Surprising Volumetric Effects in Cinema 4D*

Part 1: The Magic of Visible Lights

by Marc Potocnik, Intel® Software Innovator

Creating Clouds, Fog and Nebulae in 3D Digital Art

Volumetric effects such as clouds, fog, nebulae or fire-like structures are a common task in the daily work of a 3D artist. Technical approaches for Cinema 4D* normally are done with fluid- or particle-simulations with contemporary third-party plugins such as Turbulence FD* or X-Particles*. Additionally, Cinema 4D still hosts the aging voxel-based PyroCluster* system, a former standard tool for a variety of volumetric effects.

In addition to those mentioned, there are other techniques that are more straightforward, but which you might not think of initially: using only onboard tools rather than the approaches mentioned above. Due to their complexity, we will deal with them in two parts. Part 1: The Magic of Visible Lights will cover creating clouds, fog and nebulae and Part 2: Erupting Plasma and Sliced Clouds will address solar flares and puffy clouds.

Thin Clouds and Fog - Visible Volumetric Lights with Built-in Noises
The most simple and straightforward approach to creating basic atmospheric effects is using the built-in light sources of Cinema 4D as actual clouds or fields of fog. With a light source option of creating “foggy“ visible light and some built-in noise functions you can create thin, non-shadow casting clouds or fields of fog in a breeze. But before getting too far into the details, let’s explore some of the important basics of light sources in Cinema 4D. When creating a light in Cinema 4D the corresponding Attribute Manager will show a bunch of parameters and tabs (see figure 1) to customize.

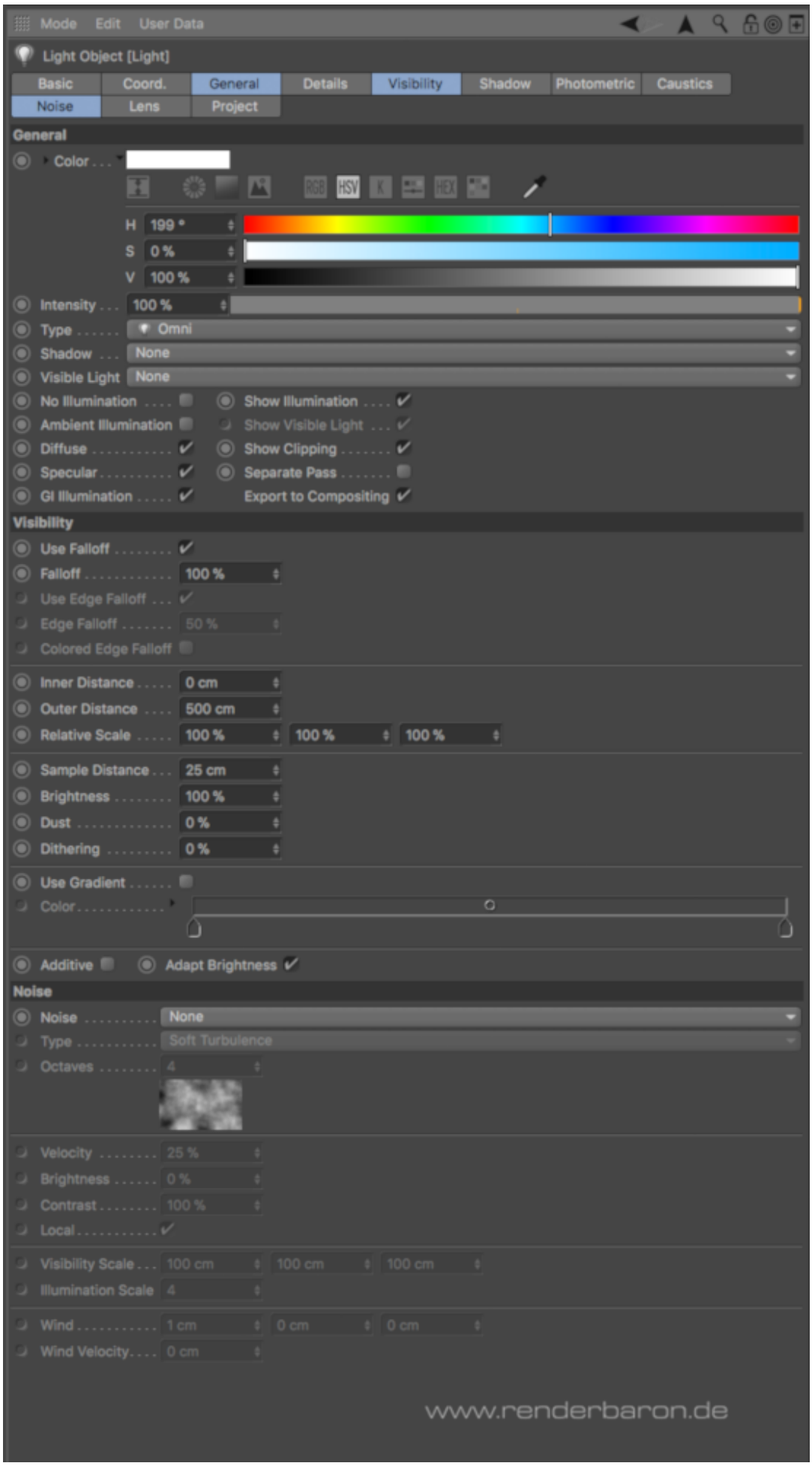

Mode Edit User Data
Light Object [Light]
Basic Coord. General Details Visibility Shadow Photometric Caustics
Noise Lens Project
General
Color
RGB HSV K HEX
H 199 °
S 0 %
V 100 %
Intensity 100 %
Type Omni
Shadow None
Visible Light None
No Illumination
Ambient Illumination
Diffuse
Specular
GI Illumination
Show Illumination
Show Visible Light
Show Clipping
Separate Pass
Export to Compositing
Visibility
Use Falloff
Falloff 100 %
Use Edge Falloff
Edge Falloff 50 %
Colored Edge Falloff
Inner Distance 0 cm
Outer Distance 500 cm
Relative Scale 100 % 100 % 100 %
Sample Distance 25 cm
Brightness 100 %
Dust 0 %
Dithering 0 %
Use Gradient
Color
Additive
Adapt Brightness
Noise
Noise None
Type Soft Turbulence
Octaves 4
Velocity 25 %
Brightness 0 %
Contrast 100 %
Local
Visibility Scale 100 cm 100 cm 100 cm
Illumination Scale 4
Wind 1 cm 0 cm 0 cm
Wind Velocity 0 cm
www.renderbaron.de

