completely black if no Transmission Bounce at all.

Velvet

Like real velvet fabric, for clothing and fabric. It gives the effect of thin hair all over the shader that bend light accordingly.


Velvet Sigma 0,35


Velvet Sigma 0,65


Velvet Sigma 1

Map Projection

Image map based shaders have an alternative for objects without UV unwrapping, they are called "*Whatever Name* NoUV".
With these shaders you will find new parameters :

- Map Projection : projection to use to map the texture.

- *Box* will map the image to the 6 sides of a virtual box, based on the normal, using XY, YZ and XYZ coordinates depending on the side.
- *Flat* will use the XY coordinates for mapping.
- *Sphere* will map the image to the sphere using Z axis as central.
- *Tube* will map the tube to the sphere using Z axis as central.

- Mapping : it's where you can offset, rotate or scale your image map.

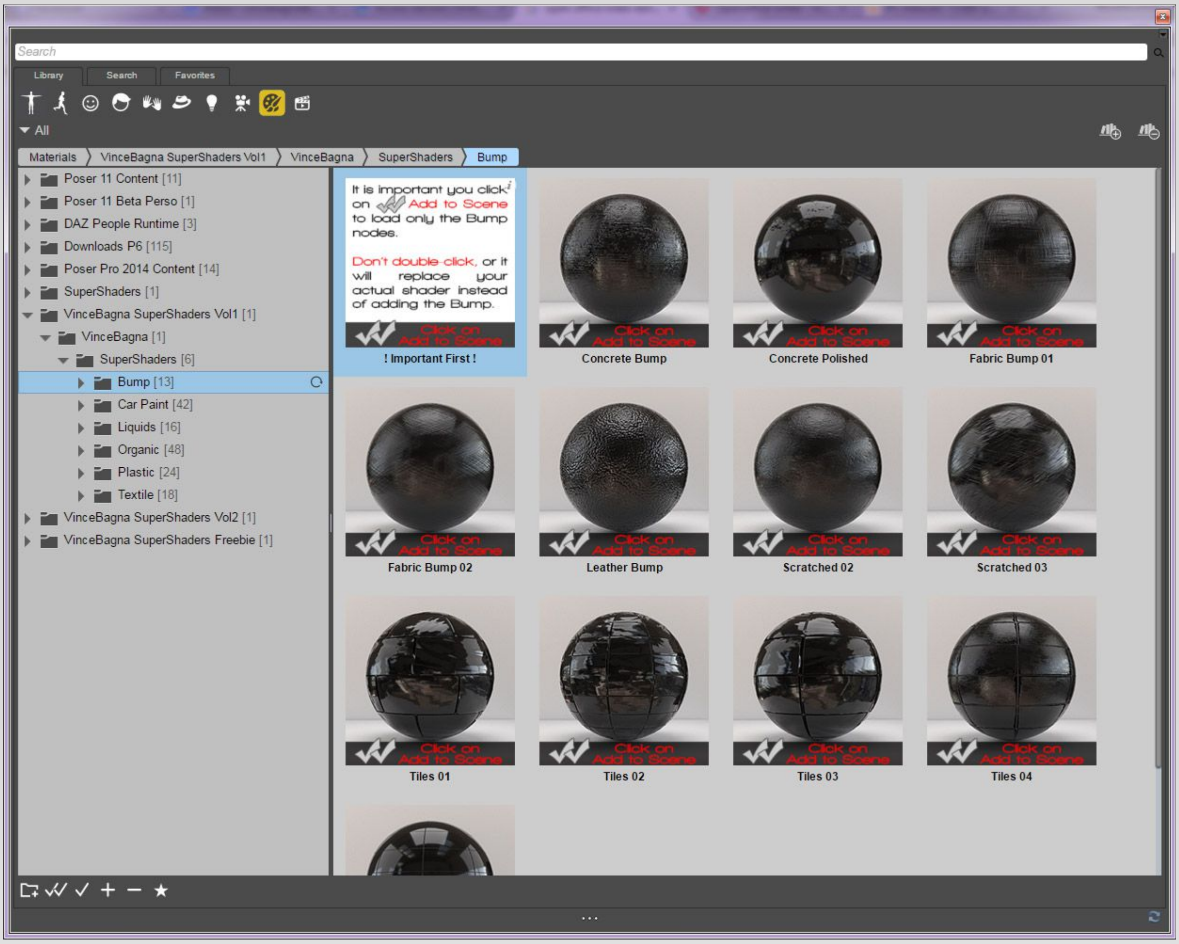
- Texture Coordinate : here you can change the type of coordinate to use for the texture. There is a bunch of different options, I will cover the 2 main ones.

- *Generated* : automatically-generated texture coordinates from the vertex positions of the mesh without deformation. In short, for non UV unwrapped objects.
- *UV* : to use the UV map of the object.

Using Box Projection with Generated Texture Coordinate for non UV unwrapped object is far to be perfect, mostly with patterned image map. Though you can quickly texture non unwrapped objects easily, even with the other Map Projection options.

Bump

You will find a folder called **Bump** with several bump nodes to add to your shader,



It is important you click on Add to Scene to load only the Bump nodes.

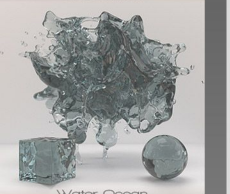
Don't double-click, or it will replace your actual shader instead of adding the Bump.

