PxrVolume

Moana Cloud rendered with RenderMan - VDB Asset Courtesy of Walt Disney Animation.

PxrVolume is a material that can be used to render volumetric effects which can vary wildly in complexity, including:

- **Constant fog** (a constant **homogeneous** volume with **single scattering**);
- **Smoke** (a **heterogeneous** volume where single scattering suffices);
- **Fire** (a heterogeneous volume with **emissive** properties);
- **Clouds** (a heterogeneous volume where **multiple scattering** is usually required)

PxrVolume is intended to be a basic material that takes as input a small number of volumetric properties. Modeling of complicated volume properties should either be handled by upstream nodes or baked into a volumetric file format such as OpenVDB by a simulation tool.

Users of PxrVolume should be aware of the difference between homogeneous volumes and heterogeneous volumes: homogenous volumes have a constant density and color, whereas heterogeneous volumes have varying density and/or color. Heterogeneous volumes are typically much slower (and noisier) to render than homogeneous volumes.

⚠ PxrVolume is a **dedicated** volume shader, optimized for dealing with "pure" volumes. It cannot deal with any surface effects such as refraction or reflection at a boundary between two different types of media, so you cannot use PxrVolume to render things like murky glass or water surfaces. If you are looking for a shader that can do surface effects, along with a limited amount of volumetric effects, take a look at the single scattering parameters of PxrSurface.

Basic Workflows

Constant fog

Fog effects that are constant (i.e. involving a homogeneous volume) can be easily modeled using just PxrVolume parameters, without any input connections. This provides the most efficient rendering.

For any fog effects that enclose the camera or envelop light sources, we require that PxrVolume be attached to a RiVolume object. Otherwise, PxrVolume can be attached to any closed piece of geometry in order to create simple effects such as a shaped region.

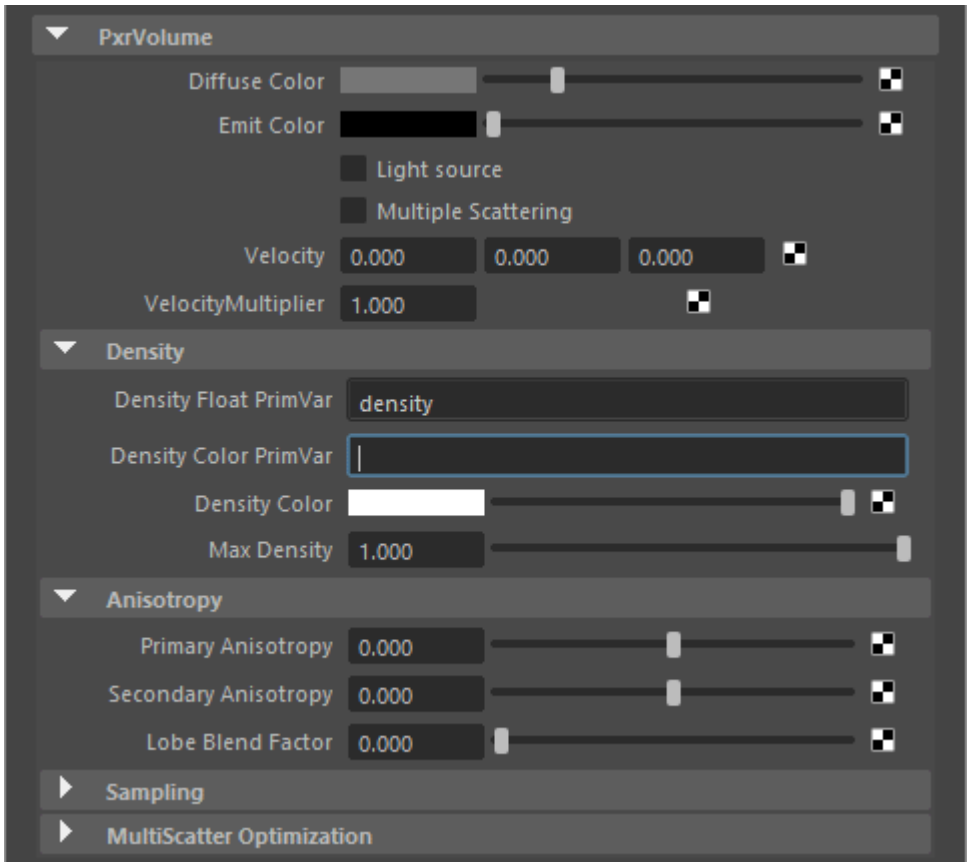
If depth attenuation is the only effect that is required, the **Diffuse Color** can be set to black: light is not scattered by the volume, it is only absorbed, and **Density** is the only parameter that is important. Otherwise, the diffuse color needs to be set to the color of the fog.

Smoke

For heterogeneous media like smoke, the typical approach is to bake the properties of the smoke into a file like an OpenVDB file. The OpenVDB file would contain a small number of properties including the density, the velocity (if using motion blur), and possibly the albedo. The OpenVDB should be attached to a RiVolume and read using the impl_openvdb plugin. The name of the density property of the OpenVDB file is then used directly with the PxrVolume **Density Float PrimVar** parameter.

Smoke tends to have a very low albedo, so a dark color is used for the **Diffuse Color** parameter. (If the color is baked into the OpenVDB file, a PxrPrimVar node can be used to extract the color property from therein.) Because of this low albedo, there is usually very low amounts of light that have scattered multiple times in the volume, so **Multiple Scattering** can be left turned off, and **Samples** can be increased beyond 1 to provide faster rendering.

Since smoke is typically isotropic, the **Anisotropy** settings can be left at their default. The **Equiangular weight** can be left as is if it is expected that there are light sources near to, or within the volume, or set to 0.0 if no light sources are expected to be anywhere close to the volume.

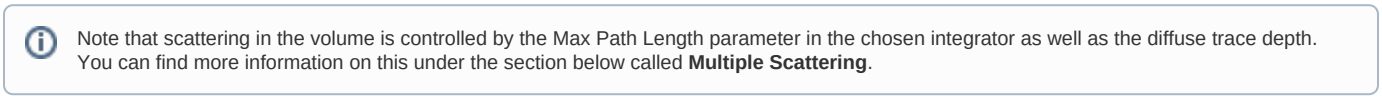

PxrVolume
Diffuse Color
Emit Color
Light source
Multiple Scattering
Velocity
0.000
0.000
0.000
VelocityMultiplier
1.000
Density
Density Float PrimVar
density
Density Color PrimVar
Density Color
Max Density
1.000
Anisotropy
Primary Anisotropy
0.000
Secondary Anisotropy
0.000
Lobe Blend Factor
0.000
Sampling
MultiScatter Optimization