Figure 1: Light Object Attribute Manager

General Tab, dropdown menu Type offers a variety of different types of lightsources. Please note: the geometrical shape of your light emission will determine the shape of your visible/ volumetric light.

- Omni:
- concentric light emission
- Spot:
- conical light emission
- Infinite:
- parallel light emission from an infinitely distant point
- Area:
- sample-based spatial light emission with an actual geometric shape
- Square spot:
- pyramid-shaped light emission
- Parallel:
- parallel light emission
- Parallel spot:
- parallel cylindrical light emission
- Square parallel spot:
- parallel cuboid light emission
- IES:
- simulates manufacturer-specific behavior of lighting systems with photometric intensity

Shadows offers a choice of three types of shadow generation:

- **Shadow Maps (soft):**
- creates a "shadow texture" from the point of view of the light source with resolution, blur radius and proximity bias to be selected
- **Ray traced (hard):**
- creates an infinitely hard shadow from the point of view of the light source, by far the oldest and most unrealistic type of shadow
- **Area:**
- the only type of shadow that becomes sharper or more diffuse depending on the distance to the shadow casting object. Useful in combination with Area lights or Infinite light

When creating thin clouds or fields of fog, shadows don't have to be used as the visible structure comes from the light source itself. However, for the creation of thick puffy clouds the use of area shadows will be discussed in Part 2.

Visible Light has three modes to choose from for the actual effect to simulate thin clouds, nebulae or foggy air:

- **Visible:**
- objects in visible light do not cast a volumetric shadow, even if a of Shadow type is activated
- **Volumetric:**
- objects in visible light cast volumetric shadows, even if there is no Shadow type activated
- **Inverse volumetric:**
- visible light is only generated in the volumetric shadow of objects

Visibility Tab

The **Visibility** tab references the **Visible Light** drop-down menu in the **General** tab and offers numerous parameters for defining the decrease of visible light along the light source axis or radius:

- **Edge Falloff:** decreases the visibility along the radius (e.g., of a Parallel Spot)
- **Inner Distance/ Outer Distance:** defines the start- and end-point of the Visible Lights concentrical Falloff (e.g., of an Omni Light)

These Falloff parameters are independently working from the light's intensity Falloff in the **Details** tab - but the handles of these parameters in the editor can easily be confused with each other.

Brightness controls the intensity of visible light. **Dust** adds a black component to the visible light at low brightness values, creating the impression of a covering cloud of dust. **Relative Scale** allows you to squash and stretch the visible light unproportionally and independently from its actual shape.

If the **Volumetric** option is selected in the **General** tab / **Visible Light** dropdown menu, objects in visible light cast volumetric shadows - even if there is no shadow type activated for the light source.

Sample-based Volumetrics

The resolution of this volumetric effect is determined by the **Sample Distance** parameter; its value has to be reduced to increase samples. Please note: a high value (low sample distance) speeds up rendering but may result in sliced artifacts. For a smooth result you will need a lower sample distance which may slow down rendering, see fig. 2 for examples of high sample distance vs. low sample distance.

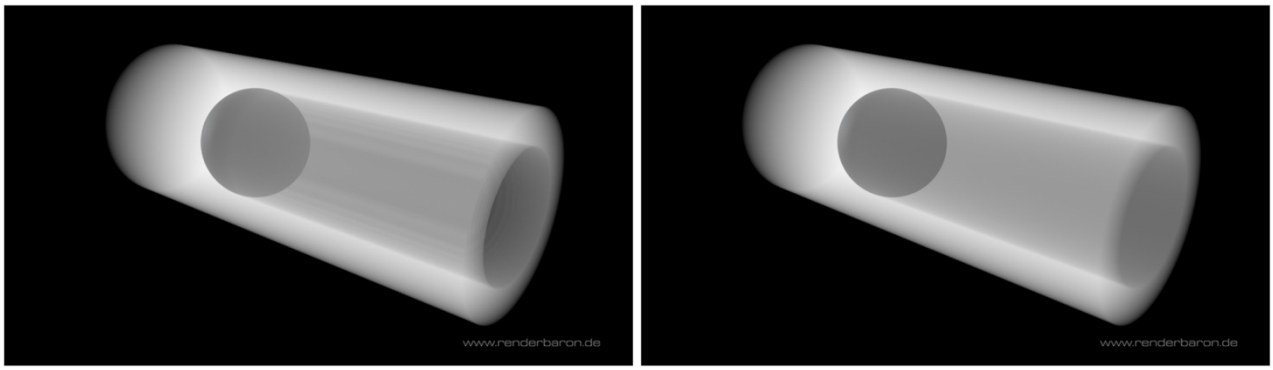


Figure 2: On Left - High Sample Distance, On Right - Low Sample Distance

When combining volumetric light and a complex scene setup (complex objects, hires textures, strong anti-aliasing) things can get exceptionally slow. Always render volumetric lights separately "on black" and then combine it with the scene during compositing.

