Sparkle Shader

by Art On Bit GmbH - www.artonbit.com

Features:

The **Sparcle Shader** can be used to create an insane amount of **‚flakes‘** on surfaces, which behave correctly in regard to **specular lightning** and **reflections**.

Advantages:

- **No UVW mapping** necessary
- A lot of **control** and advanced parameters for more **complex mappings**
- It is a **shader**, no post effect and therefore **interacts** with the scene
- Several **Flake Layers** which can be mixed together individually
- Each **Flake Layer** has its own settings: **Layers, Sparkling, Particle Settings, Specular, Reflection, Fresnel**
- The internal **Fresnel** helps reproducing the „depth“ of the layer and also optimizes rendertimes
- Additional **Fresnel** slot for shaders
- The **Sparkle Shader** can be used for example in the **Luminance Channel** -> You still can control **Reflection** and the **Specular** of the material **independently**
- Within the **AR**, the shader can **raytrace reflections**
- Very **small memory footprint,** does not need any internal maps
- Works with **Subpolygon Displacement (SPD)**
- In-/Exclude lights per **Sparkle-Shader** for speed optimization

Introduction:

What does it do?

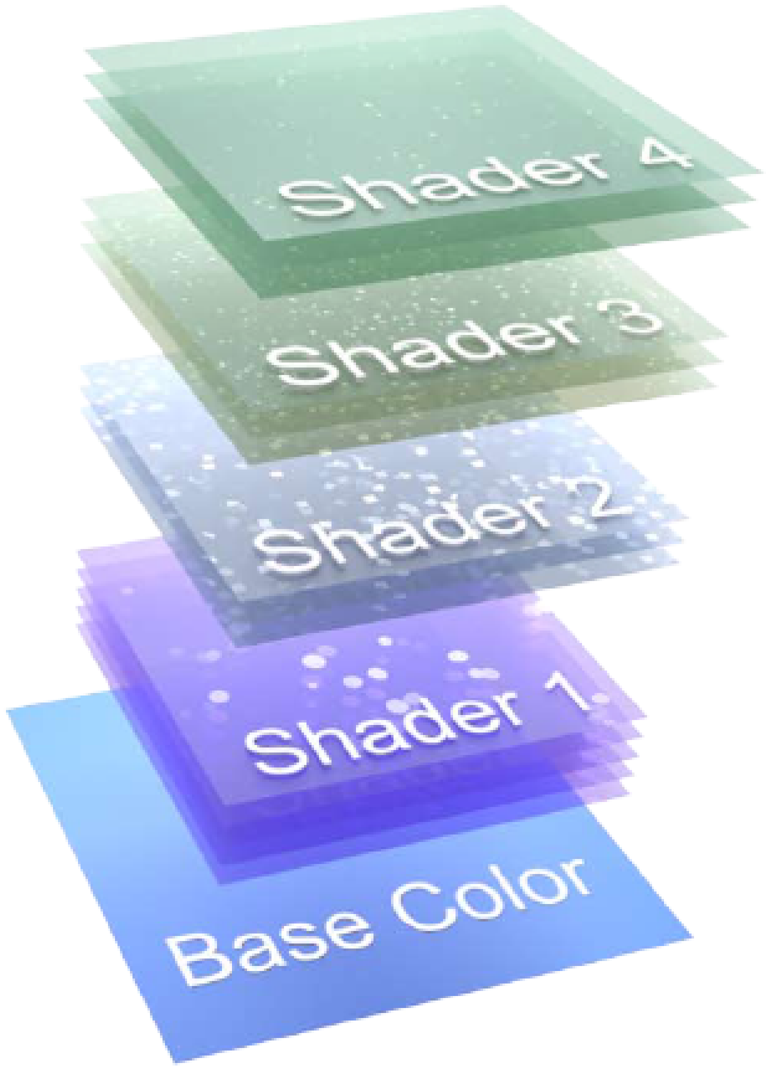
The **Sparkle Shader** generates millions of flakes which can be lit by specular(s) or by tracing reflections. Each of the four **Flake Layers** can calculate both types at the same time. The specular can be disabled by setting its strength to zero for an individual **Flake Layer**. The two effects can easily be mixed by the individual strengths. The color of the flake is either given by the light(s) or reflection, or can be controlled by shaders **(Advanced-Maps)** or the Fresnel.