Clouds

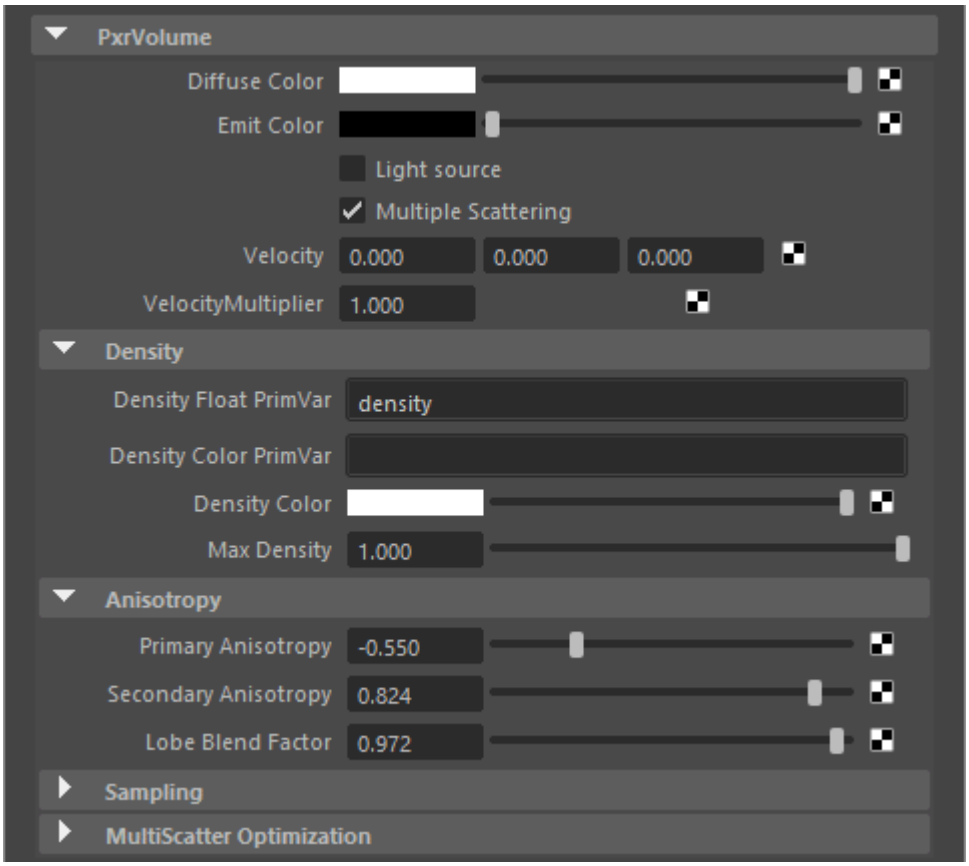
Moana Cloud rendered with RenderMan - VDB Asset Courtesy of Walt Disney Animation.

Clouds are very difficult volumes to render because the aggregate behavior of water droplets and their interaction with light is a complicated phenomenon. Unlike smoke, the light that reaches the eye from a cloud has typically bounced inside the cloud tens or hundreds of times. On top of that, the scattering behavior (the way light bounces inside the cloud) is very complicated, as it involves both forward scattering and backward scattering, with multiple peaks for both.

A typical approach to modeling the cloud itself is to bake the density of the cloud into a file like an OpenVDB file, and the name of the density property of the OpenVDB file would then used directly with the PxrVolume **Density Float PrimVar** parameter. Clouds typically reflect all light, so a simple **Diffuse Color** of (1, 1, 1) is sufficient. In order to capture the complicated light behavior of the cloud itself, **Multiple Scattering** is required, along with setting a **Primary and Secondary Anisotropy.** The settings for anisotropy below reflect a physics-based approximation to the complicated scattering behavior of real clouds. Because multiple scattering is used, the sample parameter is unused and can be set to 1. Finally, if there are no light sources inside the clouds themselves, **Equiangular weight** can be set to 0.0 as equiangular sampling would provide no benefit. (On the other hand, in the case of stormy weather where lights might be placed inside the clouds to simulate lightning flashes, equiangular sampling might prove beneficial.)



Note that scattering in the volume is controlled by the Max Path Length parameter in the chosen integrator as well as the diffuse trace depth. You can find more information on this under the section below called **Multiple Scattering**.



Advanced Workflows

Crepuscular Beams

Crepuscular beams (sometimes called "God rays") are simply the result of light scattering in a volume. When this light is isolated into beams (for example, the few rays of sunlight that escape through a forest canopy) the contrast between the parts of the volume that scatter light and the rest of the volume leads to a striking effect.

In order to efficiently render crepuscular beams, it is important to try to maximize the effect of **direct lighting** - i.e. maximize the volume that can directly trace a path to a light source. If the volume must trace a path to a light source indirectly by going through another surface or material, the convergence of the beams may be very slow and unacceptably noisy. In the image above, the stained glass window on the right is modeled as **"thin glass"**: instead of relying on a secondary material bounce to refract through the glass, shadow rays from the volume that traces towards the light source are unimpeded (do not refract), and rely only on the shadow tinting behavior of the stained glass material. The "thin" parameter of PxrSurface (located under the glass parameters) allows one to achieve exactly this optimization.

Performance Suggestions

Density is the most critical property of a volume. It is a measure of both how much light is absorbed and scattered by the volume. Volume integration may require many billions of density evaluations over the course of a single render, so it is important to ensure that the renderer can evaluate the density as quickly as possible. This is where baking a complicated signal to a volumetric file format such as OpenVDB can be useful, as it means the renderer can simply look up the value from the data. For best performance, we highly recommend using the **PrimVar density parameters** if at all possible, rather than using an input connection, even if that input connection is just a single PxrPrimVar node.

For heterogeneous volumes using the RiVolume primitive, at the beginning of the render, PxrVolume will automatically compute density everywhere inside the volume and use this information over the course of the render to greatly speed up the density sampling. The accuracy of this computation (as well as the accuracy of deformation motion blur) is controlled by the ***dice*** attribute ***micropolygonlength*** . Setting the micropolygonlength to a low number will result in a very accurate density estimation, but may cause the initial computation to take a long time (manifesting as a slow first rendering iteration) and may also cause the renderer to use a lot of memory. Setting this value too high will lead to a less accurate density estimation, which may cause isolated pockets of very high density in the volume to be missed. For a 2K render, we recommend that a setting of 5 or more be used, unless deformation motion blur requires a lower setting. If the camera is passing through or near the volume it might be advisable to change the dicing projection from planar to world or spherical. You may lose some details near the camera but it might avoid creating lots of data near the lens.

Emissive volumes may be very slow to converge, especially if they are the only light source in the scene - it may require many hundreds or even thousands of camera samples to render a noise-free image. Until volumes are supported as full light sources that can be used for direct lighting, we suggest that emissive volumes be used very sparingly.

Multiple Scattering is a very costly effect, both because it means the light is bouncing that many more times in a volume, but also because PxrVolume cannot take more samples itself (the **Samples** parameter is ignored), and is at the mercy of the Integrator settings. Moreover, multiple scattering may slow down the convergence of objects behind the volume. Therefore we suggest that whenever possible, multiple scattering should be avoided. For media with low albedo, multiple scattering adds very little to the final render. For some media like clouds, whose light transport involves many tens or hundreds of light bounces, unfortunately, multiple scattering may be unavoidable.

When using **single scattering**, it is often much more cost-effective to increase the number of **Samples** taken in the volume in order to improve the convergence, rather than increase the number of camera samples.

Equiangular sampling improves the convergence of volumes that are lit by lights that are inside or near the volume but otherwise slow down the volume if the lights are far away. The default value of equiangular sampling (0.5) is a compromise that works acceptably across most scenes. If you know that there are no light sources near the volume, then the equiangular weight should be set to 0.0 in order to speed up the rendering. Equiangular sampling is disabled for multi-scatter volumes in versions 21.5 and later.

> ⚠ Equiangular sampling may cause negative values in the volume alpha with low samples. This is a side effect of how the parameter works. Two solutions would be to set the Equiangular Weight to 0.0 or allow more samples to be taken which will correctly average out to a positive number.