With these parameters activated and adjusted, begin creating effects such as searching spots, halos, etc. Use the **Noise** tab to define more specific and irregular structures needed to create clouds and nebulae effects.

Noise Tab

The **Noise** tab references **Visible Light** (if selected) and adds random irregularities to it in terms of visibility and lighting. Available noise functions include:

- **NoiseSoft Turbulence**
- **Hard Turbulence**
- **Wavy Turbulence**

These noises can be adjusted in terms of **Octaves** (level of iterations and detail), **Brightness**, **Size** and **Contrast.** Movement can be adjusted by the **Velocity** (idle motion), **Wind** (direction of uniform movement) and **Wind Velocity** (speed) parameters.

The **Local** checkbox determines whether a global or local noise is used. In case of a deactivated checkbox, moving light sources move through a global noise. If activated (by default), moving light sources take noise structures with them. **Visibility Scale** defines the size of the noise itself.

Note: As visible light sources with noise are an older, but still up-to-date, feature of Cinema 4D the four types of Noises are not directly related to the onboard noise shaders.

Case Study – Ditching of US Airways Flight 1549

German TV documentary “Leschs Kosmos”, epsiode “The Nightmare of Flying” for channel ZDF, stages the hidden dangers of aviation. Omni Lights for thin, turbulent clouds in the foreground were used for the sequence on the emergency ditching of US Airways Flight 1549 in the Hudson River.

Omni Lights were set to **Visible** but not to **Volumetric**. The **Illumination** checkbox was deactivated to prevent the clouds from emitting light. The Omnis were slightly flattened by the **Relative Scale** parameter in the **Visibility tab**. In the **Noise tab** a **Hard Turbulence** was applied with strong negative **Brightness** and strong positive **Contrast**, as shown in figure 3.

Figure 3: Screenshot of parameters set to create thin, turbulent clouds in foreground

More information on the making of this TV documentary can be seen detailed in [SIGGRAPH 2015 Rewind: Scientific Eyecandy – VFX for TV-Documentaries with C4D R17]

This personal project is based on the Mountainvista" benchmark scene created for Intel in the spring of 2018. Landscaping creation is completely procedural from rocks, pebbles, roots and dirt to valleys and mountains. Large fields of fog in the distance are crafted with the same simple techniques as described above: large flat Omni Lights with **Visible** light only and a **Hard Turbulence** noise.

Summarizing Creating Visible Volumetric Lights with Built-in Noises
With only Omni Lights, visible light and some high-contrast noise-functions you can create thin, non-shadow casting clouds or fields of fog. This straight-forward technique is absolutely easy to use and is literally right at your fingertips when needed.

The limitation of this approach is that it utilizes just four basic types of noise, which cannot be colored or combined with the more convenient Cinema 4D shaders. The next section on Stellar Nebulae looks at how you can break free of these restrictions with another approach.

Stellar Nebulae - Visible Volumetric Lights as Containers for Volumetric Shaders

A more advanced approach to creating volumetric effects is based on the principle of texturing lights: you can easily apply any material to a light source. If the active **Transparency Channel** contains for example, a 3D noise shader, the volumetric effect of your light source can use the shader for three-dimensional volumetric texturing.

This makes your lightsource an aquarium aquarium for any shader that's being used inside the Materials Transparency Channel. With this in mind, it is easy to create more complex structures such as gas fields or stellar nebulae.

The texture projection method in the texture tag plays only a minor part – as long as you are using three-dimensional shaders such as noise shaders applied in Object- or World-Space or 3D-Gradients you don't have to care about it.

Real-world example: On the left side of figure 4 below, a 2D Tiles Shader is applied onto a volumetric Parallel Spot (no Edge Falloff in **Tab Visibility**) with flat texture projection. On the right side of figure 4, a 3D Noise Shader is applied onto the Parallel Spot. The flat texture projection doesn't have an effect as the noise works three-dimensionally (for more info on this, see Object Space below).

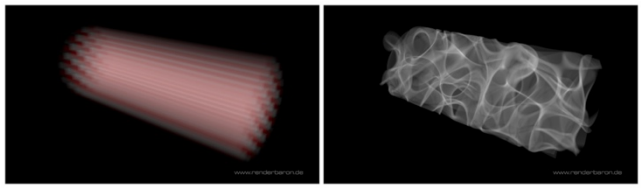


Figure 4: On Left – 2D Tile Shader, On Right – 3D Noise Shader

One of the main limitations to the first approach in this article, was the strict use of only four built-in noise functions. To use the more complex noise shaders in Cinema 4D, such as a 3D-texture for volumetric lights, it is important to get familiar with the basics of noise shaders first.

Noise Shader – Systematic Randomness
Natural phenomena such as clouds, rock or water surfaces contain random structures with different behavior and complexity. In computer graphics, noise shaders can simulate these natural structures. With noises it is possible to give surfaces a random but reproducible irregularity and thus make them appear more analogous and natural.

Noises can actually be used to modulate all material effects. Compared to conventional texture mapping, noises offer the key advantage that they can also be applied three-dimensionally to objects, thus eliminating the need for all questions of a correct texture projection or UVW mapping.

US mathematician **Ken Perlin** was the first to devote himself to the development of a noise shader. In 1981 he was involved in the development of the Disney* classic *TRON* as an employee of MAGI in Elmsford, New York. With his pioneering work on noise shaders Perlin wanted to provide generated objects with a less machine-like and more natural appearance. In 1985 Perlin wrote a SIGGRAPH paper about the resulting **Perlin-Noise**, in 1997 he was awarded an Oscar for his achievements.