The sparkling effect is created by assigning the flakes individual normals. If the resulting sparkling (especially for speculars) is not strong enough or you just need a more extreme effect, you can enable the sparkling options for a variety of different possibilities.

You can choose either round **„Ellipsoid"** or polygon shaped **„Cuboid"** flakes, which will result in a different look overall. The flakes are distributed by an internal noise (lets call it **flake noise**), which is applied by default in object space, but can also be projected into UV space for manual mapping (see global option **„Use UVWs for Flake Noise"**). UV mapping will also be necessary, if an object deforms, otherwise the deforming polygons will „travel through the noise" in the animation.

As by default the internal **flake noise** is in object space, **you do not have to worry about setting up UVs** for any objects that do not deform. This will make sure, that you always have „undeformed" flakes, no matter how „crazy" or unmappable the topology of the mesh is.

It is possible to give each flake only a single color by enabling the option **„One Color per Flake"** in the **Advanced-Maps.** If this option is not enabled, the flakes surface will be colored by the shader and can therefore show any pattern and colors. The **Sparkle Shader** can also calculate a single normal for the whole surface of the flake, which is closer to how it is in reality. This can be done by selecting **„Unique Normal per Flake"** for the **„Flake Normal".** If you use **„Surface based Flake Normals"**, you get more control, but the flakes normal will „follow" the surface normal. This means, that for example a flake on a sphere will reflect as if the flake is actually part of the surface. With **„Unique Normal per Flake"**, flakes will appear flat, also on a sphere.



In which Material-Channel do I use it?

You can use it in any channel, but the best place is probably the **Luminance Channel.** The calculated speculars or reflections can be brighter than the underlying material, which can be best achieved with the **Luminance Channel**. This allows to still control the materials specular and reflection independently.

If you use the **Luminance Channel** and the **Color Channel** in a Cinema Material, you will notice that the colors of both channels are **added**.

This is the reason, why the **Sparkle Shader** has its own global shader slot for a **„Base Color"** shader. So instead of using the Material's Color Channel, you can also use the **„Base Color"** slot to do the same, with the advantage, that the flakes will now have an alpha-blending. This will lead to **correct colors**, as the flakes will be blendet against the color sampled from the **„Base Color"** shader. The „Base Color" slot is meant to be used, if the Sparkle Shader is placed in the Materials Luminance Channel. If you use the **„Base Color"**, make sure to **disable the Material's Color Channel,** as it would be added to the Materials Luminance Channel (and colors therefore once again mixed by **addition**).

Antialiasing / Interactive Render Region

The **antialiasing** is **essential** for the **look** of the **Sparkle Shader.** If you use the **Interactive Render Region**, you won't see the same result as if you „render a region" or render in the picture viewer. So don't let you fool by the **IRR** or the look you will get with low **antialiasing**! This of course is also valid for the Material Previews, as they do not have the same antialia-sing.

See **„Not Enough Flakes?"** for more details.

Flake Layers and Sub-Layers

The **Sparcle Shader** contains four sub-shaders (**„Flake Layer 1", „Flake Layer 2", „Flake Layer 3", „Flake Layer 4"**). Each **Flake Layer** is composed of several **Sub Layers** and has individual settings for the flakes, the sparkling, the specular, a fresnel and the reflections.

The shaders are mixed according to the setting in the individual **Sub Layer**, so each **Flake Layer** can be blended differently. The **Sub Layers** of each **Flake Layer** are mixed by the „**Sub Layer Blend Type**" setting in the individual **Flake Layer**.

How are Flakes Distributed?

Flakes are distributed by an internal multidimensional **flake noise**. Use the global option „**Global Seed**" to change the random seed for the **flake noise**.