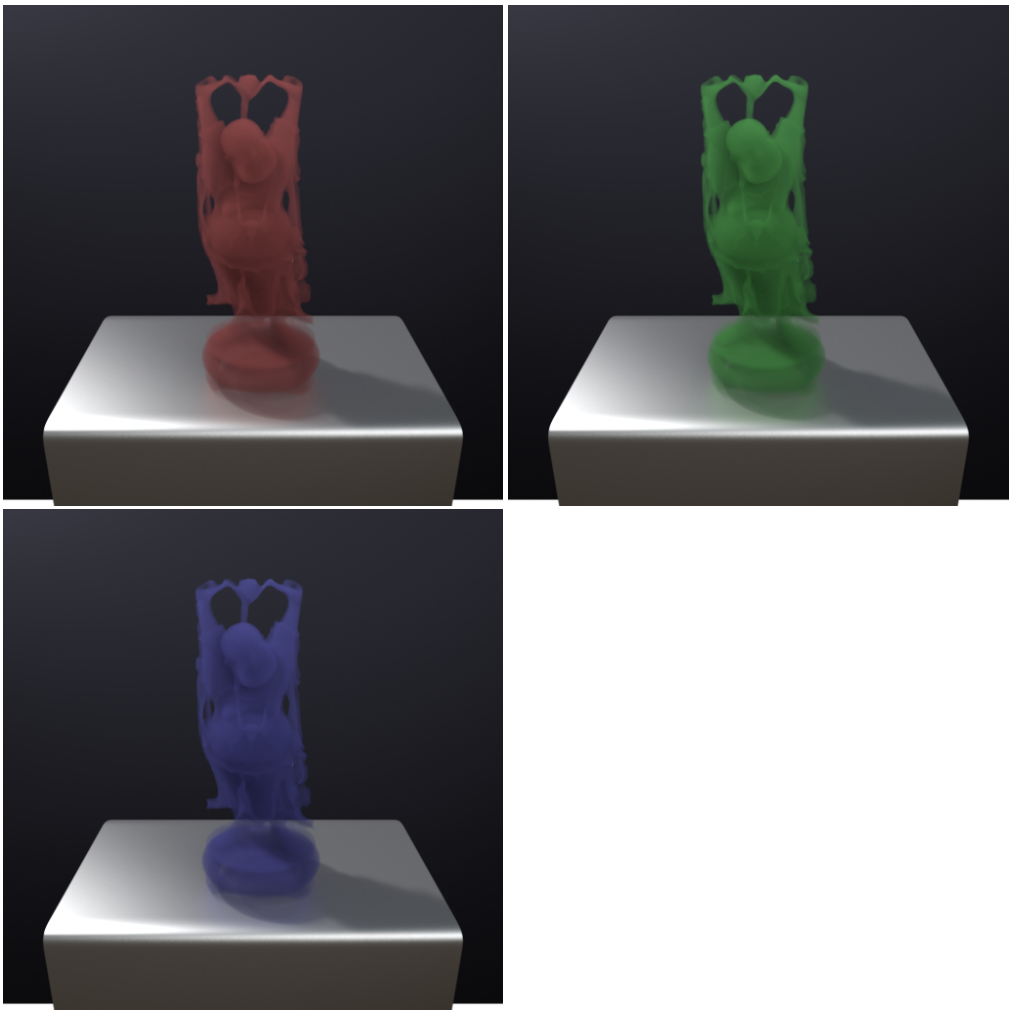
Anisotropic volumes tend to converge slower than isotropic volumes, simply by nature of the sampling that is involved.

Parameters

Diffuse Color

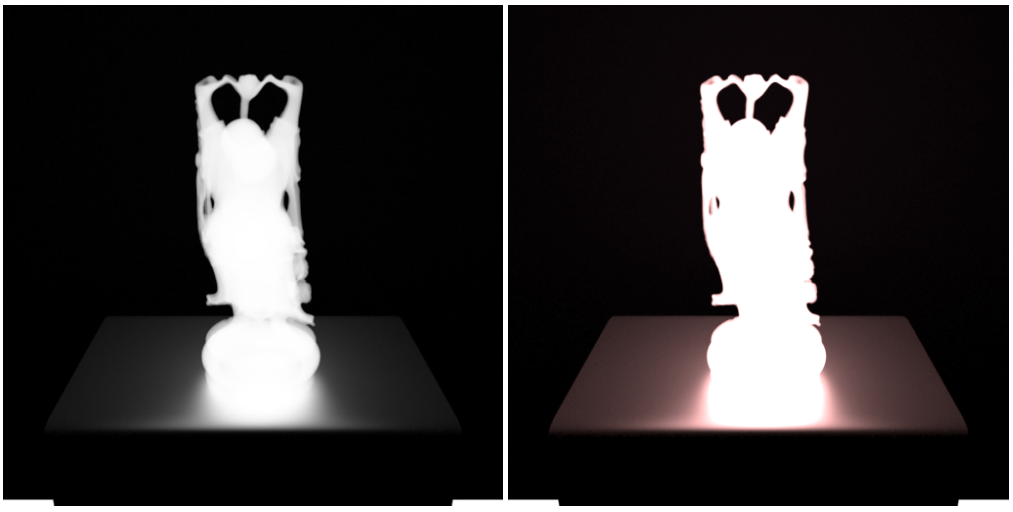
The color of the volume. Default is white (1,1,1). Making a connection on this parameter will create a heterogeneous volume.

If you are familiar with scattering, absorption, and extinction coefficients, note that the diffuse color here is the *scattering albedo*, which is the scattering coefficient divided by the extinction coefficient.



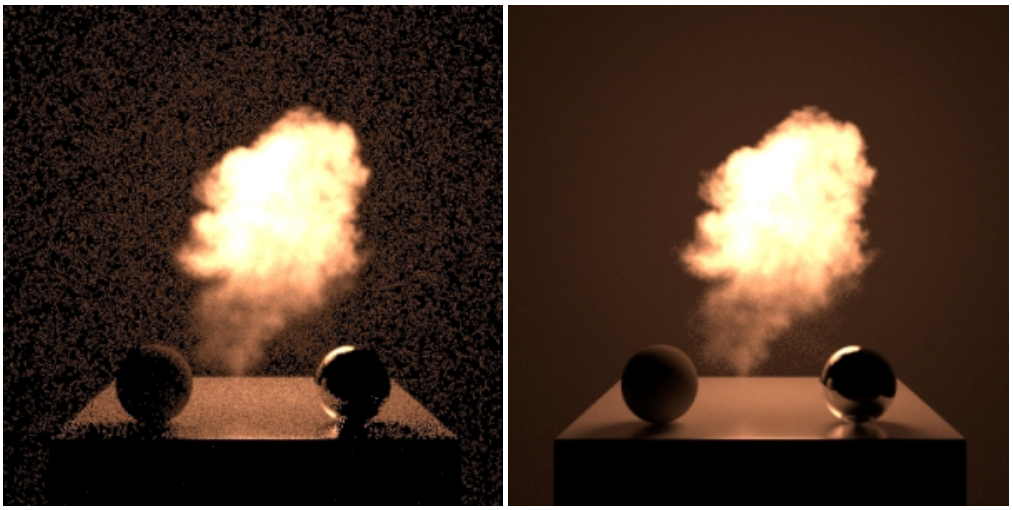
Emit Color

The emissive color of the volume. This is useful for modeling effects such as phosphorescent fog or fire. Default is black (0,0,0), i.e. the volume will not emit light. The following images demonstrate an emit color setting of (1, 1, 1) and (5, 2.5, 2.5), with no other light sources in the scene.