You will probably try first to load them like any other material files, by double clicking on it. **MISTAKE!** Haven't you seen that first thumbnail called '**! Important First !**' ? It's there to tell you there is something important about the Bumps there,,, obviously ;)

Glass Magic

Painted Wood 05

Glass Mirror



Water Ocean

Glass Broken

Plastic Translucent

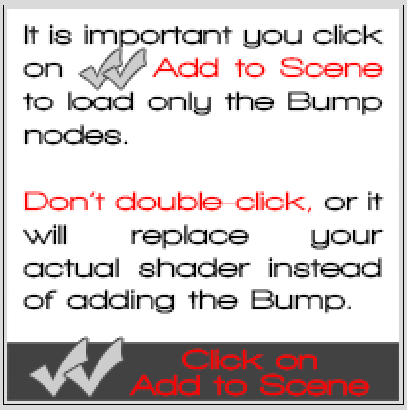
Ice 02

Concrete Polished 03

Leaf

Plastic Scratched 02

Satin Bump

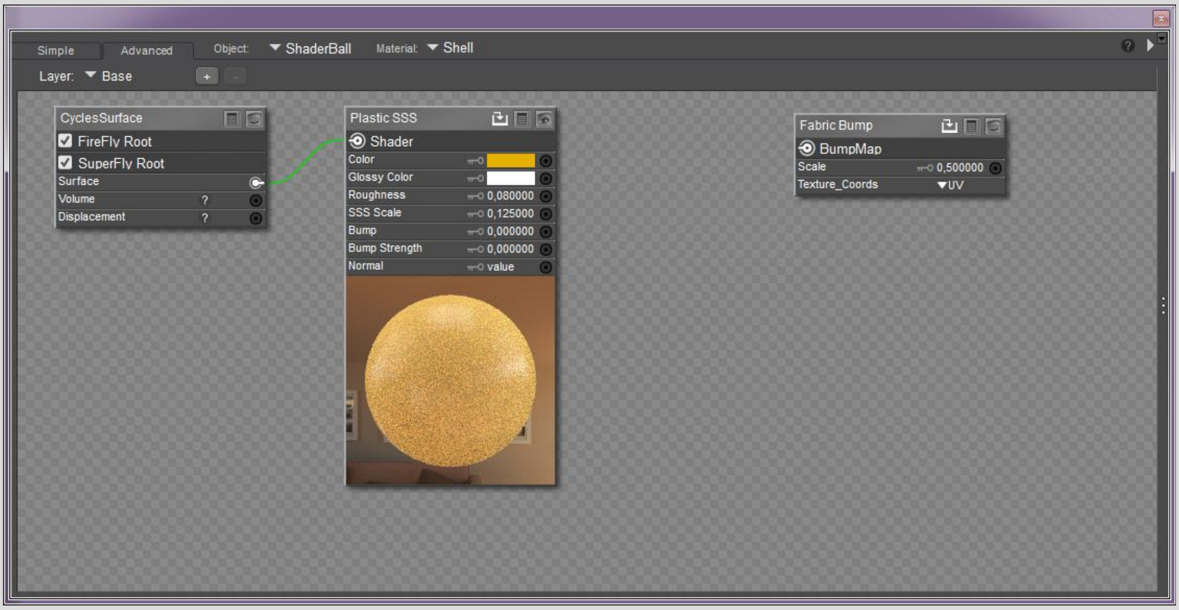


It is important you click on ✓✓ Add to Scene to load only the Bump nodes.

Don't double-click, or it will replace your actual shader instead of adding the Bump.

✓✓ Click on Add to Scene

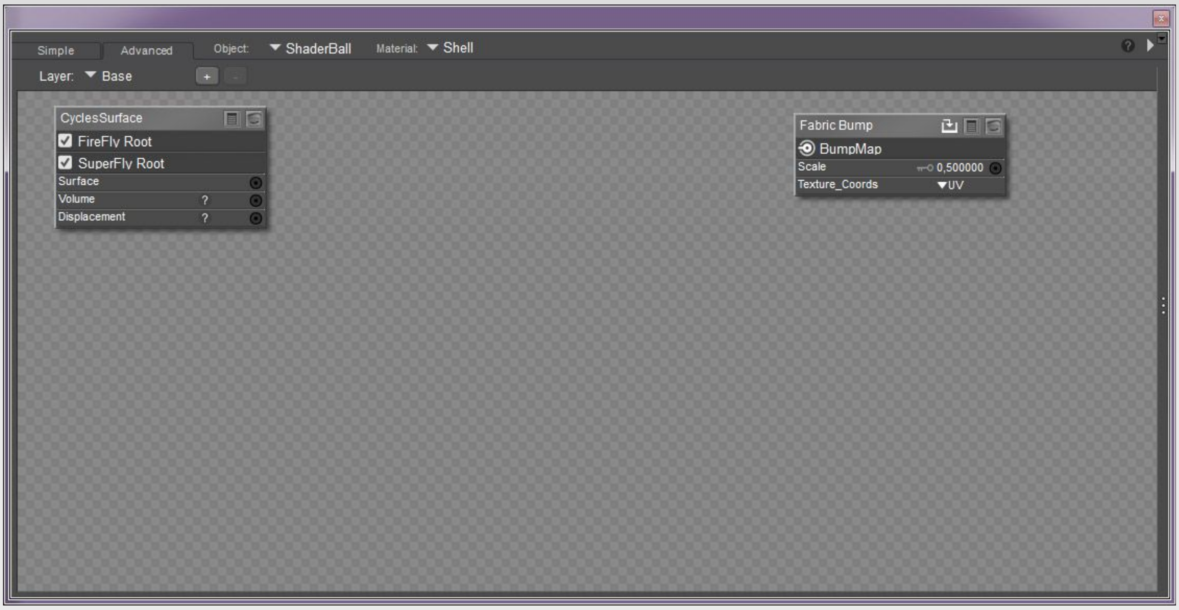
And those cute little double check marks on the bump icons ? Don't tell me I have made them for nothing. They are there to tell you to load the Bumps by first selecting the one you want, then click on the double check mark at the bottom of your library to **add** them in your material room, and to **not replace** the shader you have there,



Color
Glossy Color
Roughness 0,080000
SSS Scale 0,125000
Bump 0,000000
Bump Strength 0,000000
Normal value

Scale 0,500000
Texture_Coords UV

Add a Bump by clicking on the double check mark, then plug it into the bump input if it's a bump map node, or the normal input if it's a normal map node.