Through various further developments, including Steven Worley's Cell-/Voronoi-Noise from 1996, a handful of noise shaders found their way into the Cinema 4D Release R5 in 1998. As part of the shader plug-in, Smells Like Almonds (SLA) developed by David Farmer, and two dozen more complex noise shaders for Cinema 4D were available from 2001 on, with exotic names like Poxo, Luka, Sema or Pezo. Since the Cinema 4D 7.2 Release they have become an integral part of Cinema 4D and are still onboard.

With the classic Material system of Cinema 4D, noise shaders are available as Channel Shaders and can be found in the shader dropdown menu of the Material Editor under **Noise**. In the recent Release 20 of Cinema 4D, noise shaders are also available as Nodes of the new Node-based Material system. They can be found under **Noise** in the Node Editors Asset Manager. As the projects shown in this article were made with Cinema 4D R19 we will focus on the Channel Shader version.

Poxo, Sema, Stupl & Co – A Family of Noises
Cinema 4D's noise shaders have very individual characteristics and are predestined for different purposes. A few examples include:

- **Noise**, as the default choice, strongly resembles a Perlin Gradient Noise: all components appear to be about the same size and brightness, fluctuations are represented by soft gradients. Higher octaves, such as arithmetic iterations, are not available. Its higher octave sibling **Turbulence** can be used to create thin puffy clouds.

  [embed]https://vimeo.com/248867226[/embed]

  Poxo shows irregular structures more clearly and plays to its strengths with large scaling starting from global "1000%" creating the kind of net-like structures known as foam or turbulent clouds. Higher octaves cause details to be added while at the same time burning up bright areas.

  [embed]https://vimeo.com/189844435[/embed]

- **Luka** with its turbulent details reminiscent of mountain ridges and river beds offers a good basis for fine, thin clouds.

  [embed]https://vimeo.com/284331194[/embed]

- **Naki**, on the other hand, with its turbulent blob-like nature and disproportionate strain, is ideal for depicting energetic structures or erupting plasma, or the sun's evolving corona.

  [embed]https://vimeo.com/253405164[/embed]

The noise shaders may be a group of strangers, but they are obviously relatives: **Noise** looks like a **Turbulence** with only one octave, **Turbulence** and **FBM** seem to be siblings, and their smeared versions such as **Wavy Turbulence**, etc. look like brothers and sisters as you can see in figure 5. All of this similarity has one reason: all noises in Cinema 4D are the offspring of Perlin and Voronoi Noise.

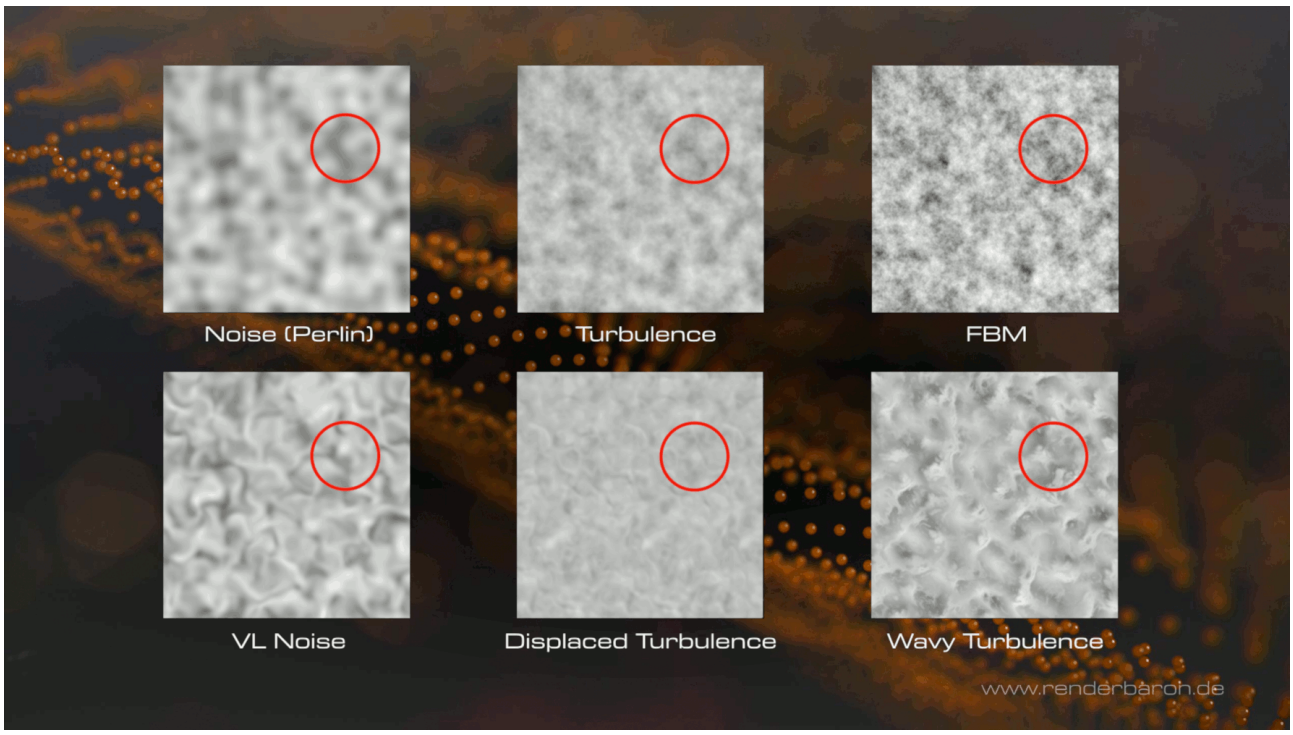


Figure 5: Examples of how similar the different turbulence noises appear

For the selection and characterization of noises, the help section of Cinema 4D offers comprehensive and large-format overview images. An additional description of the shaders can be found in the slightly older Cinema 4D Nnoise Texture Rreference.