The **flake noise** can be scaled with the „**Grid Size**" parameter for each individual **Flake Layer**. With the **„Global Grid Scale"** option you can scale all four **Flake Layers** at once. This allows to globally scale the effect created by all four layers. Like this, a **Preset** has only to be modified by this one parameter to scale the effect. Note: if you use textures in the **Advan-ced-Maps**, you will need to scale them manually too. They are not automatically scaled.

The **flake noise** contains a big amount of flakes, which can be further increased by the **„Sub Layer Count"** if necessary**. T**he **flake noise** contains so many flakes, that it will completely fill any surface it is applied to. The different **„Flake Layer"** settings will change which particles you can currently see**.** For example the „**Flake Density"** defines, how much of the flakes you get to see and will randomly blend them out. So reducing this parameter will show you less and less of the **flake noise**.

This is at the heart of how the flake engine works. Some options like the „**Sparcle Cutoff"** will actually further reduce the amount of flakes that you can currently see. The hidden flakes are not visible under the current angles (between lights and the flakes and of the camera to the flakes).

Specular

The **Sparcle Shader** calculates a specular (Blinn-Phong) for all flakes based on their orientation and the position of the light(s). If you set the „**Flake Normal Deviation**" to zero, the flakes normals will all exactly represent the normals of the surface and therefore give a flat appearance. The **„Flake Normal Deviation"** will also have an effect on the way the effect looks in an animation. Strong deviations from the normal will create more active reflections/speculars. The low angles (lets say, between 0-30°) have a very big influence how the **Flake Layer** will look.

As the specular must be evaluated for all the light sources (or the ones included or not excluded), it is more expensive to calculate as the raytraced-reflections. The specular allows to easily create different „layers of depth" by using different settings for the **„Specular Width"** and **„Specular Strength"**. You can achieve a similar „depth feeling" by using several different reflection layers and individual **Fresnel** settings.

Raytracing Reflections

Within the AR you can calculate reflections for each individual flake. You can also colorize the reflection by the **„Color Shader"** in the **Advanced - Maps**. The flakes are all directly on the surface when simulated by the shader, which is not as it is in a real paint. To prevent flakes to be visible at steep angles, you can use the **Fresnel** of the **Flake Layer**, which will also speed up calculation.

Sparkling Overview

The „default" sparkling effect is created by assigning each flake its own normal. This is controlled by the **„Flake Normal Deviation"** parameter. This parameter has only effect, when **„Surface based Flake-Normals**" is selected for the **Flake Normal**. If the **„Flake Normal Deviation"** is set to zero, all the flakes are exactly aligned to the surface normal. As this is only a simulation of the real effect, values bigger than 90° are allowed and will lead to a different look.

As In reality, there must be movement in a scene for sparkling. If you stop the camera, the object or the light that was moving, also the sparkling will stop.

To get more sparkling, in animations, especially for speculars, you can use a simulated sparkling effect by enabling a „**Sparkle Type". ** The effect is created by blending flakes in and out. If you enable the **„Sparkle Cutoff**", the flakes will be cut to create an even stronger sparkling. The Sparkling effect is created by three different options.

- The „**Sparkle Type**" defines which type of function should be used for blending the flakes in and out.
    - **Type 1** - This type is based on a **smooth** function - blends flakes slowly and softly in and out, this is the softest type
    - **Type 2** - This type is based on a **smooth** function, but has a higher amplitude - blends flakes softly but twice as strong as **Type 1**
    - **Type 3** - This type is based on a **smooth** function, it resembles loosely a saw tooth function - blends flakes more erratic than **Type 2** and **Type 4**
    - **Type 4** - This type is based on a piecewise linear (saw tooth) function - blends flakes abruptly, this is the strongest type, especially with a high **„Sparkle Frequency"**

- The **„Sparkle Frequency"** defines how fast a flake will be blended in and out by the selected **„Sparkle Type".**
- Finally you can choose between eight different **„Sparkle Cutoff"** settings. The „**Sparkle Cutoff"** creates a strong sparkling, which is meant to be used for very small flakes, as the flakes will be cut and blended in and out in different ways.