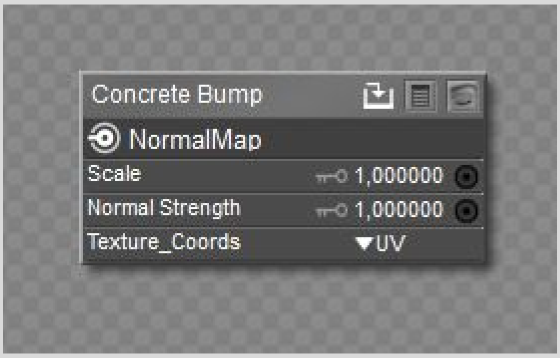


Scale 0,500000
Texture_Coords UV

Load the Bump by double clicking on its thumbnail, bad bad bad :)

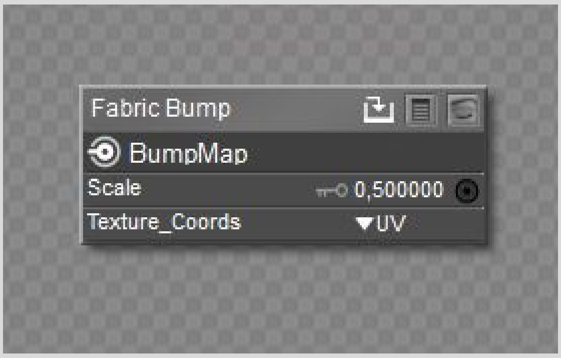
Keep calm if you already double click faster than light on it, don't forget CTRL+Z is your God ** giggle **

Now, you will probably note that some Bump nodes are NormalMap, and some others are BumpMap.


Concrete Bump
NormalMap
Scale 1,000000
Normal Strength 1,000000
Texture_Coords UV

Next to the node output, see Normal Map.

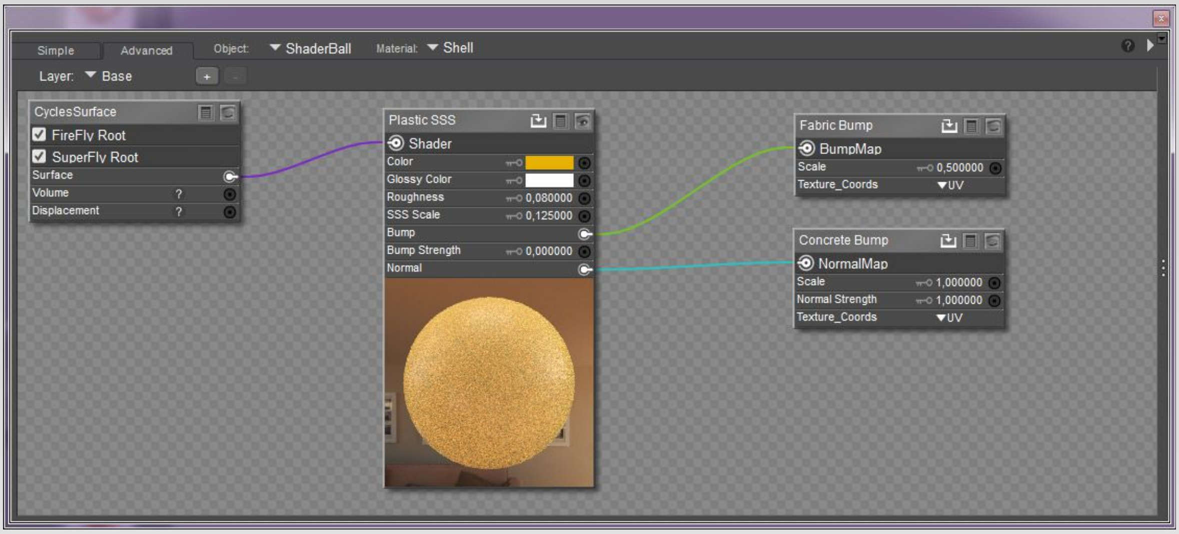
Plug the NormalMap output into the Normal input of the shader node, and set the strength with the Normal Strength slider of the Bump node.


Fabric Bump
BumpMap
Scale 0,500000
Texture_Coords UV

And here, see BumpMap.

Plug the BumpMap output into the Bump input of the shader node, and control the strength with the Bump Strength slider of the shader node.

Hey, you can even use a NormalMap node AND a BumpMap node at the same time.



Object: ShaderBall
Material: Shell
Layer: Base

CyclesSurface
☑ FireFly Root
☑ SuperFly Root
Surface
Volume ?
Displacement ?