Usage of Noise Shaders Whether using noises as channel shaders or as nodes – noises follow a principal function and have several important parameters in common: the type of noise is selected via the drop-down menu **Noise**. The **Octaves** parameter indicates the calculation iterations, followed by **Size**, **Animation speed** and gradation (**Clipping**, **Brightness**, etc.) parameters.

Noises are applied in various reference systems. They can be selected via the drop-down menu **Space**. Four of these reference systems are of practical importance:

- **UV (2D):** projects the noise onto the UV coordinates of the texture and deformations of the object are taken into account
- **Texture:** uses the coordinates specified in the texture tag and deformations of the object are ignored
- **Object:** applies the noise three-dimensionally to the axis system of the object and rotations, movements and scaling of the object are taken into account
- **World:** applies the noise three-dimensionally to the axis system of the world so the noise stays in place while rotations, movements and scaling of the object are ignored

In terms of the creation of volumetric effects, the Spaces **Object** is the most interesting as in this case noises range three-dimensionally through the volume of the objects they are applied to – or the volume of a visible volumetric light source.

To create more complex volumetric structures, it is important to have the possibility to combine, layer and mask different shaders inside the same Material channel. For this purpose, the Cinema 4D Shader toolset contains the Layer Shader.

Layer Shader

In order to combine Cinema 4D shaders in a modular and flexible way, the Layer Shader is key. It acts as a kind of "container" in which layers (shaders or bitmaps) are mixed in a Photoshop* style, masked with others or used in different layer modes. The layer modes include common modes such as **Normal**, **Multiply** and **Screen** but also a **Layer Mask** mode. In this layer mode, the layer above, like a shader or bitmap, is masked with the grayscales of the layer below.

In the Layer Shader, bitmaps can be loaded by clicking on **Image**. Shaders created by clicking on **Shader**. With a right click on a layer, bitmaps and shaders can also be copied (Copy Shader/ Image) and pasted (Insert Shader/ Image).

For better organization, the **Folder** button can be used to create a folder and layers can be inserted into it. In addition, with the **Effect** button manipulations like Colorization, Distortion, Gradation Correction etc. can be created on the layers below.

Tip: If you load a shader/ bitmap in a material channel and then create a layer shader (by clicking on the triangle button on the right-hand side of “Texture”), the texture is automatically contained in the layer shader as a layer.

Masking is an important topic inside a layer shader as it can be used to restrict certain aspects of the shader setup to particular areas of your lights volume.

Gradient Shaders

The **Gradient Shader** creates **2D gradients** (on surfaces) or 3D gradients (though objects or lights volumes). 2D gradients can run linearly along the U or V texture axis or in certain shapes, such as circular, box or star. They can be rotated by angle, broken up by turbulence and frequency with animated irregularities and applied beyond texture tiles by deactivating cyclic.

3D gradients, on the other hand, penetrate the volume of the object or the volumetric light source. Start and end values of 3D gradients refer to the axis system selected in the space drop-down menu, e.g., object or world. Start and end values on several axes can be used to angle such as a linear or cylindrical 3D gradient. 3D gradients are ideal companions to mask 3D noise shaders inside a layer shader.

Real-world example: On the left in figure 6 below, the three-dimensional Sema-Noise is exactly determined by the edges of the Parallel Spot (radius 100cm). On the right in figure 6 a gradient in **3D Cylindrical** mode is used as a layer mask. The gradient is applied with a radius of 150cm along the Z-Axis of the light source and uses large scale turbulence. All of this breaks up the edges of the 3D noise with some irregularity.

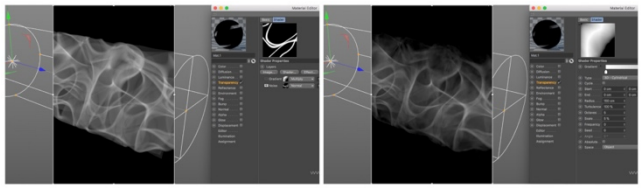
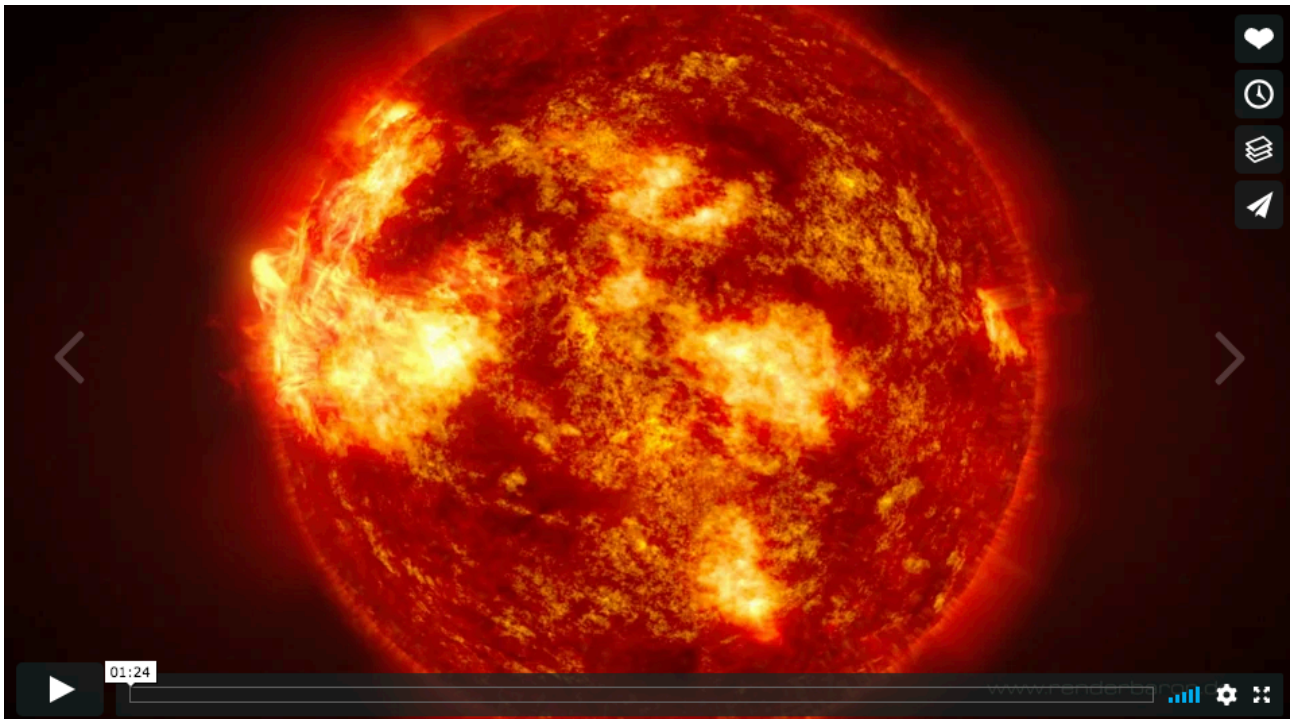