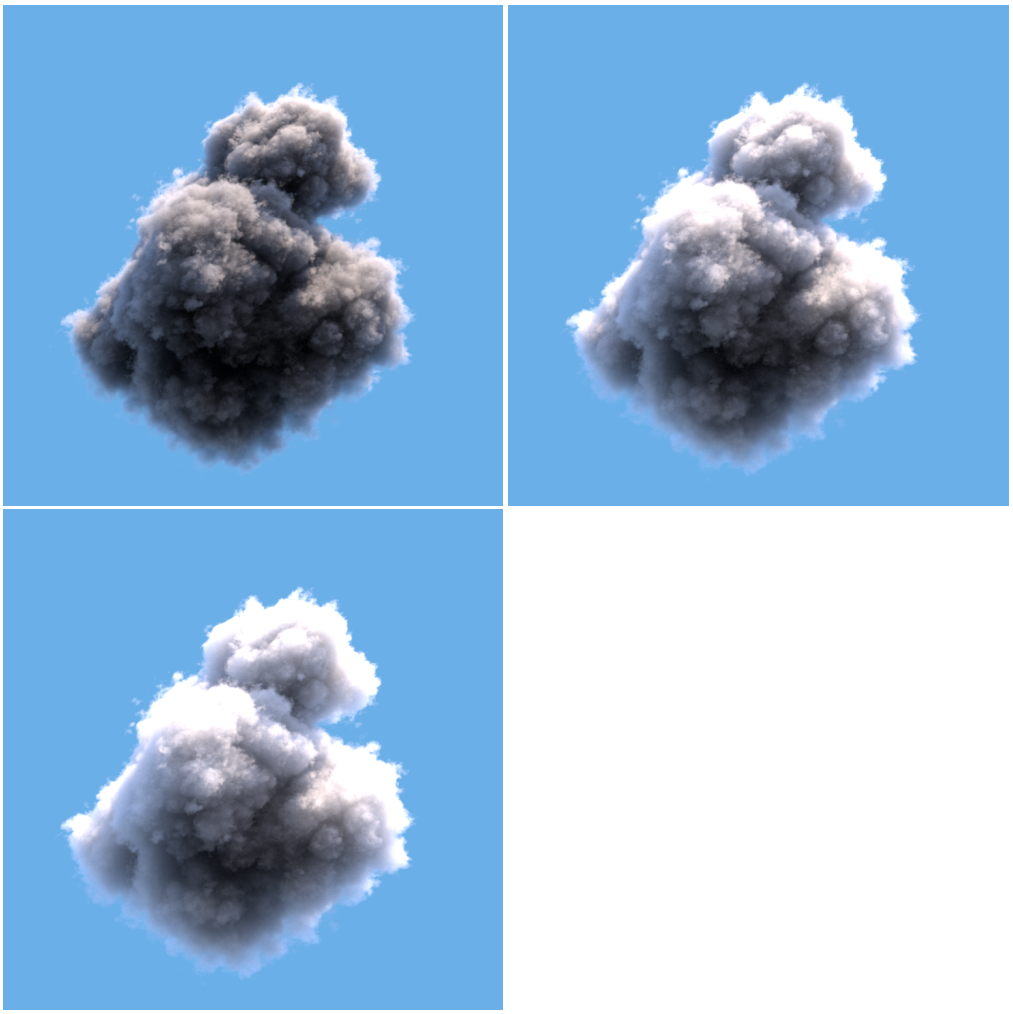
Light Source

When this option is selected, the emissive properties of the volume are sampled as a light source. This may brighten your scene through improved sampling. Note that the emit color cannot be black for this to operate. With a single scatter volume, even at maximum path length 1, it is equivalent to a multiscatter volume with the light source option turned off. Below is off versus on.



Multiple Scattering

This parameter is used as a hint as to whether the volume should compute indirect illumination inside the volume (also known as *multiple scattering*, because light will scatter *more than once inside the volume*). If the multiple scattering parameter is set to 0, and the integrator respects this hint, PxrVolume will only perform *single scattering*: points inside the volume will only be lit directly by light sources. If set to 1, points inside the volume will be lit by indirect illumination as well. The first image below has multiple scattering set to 0, i.e. it is a single scatter volume. The middle image has multiple scattering turned on, with 2 bounces of light. The right image has multiple scattering turned on with 4 bounces of light.



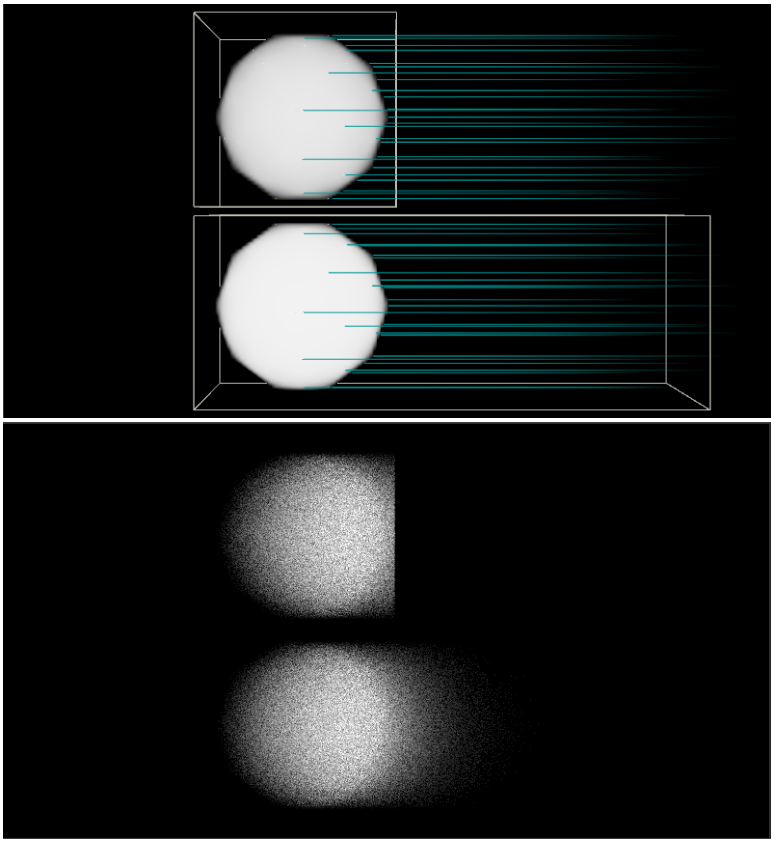
For very dense volumes with high anisotropy, it is often the case that light will scatter many times inside the volume before reaching the eye, and multiple scattering is the only way to achieve the correct look. It is also often the only way to correctly render certain effects such as volume caustics. On the other hand, multiple scattering can be very expensive, which is why the parameter defaults to 0 (off). Moreover, when rendering homogeneous volumes, turning off multiple scattering may result in less noise for directly lit *surfaces* behind the volume.

Note that due to the nature of their algorithm, some integrators (in particular, bidirectional path tracers such as PxrVCM) have no choice but to implement multiple scattering of volumes, and therefore will ignore this hint.

Velocity and Motion Blur

Unlike all other geometry types in RenderMan, **deformation motion-blurred volumes are enabled primarily using Bxdf controls**. Enabling deformation motion blur with PxrVolume is straightforward: specify a value for the velocity parameter, which is vector-valued. Under the hood, the renderer will automatically use the velocity vector to generate temporal varying data for all varying inputs. This preprocessing step occurs at the beginning of the render, and will increase the time to the first pixel; however, it is necessary in order to greatly increase the efficiency of rendering blurred volumes.