Not Enough Flakes?

As mentioned in **‚How are flakes distributed‘**, stronger sparkling modes will reduce the amount of flakes that you see at the time. Sparkling does not add new flakes, but blends the existing ones in and out. So when using strong sparkling settings, it may be necessary to increase the **„Sub Layer Count“.**

If you see not enough flakes, make sure to check the following points, before adding more flakes by increasing the **„Layer Count“** or enabling another **Flake Layer**:

• **“Flake Density“:** can you still increase the **„Flake Density“**?

• **Antialiasing: Antialiasing** has a big influence, once you have flakes of the size of a pixel or less and/or if you use reflections. The **AA** is very important if you need such fine structures. For example if you make the **„Grid Size“** parameter smaller and smaller, you will notice, that from the point where particles get smaller than the size of a pixel, the noise does not seem to get smaller anymore. This is not due to the **flake noise** (you can zoom in to verify), but from the fact that the **AA** will „define“ how the noise will look with very small flakes, and not re-ally the shader anymore. If you render with **Min AA 1x1,** you will get a very different result than with **Min AA 2x2. Max. AA 2x2** also looks very different to **Max. AA 4x4**. So always keep this in mind, if the structures are not fine enough, the reason is most probably the antialiasing.

• **Specular Width**: if you use the specular, the **„Specular Width“** defines the size of the specular highlight. Smaller values will therefore show smaller definitions and less of the flakes.

• **Specular Environment**: is there enough light in the scene to actually show the flakes? Several lights will show more of the structure, as light will come from more angles. Area Lights create a more vivid definition.

• **Reflection Environment**: if you use reflection, are there enough bright objects/materials around which can be reflected?

• **„Small Grid Size“** if you use **„Unique Normal per Flake“** for the **„Flake Normal“**, you will get a more realistic behavior for the flakes, which is good for bigger flakes. In case the flakes are very small, you can change the **„Flake Normal“** to **„Surface based Flake-Normal“** and adjust the **„Flake Normal Deviation“.** Especially small values will „reveal“ a lot more flakes, but will also make the appearance more flat the closer you get to zero. You can also create sparkling with small angles, by enabling the simulated sparkling effect.

• **„Sub Layer Blend Type“**: if you set the **„Sub Layer Blend Type“** to **„Brightest“**, you will see a lot more of the flakes. This will also change the appearance.

If there are still not enough flakes around, you have the following options:

• Increase the **„Sub Layer Count“** (you can add a lot of additional layers)

• Enable another **Flake Layer.**

Is the Flake Noise not getting smaller?

If you need a very fine **flake noise,** which is actually smaller than a pixel, which you often do for this effect, you need good **antialiasing**. If your **max AA** is set to **4x4** for example**,** then the smoothness and the appearance of the **flake noise** is limited by that setting. You can make the **„Grid Scale“** smaller and smaller, but the appearance will stay the same. If you need finer **flake noise**, then you have to **increase the antialiasing settings**! With higher antialiasing settings, the flake noise is sampled at much more points for the same pixel, which finally allows to create a smoother/finer appearance.

Advanced - Maps, Even More Control

You can control many parameters by applying textures. You can use **3D volume shaders** or **UV mapped shaders**. The **„Color Shader“** map is the only of the **Advanced Maps** which does not „overwrite" another parameter (and therefore gray out a parameter).

The **„Color Shader"** can be used to either color flakes that use a specular or to colorize reflections (or both). If you want to colorize flakes by angle, you can either just use the built-in **„Fresnel Gradient"** or use a fresnel either in the **„Fresnel Shader"** slot or in the **„Color Shader"** slot.

If you use a **3D volume shader** in the **„Color Shader"** slot, you can choose to apply only a single color per flake when enabling **„Single Value per Flake"**. For example you can use **Cinema's Noise Shader** and set the **„Space"** option to any mapping except **„UV (2D)"**. In the case where you use a **2D Texture** or the **„UV (2D)"** option for the **Noise**, it will be a simple UV mapping, where colors can change anywhere in a flake.