Figure 6: On Left - 3D Sema-Noise, On Right - 3D cylindrical gradient

Case Study: A Question of Time (Stellar Nebulae)

A project for germen TV-documentary “Terra X” of german channel ZDF shows stellar nuclear fusion, the structure of salt and the principle of uranium-lead dating. The stellar nebulae in the sequence about hydrogen and helium atoms used exactly the principle as described above: one volumetric light source as a Light Container**.**



As the camera travels linearly through the visible volumetric light of a parallel spot, a hard turbulence noise roughly shapes the large-scale structure of the light. A material named “Nebulae” is applied to the light source and the visibility is set to 8000% to depict even darker noise details. The material only contains an active transparency channel with a layer shader inside.

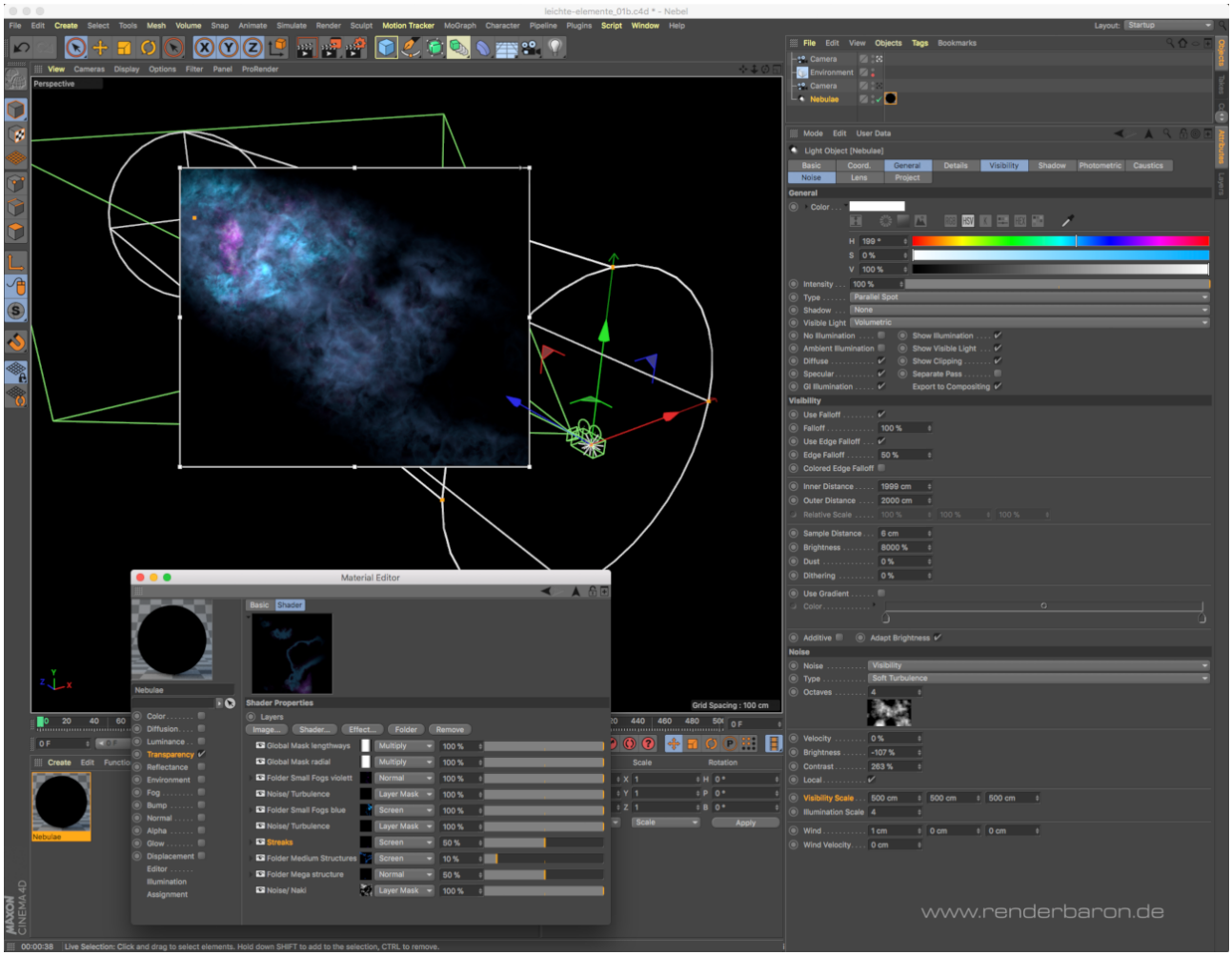


Figure 7: Screenshot displaying settings to create nebulae

Inside the layer shader, 3D noises and 3D gradients are organized in folders related to the logical structure of the stellar nebulae: mega structures, medium structures, small fogs, blue, etc. Folders are treated like normal layers and are masked by noises or gradients in layer mask mode.