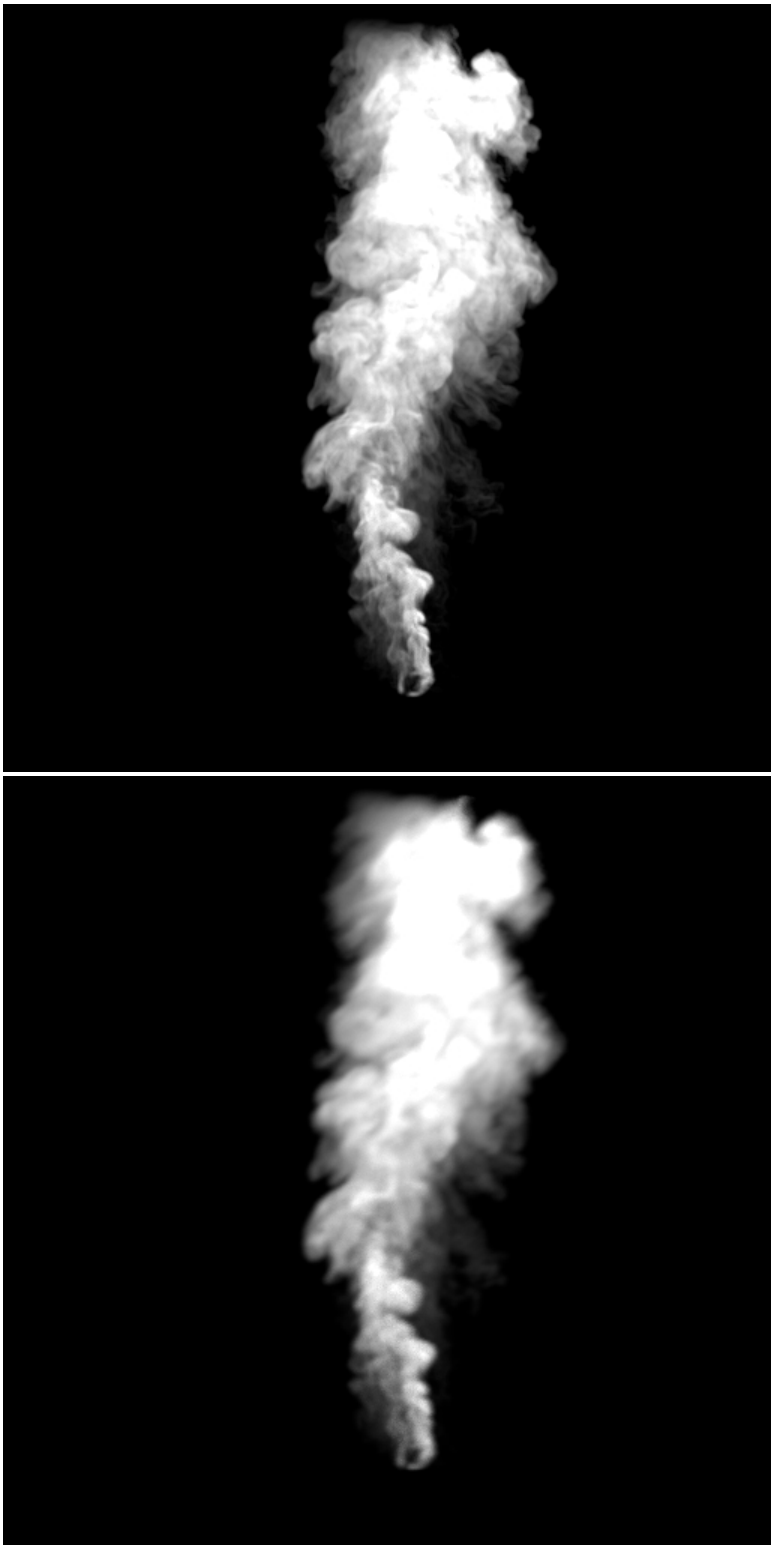
The renderer supports two techniques for deformation motion-blurred volumes. The "Eulerian" technique, supported as of 23.5 (and the default since that version), has low preprocessing time, very low memory overhead, has a low bias (assuming that the micropolygon length is set to a reasonable value), and is reasonably accurate. The downside of this technique is that the velocity values **must be non-zero and supplied everywhere in the path of the motion** , even if the volume has no "presence" in that path at the current baked shutter time. This is because the Eulerian technique assumes that the time-varying property of a volume (e.g. the time-varying density) can be approximated by using the velocity at the current time, backtracking using that velocity value, and looking up the density at the backtracked location. The following images from Houdini demonstrate these boundary velocity requirements for Eulerian motion blur. The 1st image shows two spheres with density and wireframe boxes representing the velocity field bounds. The green lines show the intended motion of the volume. If the velocity field is not fully written out over the range of motion, Eulerian motion blur will clip the velocity as shown on the right rendered images. Note that in some highly chaotic fluid scenes, even if this requirement is not satisfied, the rendered image may still look satisfactory (the clipping may not be obvious).



The "Reves" technique, which was the algorithm used for deformation motion blurred volumes prior to 23.5, is more accurate than the Eulerian technique and has a faster convergence time after the renderer has started. Unfortunately, it has a very high preprocessing cost, very high memory requirements, and is inherently a biased technique. As such, it requires the micropolygon length to be set to low numbers (exacerbating both the time and memory costs) in order to avoid visible biasing which manifest as blurry volumes. In most cases, the Eulerian technique should be preferred unless the velocity boundary conditions mentioned above cannot be satisfied for some reason.

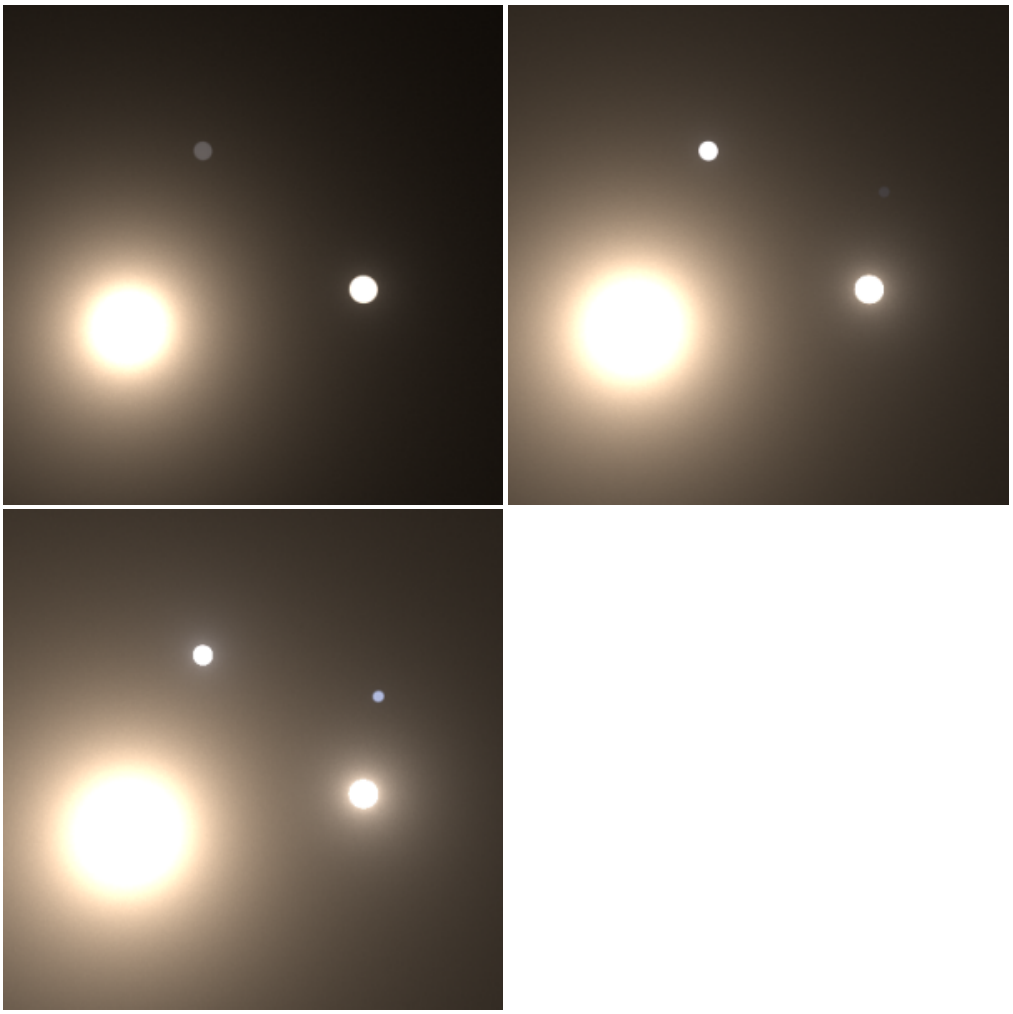
The velocity vector value is expected to be relative to the entire shutter range. If you are working with velocity data measured in units per second and the shutter is not set to an entire second's worth of time, you may need to scale the data by the number of frames per second (i.e. 1/24.0) for a correct picture. We supply a control as a Velocity Multiplier to aid in conversion should your data be in the wrong measurement or allow you to make artistic tweaks to the motion blur. Alternatively, the Attribute "volume" "fps" may be set on the geometry directly to accomplish the same thing; in this case, the value of this attribute would be number of frames per second directly (24), not the inverse.