You can also use a **3D volume shader** in all the other slots. There too, if you use a shader which is based on UV mapping (ie. 2D, or choosing settings which result in a 2D mapping), the mapping will be a simple projection, which is based on the mapping of the **Texture Tag**.

Note: The distributions of the shader(s) will be slightly altered, if the noise is a **3D volume shader**, as the internal **flake noise (see „How are Flakes Distributed")** is a multidimensional function, which will sample the noise in a special way. So with **„Single Value per Flake"** enabled, you will not get an exact match when using the same noise at the same time in another material channel.

Note: If you choose „**Texture**" as the option for „**Space**" in a Cinema noise, the mapping will also be dependent on the texture tag's projection. You can read more about the projections of the noise shader in the Cinema help.

Specular and Reflections, In-/Exclude

The strength of the specular is controlled by the „**Specular Strength**" and the stength of the reflections by the „**Reflection Strength**". Those two settings control the mix of the two effects. For efficiency, there is an include/exclude field for lights in the **Global** tab (also see **Exclusion Mode**). You can control exactly which lighs will affect the **Sparkle Shader**. The rendering will speed up, if you just include significantly contributing lights (or exclude non-contributing).

Area Lights

If you use **Area lights**, you will get more realistic speculars and a good feeling of depth, as light is coming from different angles. Of course you can achieve the same when using several **Omni Lights. Omni Lights** are in general faster to calculate than **Area Lights**.

Don't forget to enable the option **„Show in Reflection"** in the lights detail tab, if you want to see the light source in the reflections.

Fresnel / Global Flake Brightness

Each **Flake Layer** has its own **Fresnel,** which can additionally help to create different definitions and also the feel of depth. The in-built **Fresnel** („**Use Fresnel Gradient**") is additionally useful to optimize render-times. You can also use the in-built **Fresnel Gradient** to colorize flakes **by angle**.

You can additionally use the „**Fresnel Shader"** slot by enabling **„Use Fresnel Shader"**. This allows you to use a shader which will be multiplied to the calculated result of that **Flake La-**

yer. Note that you can actually use whatever shader you like in this slot, not just a **fresnel shader**. The calculated color for the **Flake Layer** in question will be multiplied by this channel.

If you want to control the Color/Brightness of the outcome of all the Flake Layers, us the „Global Flake Brightness" in the global tab. The „Global Flake Brightness" will be multiplied by the result of all **Flake Layers**.

So, what is the difference between the **„Color Shader"** and the **„Fresnel Shader"** slot?
The **„Color Shader"** can be mixed-in gradually by the **„Colorize Reflection"** and the **„Colorize Specular"** parameters and is not multiplied as is the **„Fresnel Shader"**. The **„Color Shader"** will be mixed by an alpha blending and not a multiplication.

Presets

The presets give you a base, where you can see how **Flake Layers** can be set up to create a convincing look.

If you want to adapt a preset, these are the places to look for:

- As the first thing, you probably want to change the **„Global Grid Scale"** in the **Global** Tab to scale the **flake noise** to fit to the new object. With the **„Global Grid Scale"** you can adapt all the **„Grid Scale"** parameters of all **Flake Layers** at once**.**
- If you want to change the colors, there are some places to look for:
    - The **Flake Layer's Fresnel** can define colors.
    - The **„Base Color"** in the **Sparkle Shaders** Global tab defines the „base color", which is the color used when no flake is around and also as the background for the flakes.
    - If you use reflections, the parameter **„Colorize Reflection"** controls the mix of the **„Color Shader"** in the **Advanced - Maps** with the color of the reflections.
    - If you use the specular, the parameter **„Colorize Specular"** controls the mix of the **„Color Shader"** in the **Advanced - Maps** with the color of the specular.
    - Don't forget that your lights will also color the flakes if you use the specular

Use with Global Illumination

If you want to use the a **Sparkle Shader** which is using the **„Base Color"** channel in the **Sparkle Shader's Global** tab (see **„Limitations"**) with **Global Illumination**, you have to copy the content of the **„Base Color"** channel to the materials own color channel and delete the content of the **„Base Color"** channel. You probably have to adjust colors for the flakes, as now the materials luminance channel will be **added** to the color channel (see **„In which Material-Channel do I use it?"**).