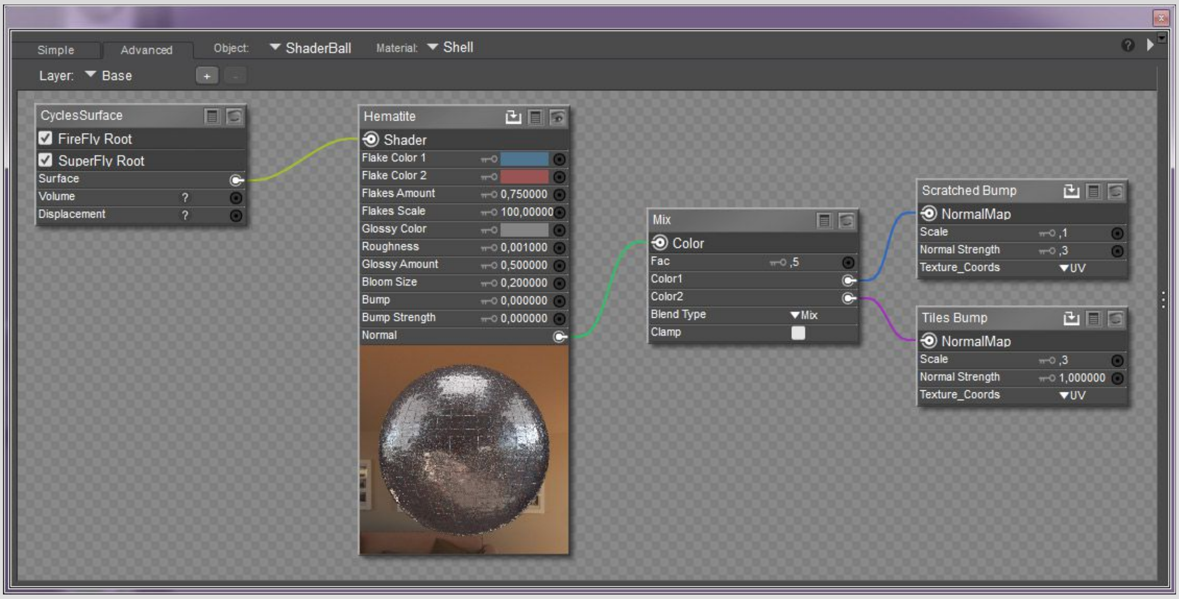
Plastic SSS
Shader
Color
Glossy Color
Roughness 0,080000
SSS Scale 0,125000
Bump
Bump Strength 0,000000
Normal

Fabric Bump
BumpMap
Scale 0,500000
Texture_Coords UV

Concrete Bump
NormalMap
Scale 1,000000
Normal Strength 1,000000
Texture_Coords UV

It can give great results!

Or even mix 2 NormalMap nodes or 2 BumpMap nodes together with a Color Mix node!



Simple | Advanced | Object: ShaderBall | Material: Shell

Layer: Base

CyclesSurface
☑ FireFly Root
☑ SuperFly Root
Surface
Volume ?
Displacement ?

Hematite
Shader
Flake Color 1
Flake Color 2
Flakes Amount 0,750000
Flakes Scale 100,00000
Glossy Color
Roughness 0,001000
Glossy Amount 0,500000
Bloom Size 0,200000
Bump 0,000000
Bump Strength 0,000000
Normal

Mix
Color
Fac ,5
Color1
Color2
Blend Type Mix
Clamp □

Scratched Bump
NormalMap
Scale ,1
Normal Strength ,3
Texture_Coords UV

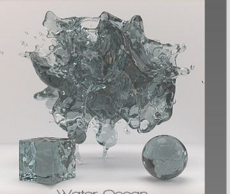
Tiles Bump
NormalMap
Scale ,3
Normal Strength 1,000000
Texture_Coords UV

A lot of possibilities and so much fun!

Glass Magic

Painted Wood 05

Glass Mirror


Water Ocean

Glass Broken

Plastic Translucent

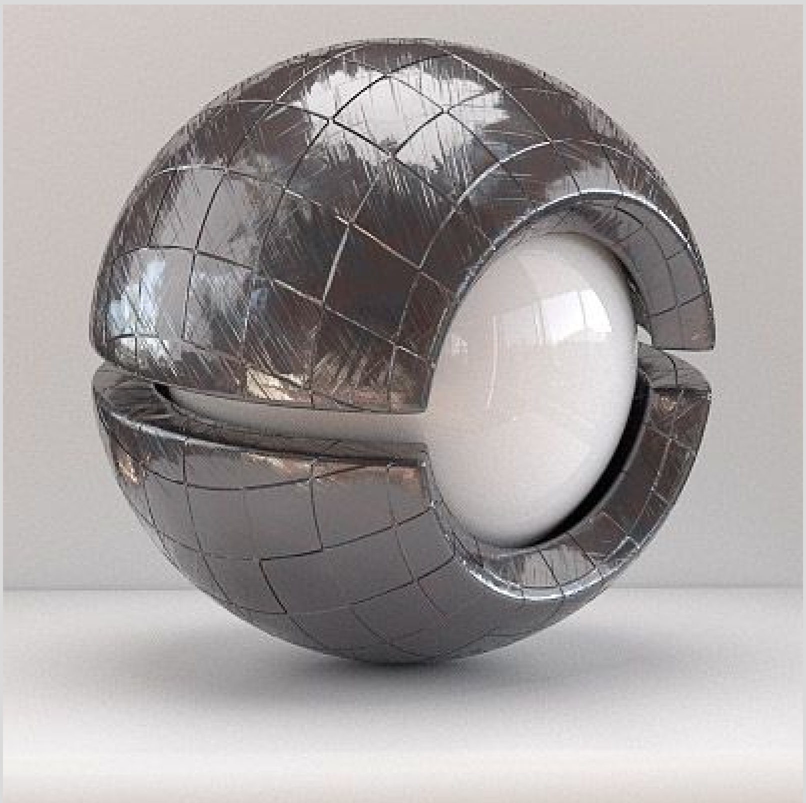
Ice 02

Concrete Polished 03

Leaf

Plastic Scratched 02

Satin Bump



Car Paint

Not much to say about the Car Paint shaders as their settings are self explanatory.

However, don't forget you can play with the settings to alter the look, and add some bump too. Don't stay limited with the base colors from the library, **be creative**!