The following images were rendered using Eulerian motion blur, in conjunction with an OpenVDB file containing a velocity grid. The image on the left was rendered with no blur, while the image on the right has velocity supplied by a connection to the PxrPrimVar node reading a velocity grid from the file. The velocity has been exaggerated by a factor of 10. Note that this OpenVDB file (from openvdb.org) does not satisfy the boundary requirements of having velocity data in all areas of movement - this can be seen in the fact that the silhouette edges of the volume do not change significantly despite the exaggerated movement. However, there are no obvious objectionable artifacts, and the area of the volume away from the boundary conveys the desired velocity detail.



Extinction Distance

This is the distance (in scene units) at which a homogeneous volume becomes opaque. If this is non-zero, it overrides the density connections, maintaining a homogeneous volume. This is great for creating fog/mist that fills a scene. Below we set the parameter to a low value and then increase it until the last light in the scene is clearly visible.



Density Float PrimVar

A primvar from the geometry which overrides the density float of the volume, unset by default. Setting this overrides the densityFloat input, and is more efficient than simply using a PxrPrimVar pattern connection.

For workflows that use a baked volumetric representation (such as an OpenVDB file) we highly recommend the use of this parameter setting over the use of a noise pattern connected to the Density Float parameter.

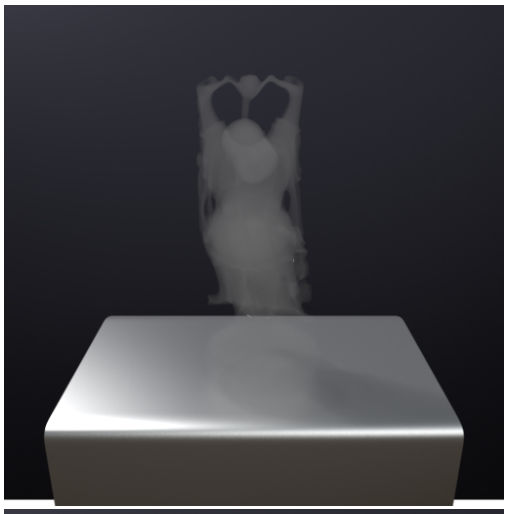
Density Float

The density of the volume directly controls how light is attenuated by the volume - i.e. it directly affects how the volume casts shadows. The density float parameter is overridden by the density color if the density color is changed from its default value. Unless you require colored shadows, you should prefer to set this parameter rather than the density color parameter as the volume will render more efficiently.

Making an input connection on this parameter will create a heterogeneous volume.

The density is also known as the extinction coefficient (the sum of the absorption and scattering coefficients).

Below, from left to right: density of 0.1, 0.25, and 2.0.



Density Color PrimVar

A primvar from the geometry which overrides the density color of the volume, unset by default. Setting this overrides the densityColor input, and is more efficient than simply using a PxrPrimVar pattern connection.

For workflows that use a baked volumetric representation (such as an OpenVDB file) we highly recommend the use of this parameter setting over the use of an input connected to the Density Color parameter.

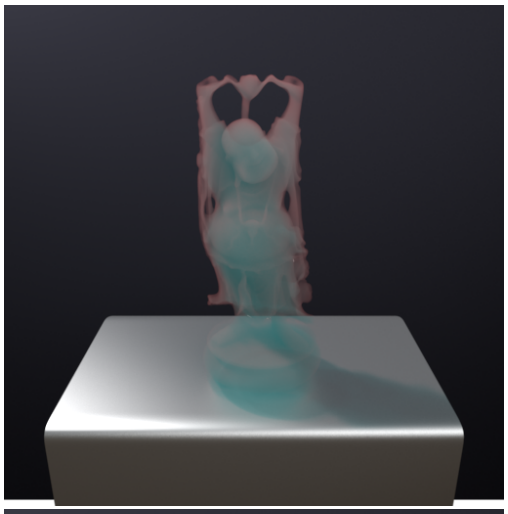
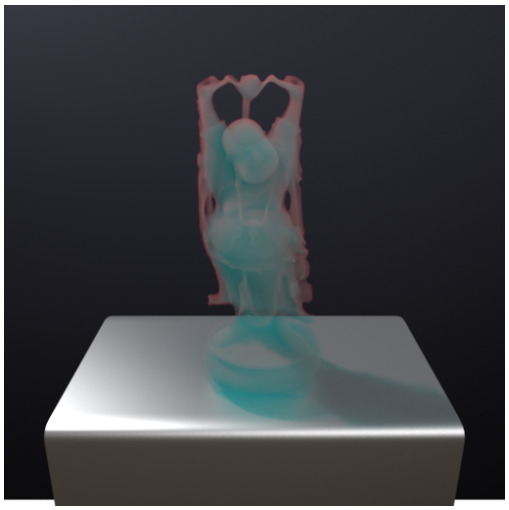
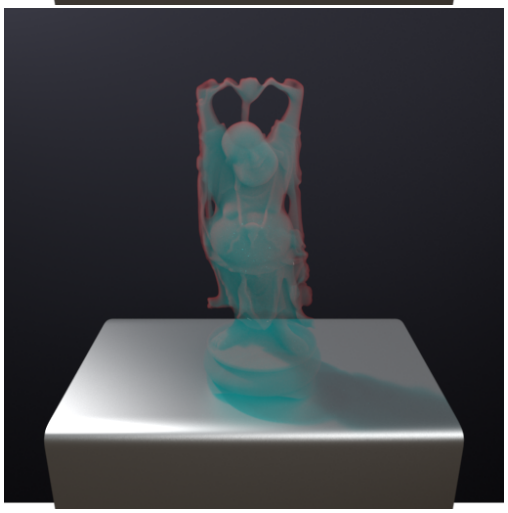
Density Color

The density of the volume directly controls how light is attenuated by the volume - i.e. it directly affects how the volume casts shadows. The density color parameter overrides the density float if the density color is changed from its default value. If you do not need colored shadows from the volume, then you should set the density float parameter and leave the density color parameter untouched; the volume will render more efficiently.

Making an input connection on this parameter will create a heterogeneous volume.

The density is also known as the extinction coefficient (the sum of the absorption and scattering coefficients).

Below, from left to right: density color of (1.0, 0.25, 0.25), (2.0, 0.25, 0.25) and (5.0, 0.25, 0.25). Note that since the density of the volume is higher in the red channel, more red light is scattered away, leaving behind a cyan contribution.

Max Density

The max density parameter is only used by heterogeneous volumes and controls the step size used to sample the volume. For correctly unbiased rendering, the max density must be higher than any density encountered inside the volume. A high value of max density may result in slower renders since more steps will be taken to sample the volume. Setting the max density too small will speed up your render, but will also lead to incorrect (biased) rendering as dense regions of the volume will be undersampled.