Performance

Which options are the most demanding in terms of performance?

- The **„Sub Layer Blend Type"** is probably the most expensive option if set to **„Brightest"** together with a high density. On the other hand, **„Brightest"** allows to create very dense flake distributions.. In the Presets **„Brightest"** is often used as **„Sub Layer Blend Type"** for the **„Flake Layer 1"**, the bottom-most of all the flake layers (especially for metallic car paint).
- **Reflections** are faster to calculate than **Speculars**, if there are **many contributing lights**. Area lights take more time to calculate than Omni lights.
- **Alpha/Transparency:** If you use the Sparkle Shader in the Alpha or Transparency Channel and you use reflections, render times will go up fast. It also depends on the complexity of

the scene of course, but this is a very expensive combination, as you can get a lot of light bouncing from flake to flake. This in combination with transparency or alpha will need additional time. In such a case you can limit the **„Raytracing Depth“** in the **Global Tab**.

Limitations

• If you use **Global Illumination** and the **„Base Color“** slot of the **Sparkle Shader**, the **Sparkle Shader** will only show the **direct lightning** (as if there was no **Global Illumination**). In order to make a Preset **„GI ready“**, just copy the contents of the **„Base Color“** shader to the Materials own Color Channel and delete it from the **„Base Color“**. See also **„Presets/Use With Global Illumination“** for details.

Global Shader Options

• Base Color

The „**Base Color**“ slot can be used to define a color shader, instead of using the Cinema material's color channel. This allows to mix the flakes with an alpha-blending towards the **„Base Color“** which results in correct colors for the flakes. See **„In which Material-Channel do I use it?“** for more information.

• Allow only brighter Flakes as Base Color

If this option is enabled, flakes will not be darker than the **„Base Color“.** This option is useful for example to create metallic paint.

• Raytracing Depth

Determines how much times the shader is allowed to reflect itself and also how much times a reflection is allowed to bounce. I recommend to keep this value small, as rendertimes will increase pretty fast and usually there is no big gain. However, if you are to render something right in-between two mirrors and you use reflection, you maybe need to increase this value to actually get a color for the reflection.

• Use UVWs for Sparkle Noise

Flakes are by default in object space. If you want to UV map the flake noise, enable this option. This can be useful for deforming objects or custom mappings.

• Global Seed

Change the **„Global Seed“** if you want to change the distribution of the „flake noise“.

• Global Grid Scale

The **„Global Grid Scale“** scales all the **„Grid Size“** parameters of all the **Flake Layers**.

• Global Flake Brightness

This channel is multiplied with the final outcome of he **Flake Layers**, so you can adjust the brighness of the flakes globally.

• Exclusion Mode
Here you can define which lights should be part of the calculation.

Flake Layer Settings

• Enable Flake Layer X
Enables the **Flake Layer** in question. **Flake Layer 1** is the bottom-most layer, **Flake Layer 4** is the top most layer.

• Grid Size
The **Grid Size** scales the **flake noise** (see **„How are Flakes Distributed?"** for more information).

• SuB Layer Count
Add more layers by increasing the **„Sub Layer Count"** (see **„Not Enough Flakes?"** for more information about when to use it**).**

• Flake Layer Blend Type

The **Flake Layer Blend Type** determines how the current layer will be mixed:

- **Add**: This will add the current layer to the layers before
- **Hierachy**: The shaders lie on-top of each other (if a shader is higher in the hierarchy and has a value, the shaders below won't be evaluated), this is the fastest mode
- **Substract**: Substracts the shaders that lie above the bottom shader, from the bottom shader
- **Multiply**: This will multiply all shaders together

The first layer is always the base and has no **„Flake Layer Blend Type"**. You can disable the first layer, so you can easily just enable a specific **Flake Layer t**o test its settings. In that case, the **„Flake Layer Blend Type"** of the first enabled **Flake Layer** will be ignored (as this is now the new base layer).