Liquids

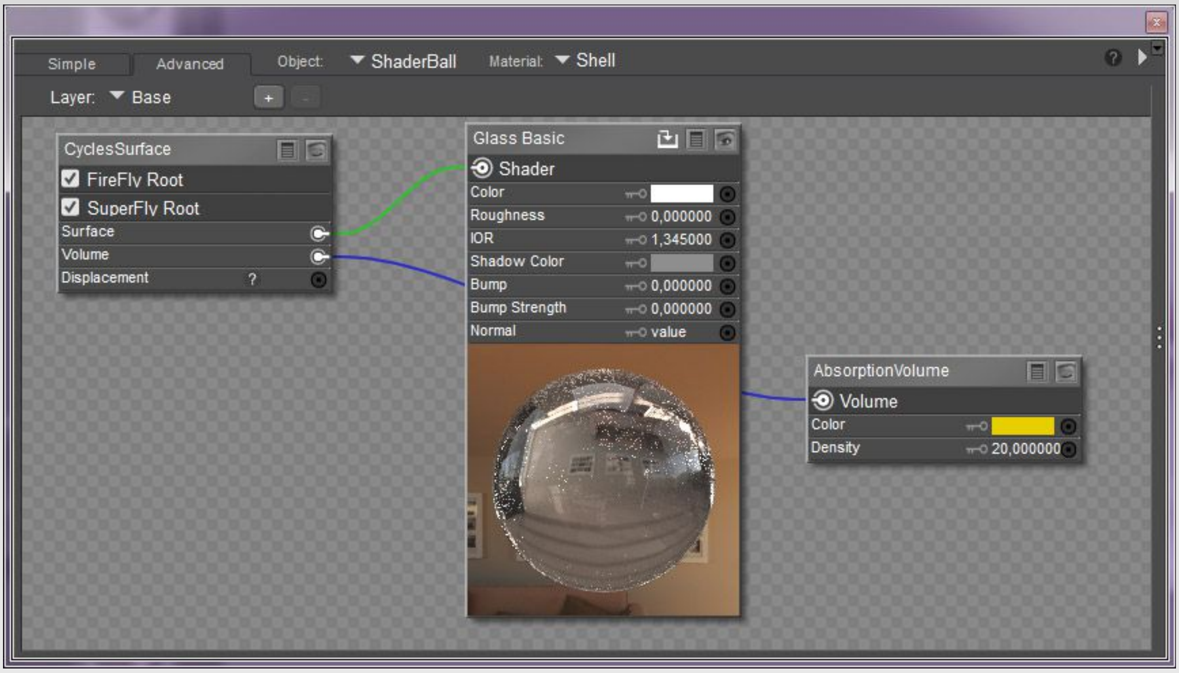
The Liquids shaders are composed of 2 kinds of shaders: **volumetric** and **non volumetric** shaders.

Non volumetric shaders are the usual shaders, nothing special there.

Volumetric shaders are shaders that use the Volumetric rendering ability of SuperFly/Cycles. For now, you can render **volumetric shaders only with CPU**. It may change in the future with some Poser updates.

So it may be slower to render, because of CPU rendering speed vs. GPU rendering speed. Though, and it's what makes SuperFly great to not be a PBR and to be able to play with the 'virtual reality' of the shaders, we can ***'fake' the volumetric shader so the GPU can render it***, and the result is very similar.

So each time you see something plugged into the Volume input of the Root node, it's for the CPU.



Object: ShaderBall   Material: Shell

Layer: Base

CyclesSurface
☑ FireFly Root
☑ SuperFly Root
Surface
Volume
Displacement ?

Glass Basic
Shader
Color
Roughness 0,000000
IOR 1,345000
Shadow Color
Bump 0,000000
Bump Strength 0,000000
Normal value

AbsorptionVolume
Volume
Color
Density 20,000000

In this example (I loaded the Beer shader), if you render with GPU, you will have a lovely Water shader instead of your Beer one.

I hear from here: *'Hmmm... wait! Water instead of Beer?! Are you kidding me?!! Give me my beer!'*

No panic! For this particular example, there is a Beer Fake Absorption shader that will give you Beer with CPU and GPU ;)

Isn't it great?!! ^^

It's not true for all volumetric shaders of this pack though. It's just that I have a HUGE crush for beer... :D Hey it's a joke there, well, I like beer too but no, though from time to time I like to have a... well, OK let's go on! ;)