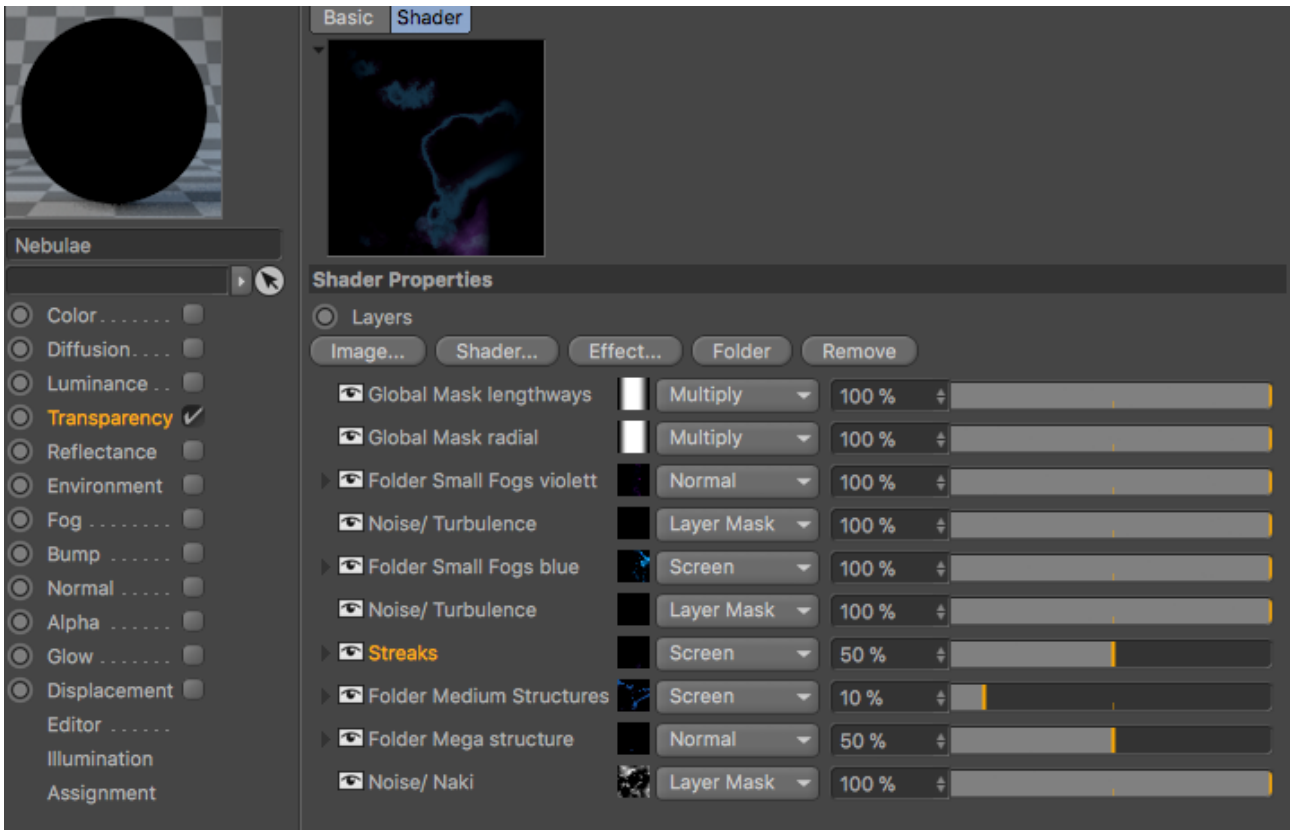


Figure 8: Screenshot of parameters to set to create nebulae

Click on the tiny triangle beside a folder to reveal its content: noises of different style and behavior are colorized by the same-named layer effect and restricted/masked to certain areas of the light source by 3D gradients. For a better understanding, all layers are named accordingly.