• Sub Layer Blend Type
Sub-Layers can be blended in two ways:

- **Hierarchy**: Always the top most particle is visible. This is faster than „**Brightest**" and shows less flakes
- **Brightest**: Always the brightest particle is visible. This is slower than „**Hierachy**" but shows much more flakes

If you use „**Hierarchy**", you have the effect of darker flakes hiding brighter ones. This helps to create more contrast. If you use „**Brightest**", you always see the brightest flake and therefore you don't have darker flakes masking brighter ones. Brightest is a good choice for the **„Flake Layer 1"** to create a dense base layer.

Sparkling

• Sparkle type/Frequency/Cutoff

The Sparkle Type to use (see **„Sparkling Overview“**).

Flake Control

• Flake Type

There are two types available:

- **Ellipsoid** - This will create ellipsoid particles.
- **Cuboid** - This will create „polygon“ shaped particles (all possible variations of „cutting“ a cuboid)

• Flake Density

The **„Flake Density“** defines how much of the **flake noise** you can see.

• Flake Form Variation

The higher you set the **„Flake Form Variation“**, the more the sizes of the flakes will vary,

• Flake Normal

You can choose between the follwoing two options (see **„What does it do?“** for more information)

- **Surface based Flake-Normals**: You can control the maximum angle for the deviation of the flake normal to the surface normal, more control
- **Unique Flake Normal**: Each flake has its own normal, also for

• Flake Normal Deviation

If you use **„Surface based Flake-Normals“**, you can choose the deviation of the flake normal to the surface normal. If you set a value of zero, all flakes are oriented exactly like the surface normal. The higher the value, the more the normals will randomly deviate from the surface normal. Values higher than 90° are allowed and create a different look.

Light Control

• Specular Strength

Defines how strong the specular should be.

• Specular Width

Defines how „wide“ the specular appears.

• Colorize Specular
It is possible to colorize the specular with the shader used in the **„Color Shader“** slot in the **Advanced - Maps** settings.

• Disable Specular Color
If you disable the specular color, only grayscale values will be calculated. For example plastic and glass materials often show just white speculars.

Raytracing - Reflection

• Enable Raytracing
Enables the raytracing to calculate reflections for this **Flake Layer**.

• Reflection Strength
The defines the overall strength of the reflection.

• Use Surface-Normal For Reflection
If this options is enabled, then flakes are all aligned to the surface normal.

• Colorize Reflection
It is possible to colorize the reflections with the shader used in the **„Color Shader“** slot in the **Advanced - Maps** settings.

Fresnel

• Use Fresnel Gradient
Enables the **„Fresnel Gradient“** for this **Flake Layer**. You can also use the **„Fresnel Gradient“** to colorize the flakes, if you chosse colors instead of grayscale values in the gradient. You can use the **„Fresnel Gradient“** to **colorize the flakes by angle**, which is different, as if you use a color in the **„Color Shader“** slot of the **Advanced - Maps** settings.

• Fresnel Gradient
The **„Fresnel Gradient“** defines how light will be reflected, based on the angle of the view-ray and the surface normal.

Advanced - Maps

• Single Value per Flake

If you enable **„Single Value per Flake“**, **all shader slots** will evaluate only **one value per flake**. This makes sure that flakes are not „cut“, if the maps values change within a single flake. Note: the distribution of a 3D volume shader will slightly change if this options is enabled. The internal flake noise will sample a 3D volume shader in a special manner. So if you put the exact same noise in another material channel, they will not match.

• Map Slots

The **„Color Shader“** is the only slot which is not based on a **Flake Layer** Parameter. Every other slot will replace the value of a parameter. See **„Advanced - Maps, Even more Control“** for more information.