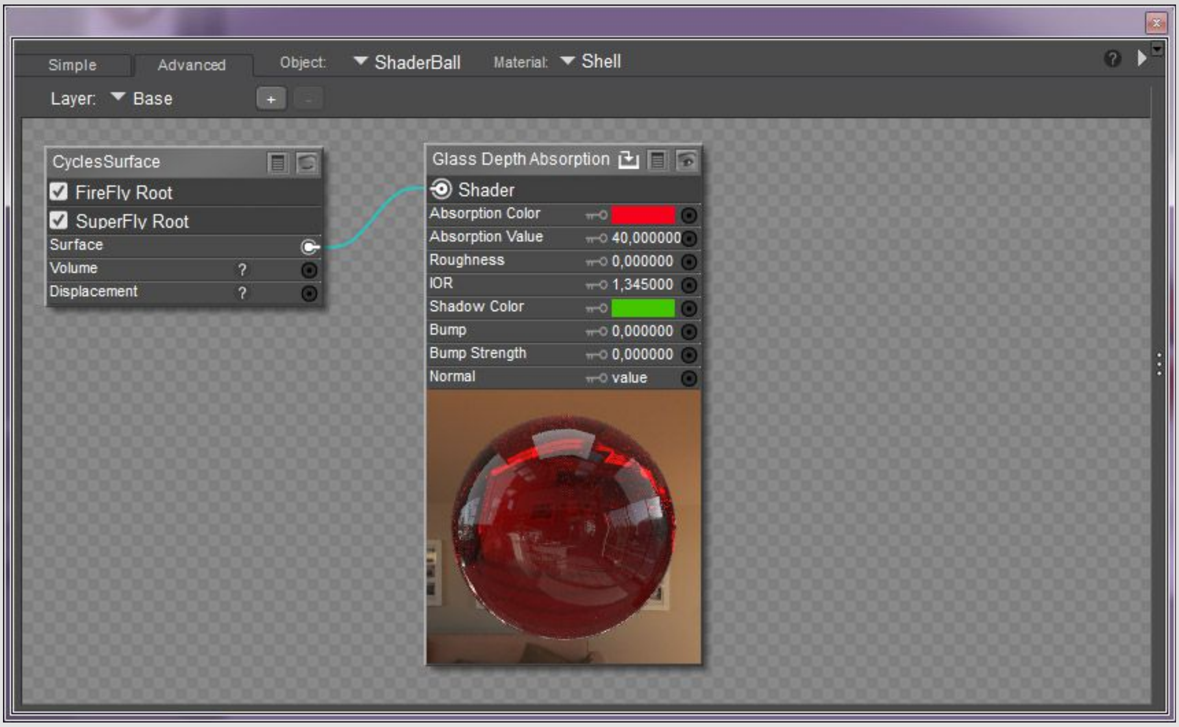
You can change the color, may it be in the volume nodes if it's a volumetric shader, or in the shader node if it's one that is faking it, and you will get another liquid! Try, you will see it's fun!

You will also note a **Shadow Color** input on some Liquids shaders. What's this?

In fact, the original Cycles render engine has some issues when rendering the shadows with glass shaders. They are too solid, and are not that good to let the light pass through. SuperFly has of course inherited this issue.

No problem though, as we can easily turn around this problem by saying to SuperFly how much solid the shadow has to be! This is what the Shadow Color is used for. The grey color acts like a rule from 0 to 1, from white to black. Darker color gives darker shadow, thus less light going through. Lighter color gives lighter shadow, thus more light going through. Logical, no? :)

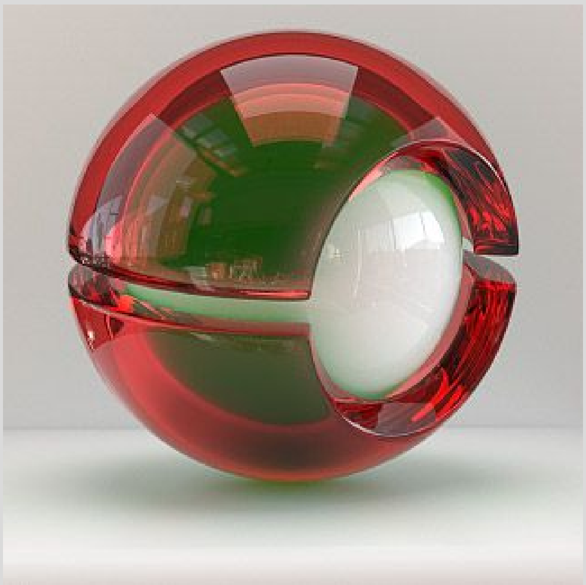
There is also a cool thing with that Shadow Color thing. **You can color the shadow!!** ... You already thought of that? ... So try now! ;)



CyclesSurface
☑ FireFly Root
☑ SuperFly Root
Surface
Volume ?
Displacement ?

Glass Depth Absorption
Shader
Absorption Color 🟥
Absorption Value 40,000000
Roughness 0,000000
IOR 1,345000
Shadow Color 🟩
Bump 0,000000
Bump Strength 0,000000
Normal value

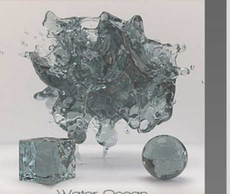
See the green shadow here? Hey! Now that is weird, but so cool! ^^



Glass Magic

Painted Wood 05

Glass Mirror


Water Ocean

Glass Broken

Plastic Translucent

Ice 02

Concrete Polished 03

Leaf

Plastic Scratched 02

Satin Bump

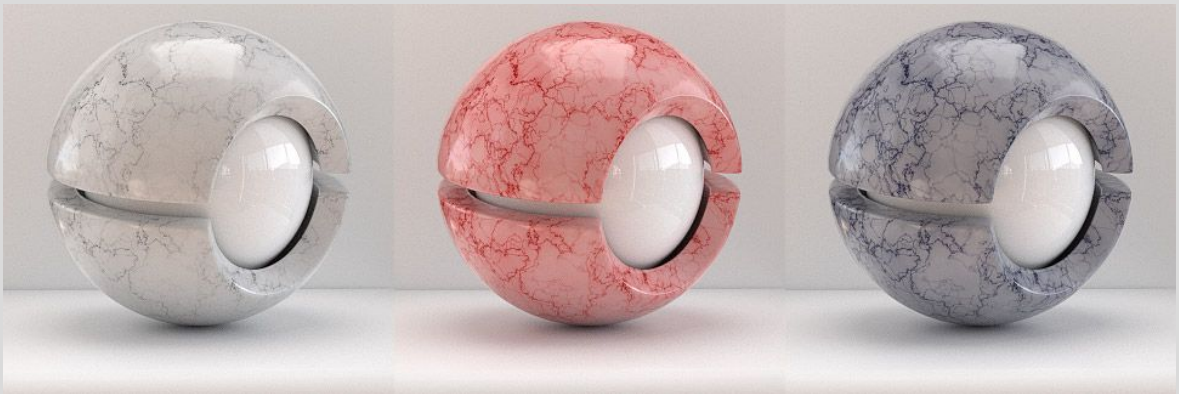
Organic

Organic is a collection of very different shaders.