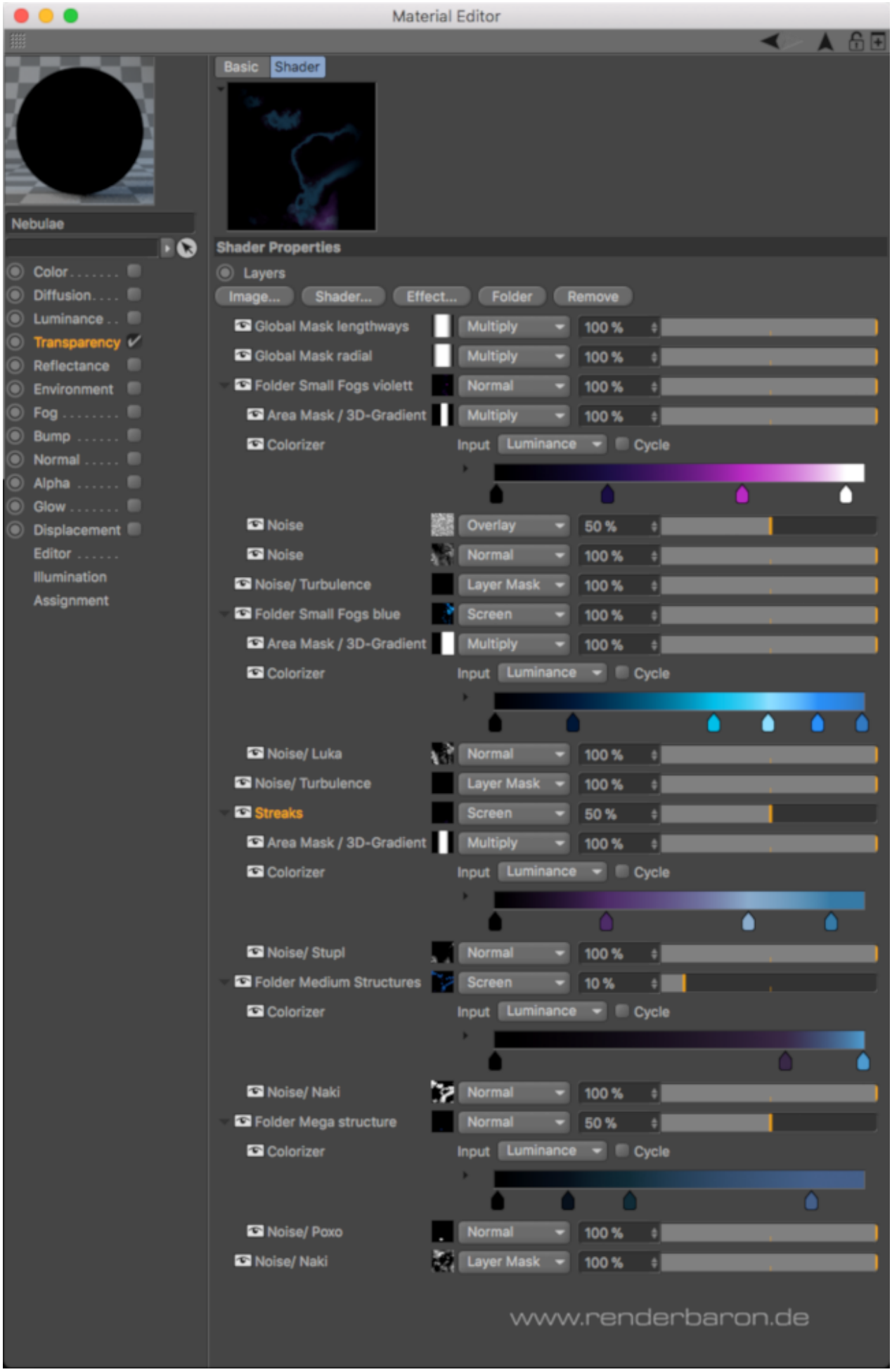

Material Editor
Basic
Shader
Nebulae
Color
Diffusion
Luminance
Transparency
Reflectance
Environment
Fog
Bump
Normal
Alpha
Glow
Displacement
Editor
Illumination
Assignment
Shader Properties
Layers
Image...
Shader...
Effect...
Folder
Remove
Global Mask lengthways
Multiply
100 %
Global Mask radial
Multiply
100 %
Folder Small Fogs violett
Normal
100 %
Area Mask / 3D-Gradient
Multiply
100 %
Colorizer
Input
Luminance
Cycle
Noise
Overlay
50 %
Noise
Normal
100 %
Noise/ Turbulence
Layer Mask
100 %
Folder Small Fogs blue
Screen
100 %
Area Mask / 3D-Gradient
Multiply
100 %
Colorizer
Input
Luminance
Cycle
Noise/ Luka
Normal
100 %
Noise/ Turbulence
Layer Mask
100 %
Streaks
Screen
50 %
Area Mask / 3D-Gradient
Multiply
100 %
Colorizer
Input
Luminance
Cycle
Noise/ Stupl
Normal
100 %
Folder Medium Structures
Screen
10 %
Colorizer
Input
Luminance
Cycle
Noise/ Naki
Normal
100 %
Folder Mega structure
Normal
50 %
Colorizer
Input
Luminance
Cycle
Noise/ Poxo
Normal
100 %
Noise/ Naki
Layer Mask
100 %
www.renderbaron.de

Figure 9: Screenshot displaying the named folders within the layer shader settings

Additionally, the scene contains an environment object which adds a simple but efficient black fog along the camera's Z-axis. This fades out superfluous visual detail in the distance which otherwise would distract the eye.

Conclusion

This article describes two approaches on using visible volumetric lights for creating clouds or complex nebula-like effects. The next article of this series, Surprising Volumetric Effects in Cinema 4D* Part 2: Erupting Plasma and Sliced Clouds, will explore creating an animated sun corona and shadow casting puffy clouds.

More detail on Marc and his work can be found the Intel® Developer Zone, including Mountainvista, SIGGRAPH 2018, and Comparing 3D Rendering Performance Using the Cinema 4D Mountainvista Scene Workload. Also, check out the Cinema 4D Release 20 that was announced during SIGGRAPH 2018. Visit the Intel® Game Dev program to for the latest updates and resources to help you elevate your game development.

Resources

Intel® Core™ i9-7980XE Extreme Edition processor
Cinema 4D

About the Author

Intel® Software Innovator Marc Potocnik is a German designer and founder of animation studio renderbaron (www.renderbaron.de). renderbaron specializes in the production of high-quality 3D animation and visual effects for a wide range of high-profile companies including ZDF, Audi, Siemens, BMW, a.o.

Marc studied communications design at the University of Applied Sciences in Düsseldorf entering the world of 3D with Cinema 4D R4 in 1997. He founded renderbaron in 2001 and went on to become an authority in shading, lighting and rendering with Cinema 4D. Marc is qualified as a Maxon* Lead Instructor and has correspondingly written the Maxon QuickStart Training: Shading, Lighting & Rendering. He also regularly shares his knowledge at conferences around the world including SIGGRAPH, IBC and FMX. Follow along with renderbaron projects on Facebook.