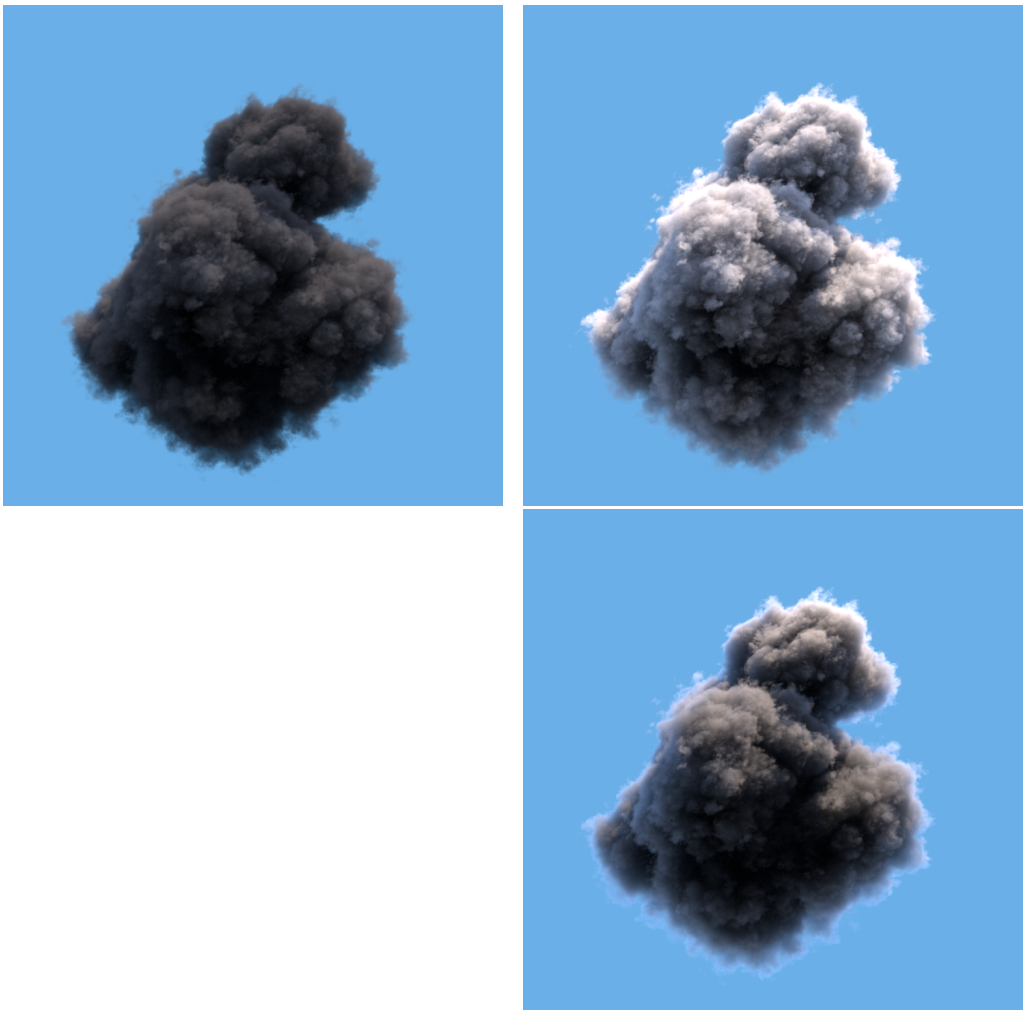
Starting in version 20.7, for RiVolumes only, PxrVolume will also automatically compute the density bounds over the subsections of the volume, and use this information to vary the step size over the ray. This means that more steps will only be taken in denser parts of the volume. Therefore, for very heterogeneous volumes, setting a higher value for max density will no longer have a detrimental effect over the entire volume; in practice, it is often possible to simply set this to a very high value and not worry about this parameter any further.

The default value of max density is 1.0.
Primary Anisotropy

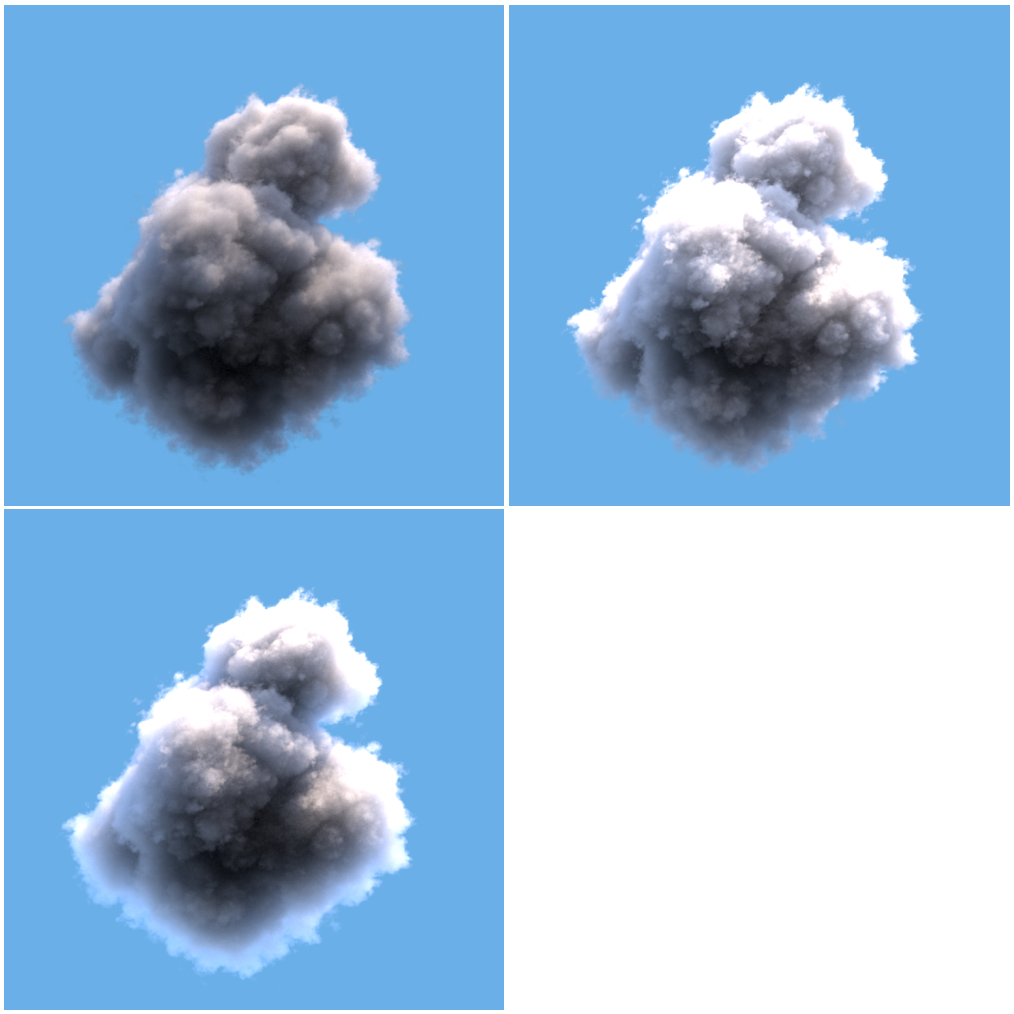
Controls the direction in which the volume scatters light. The anisotropy parameter has a range of -1 to 1, with the default being 0. A value of 0 means the volume is isotropic: light is scattered in all directions with equal probability. A positive value greater than 0 means the volume is forward scattering: incoming light has a higher chance of being scattered in the same direction (away from the incoming light). A value of anisotropy less than 0 means the volume is backward scattering: incoming light has a higher chance of being scattered in the reverse direction (back towards the direction of incoming light).

Many volumes (such as smoke) are generally isotropic, but others can be highly isotropic. For example, organic materials like flesh and skin tend to have highly forward scattering characteristics. Note that the effects of high anisotropy diminish once multiple scattering is involved, as such materials tend to converge upon isotropic behavior after many bounces of light.

Below are examples at -0.8, 0.0, and 0.8 anisotropy for a single scattering volume.