Some are very specific (like chocolate… hmmm chocolate!). Though if you change some settings like roughness, SSS color, etc… you will get a completely new shader. So taste it as you wish!!

Others are quite general like ceramic or pearl. They can cover a wide range of materials with only a change of color. But be honest, you will try the other settings too, no? ;)

Some shaders like the Concrete ones have a Color Overlay setting. It means that while the shader is map based, you can change the color using this setting.



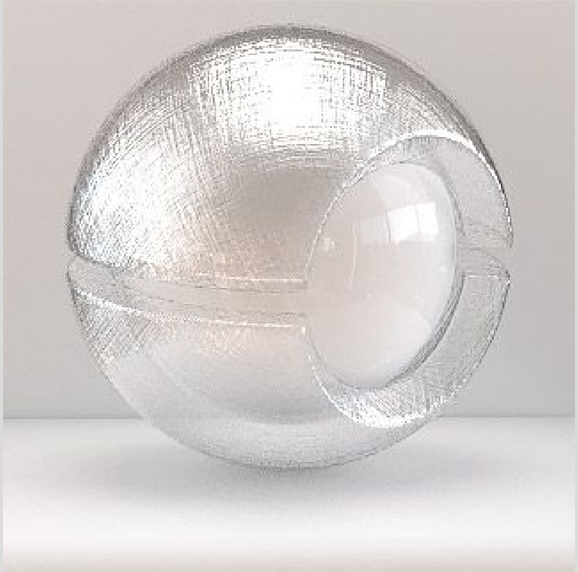
Plastic

Hey that's a lot of plastics there! ^^

Though they may not cover all your plastic needs... so I encourage you to change the exposed settings to alter the look of the shaders. And if you're daring (just kidding, I more than encourage you too!) **you can open the compound nodes** to see what's inside! (click the little arrow thingie next to the expand icon).

Here again, don't forget that you can add a Bump to any shader, it could change drastically the look, thus lead to a new and interesting thing!

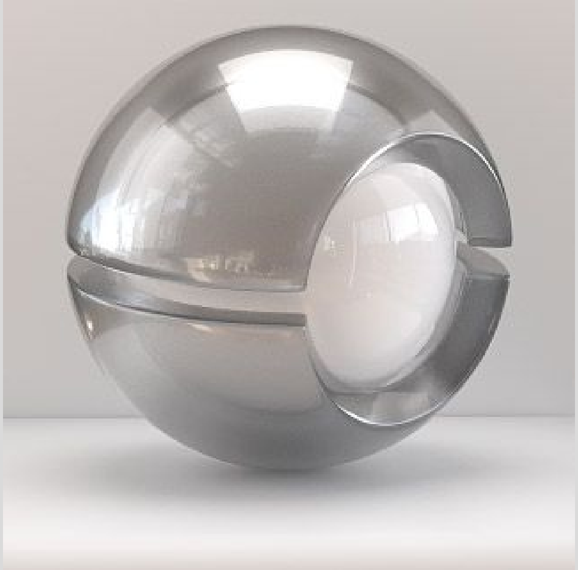
Clear Plastic Light as loaded.



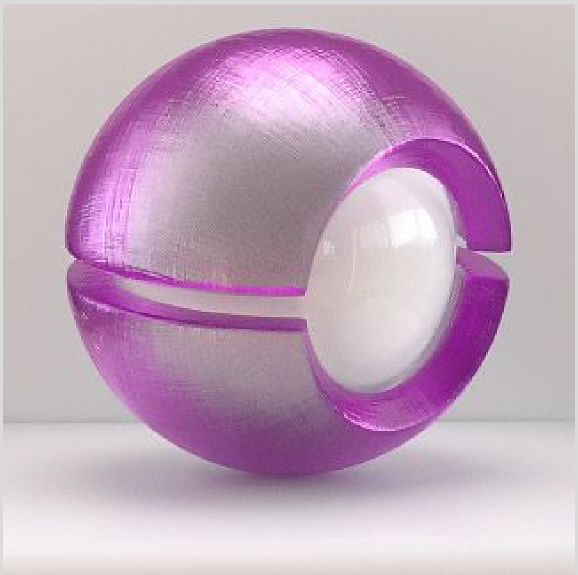
Clear Plastic Light with Fabric Bump 02.

Plastic Scratched 02 from the library.

Changed Color, Roughness, Blend and Bump node scale.



Plexiglas from the library.



Changed Edges Color and added Fabric Bump 01 ---> you get some transparent fiber plastic!

So now you see you have those wooonnnnnnderful shaders from the pack (*shameless self congratulations...*), but you're not restricted to those beauties, you can have way much more by adding Bump and playing with the settings.

And remember, you can open the compound nodes to see the inner guts, and have some fun there too ^^

Textile

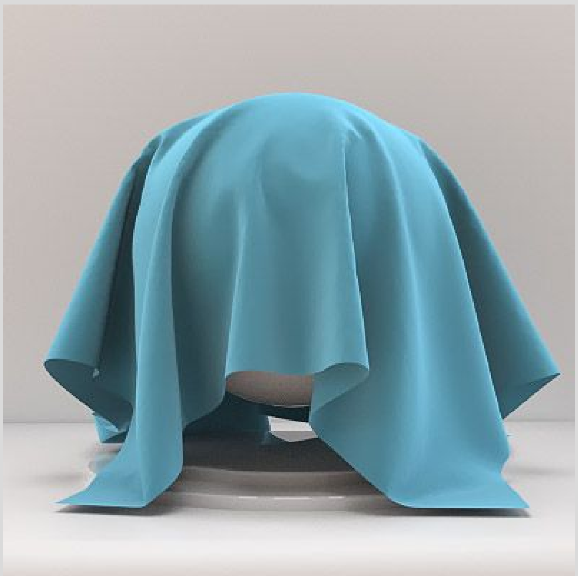
I will talk about the basic textile shader called Fabric, because a lot of the other fabric shaders are based on this one.

Diffuse Color : what it says, the diffuse color or diffuse map :)
Diffuse Texture : if you want to add a texture to add more details (you can plug your bump map here)
Texture Mix : the mix factor for the Diffuse Texture
Velvet Mix : the mix factor for the velvet effect
Velvet Strength : how strong is the velvet effect
Velvet Saturation : to saturate the velvet effect ^^
Velvet Brightness : can you find it? ;)
Translucency Color : color of the light passing through the fabric
Translucency Strength : how much light is passing through the fabric
Bump, **Bump Strength** and **Normal**, you know them ^^

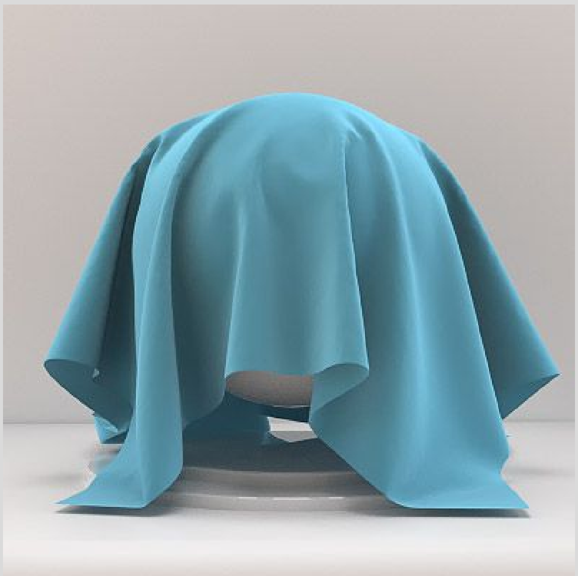
Now you can play with the translucency settings to change the thickness visual effect of the fabric. Dense fabrics are more opaque to light. Thin fabrics let some light passing through. This is translucency.

You can set the Translucency Color the same as Diffuse Color, or use the

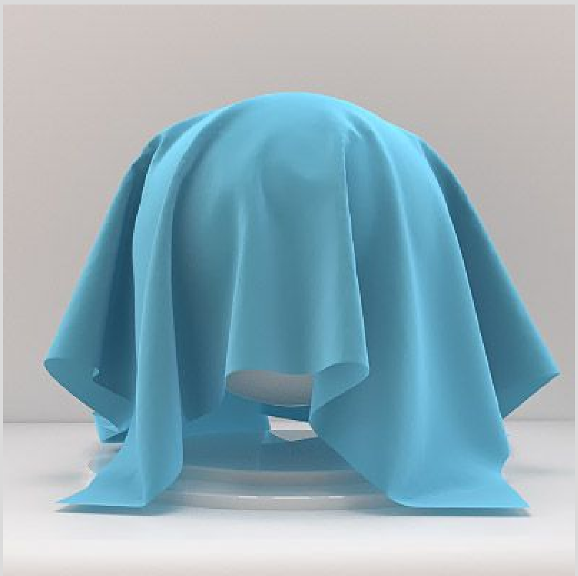
same map. Or you can set it to something slightly brighter to accentuate the translucency effect. Or even a completely different color or map :)



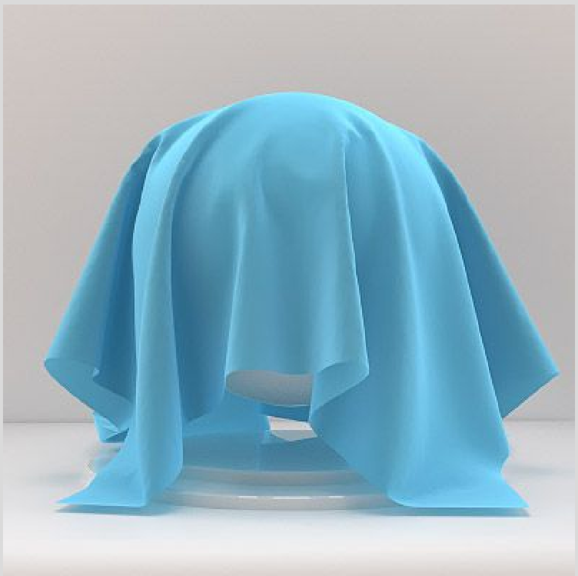
Translucency Strength set to 0



Translucency Strength set to 0,2



Translucency Strength set to 0,5



Translucency Strength set to 0,75

Notes

As these shaders use the Cycles Root, they are **only for SuperFly**. They won't work with FireFly.

Some shaders will take more time to render than others. This is due to the very nature of the shaders themselves, i.e. shaders with glass, translucency or SSS need more time to get rid of the noise while rendering.

Shaders are very dependent of the scene lighting. In fact, to get best results, you should always have something to reflect, i.e. some props around your scene, or easier : a HDRI on an environment sphere surrounding your scene, or plugged into Poser Background ticking the Cast_Light box. It will then be used for lighting.

In any case, **reflective shaders need something to reflect**.

Don't hesitate to mix shaders using the MixClosure node. You will get interesting results! ^^

The most important thing is : ***HAVE FUN !!!***

Special thanks to Dead Reckoning for his precious help and advice, and for having suggested some of the SuperShaders. Thank you Dead! ^^

Glass Magic

Painted Wood 05

Glass Mirror

Water Ocean

Glass Broken

Plastic Translucent

Ice 02

Concrete Polished 03

Leaf

Plastic Scratched 02

Satin Bump