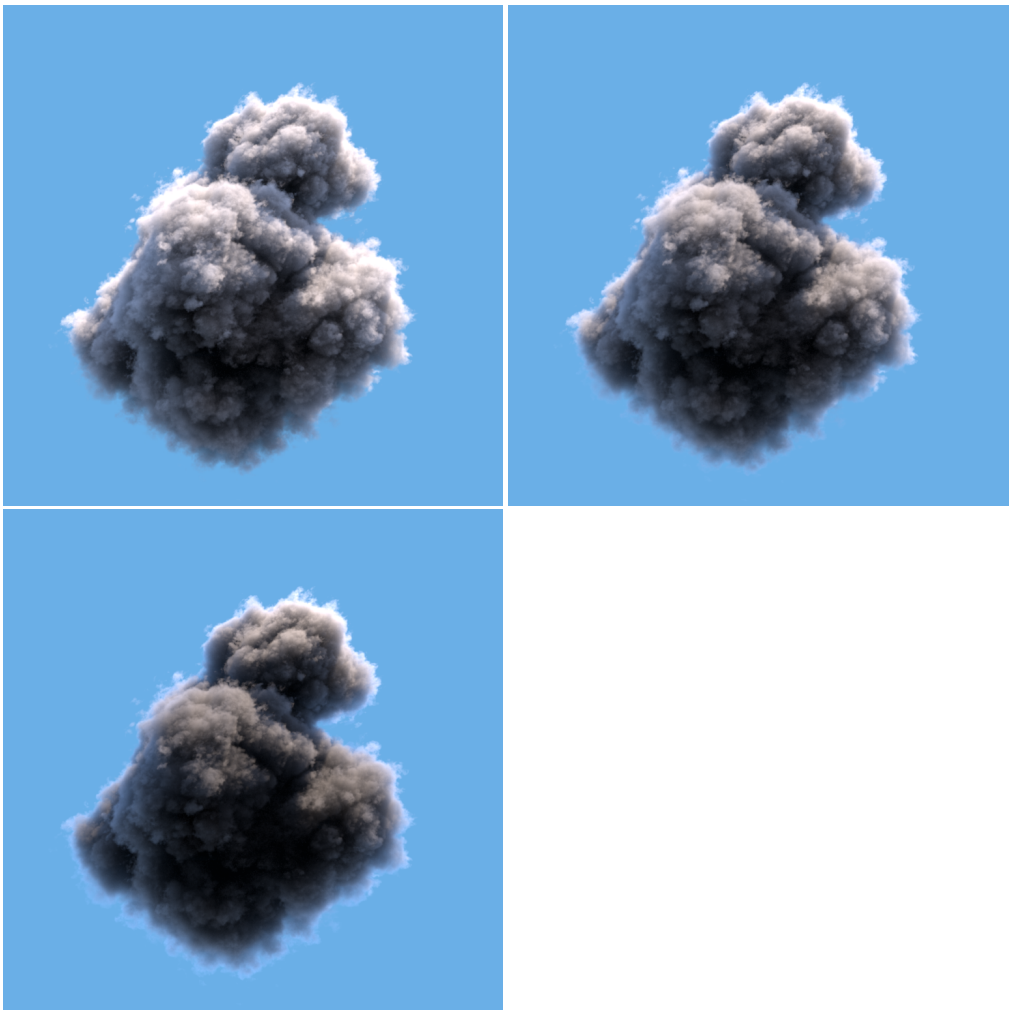
Here are more examples at -0.8, 0.0, and 0.8 anisotropy for a multiple scattering volume.



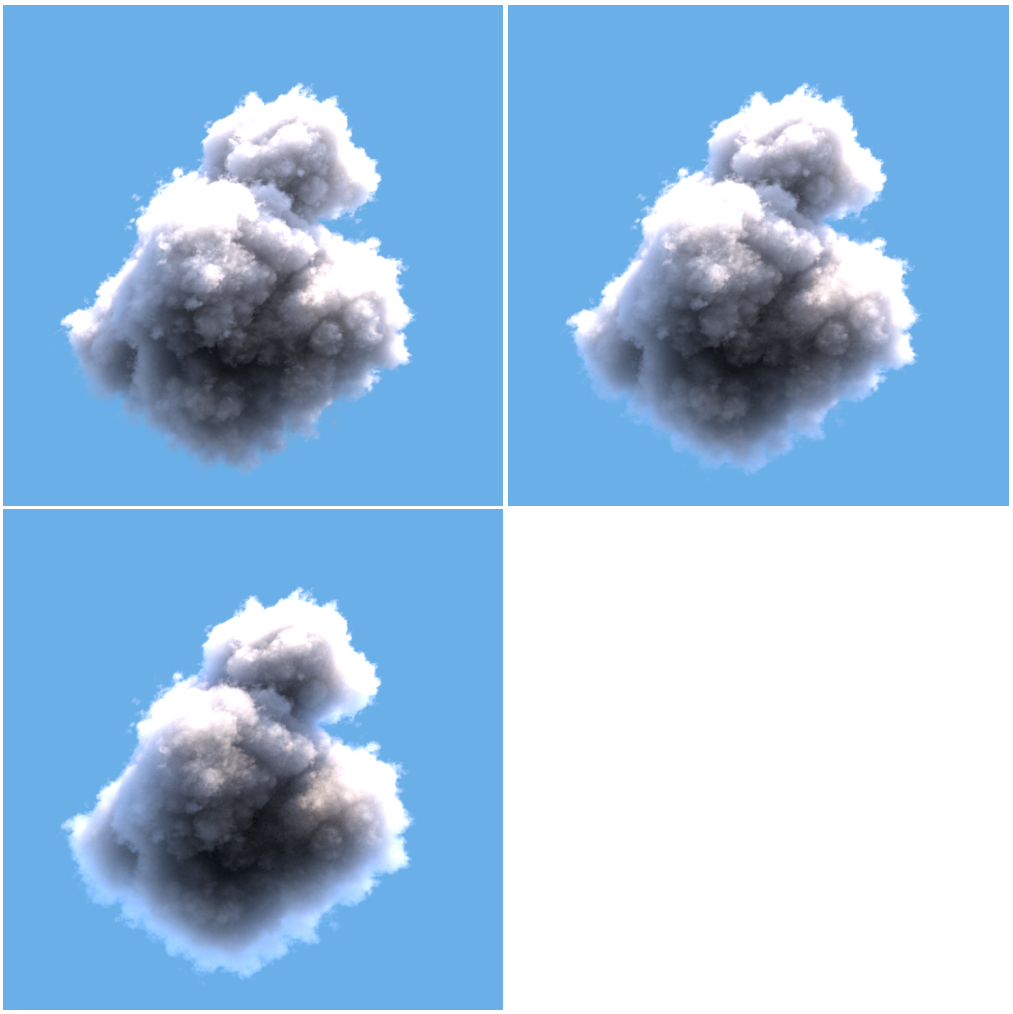
Secondary Anisotropy and Lobe Blend Factor

Some volumetric materials have scattering characteristics that cannot be captured by a single lobe of anisotropy. For example, clouds have a very complicated scattering behavior: they are mostly highly forward scattering, but still have a substantial backwards scattering component, which is largely responsible for the "silver lining" seen on otherwise dark clouds. In order to render such materials, a second anisotropy lobe can be enabled on the volume, along with a blend factor which determines how to blend between the two separate lobes. A blend factor of 0 (default) means that only the primary anisotropy will be considered. A blend factor of 1 means that only the secondary anisotropy will be used, and a blend factor of 0.5 means that each lobe will be considered equally.

From left to right: lobe blend factor set to 0, 0.5, and 1.0 for a single scattering volume with primary anisotropy = 0.0 and secondary anisotropy = 0.8. Note that the first and third images are essentially rendered using single anisotropic lobes.



From left to right: lobe blend factor set to 0, 0.5, and 1.0 for a multiple scattering volume with primary anisotropy = 0.0 and secondary anisotropy = 0.8. Note that the first and third images are essentially rendered using single anisotropic lobes.



Equiangular Weight

Sets the probability of equiangular sampling being used instead of density sampling. Equiangular sampling improves the convergence of volumes close to light sources, while density sampling can be a better technique when dealing with volumes with dense or highly varying density. The default value of 0.5 means both techniques will be used equally and combined with multiple importance sampling. If the volumes are dense and far away from light sources then decreasing the equiangular weight may result in better convergence. This does not apply to multi-scatter volumes.

min/max Samples

This setting controls sample quality for single scatter volumes. A higher number of volume samples will improve the convergence of volumes at the cost of render time. However, the trade-off in scenes with other objects is that the volume will sample more without increasing costs elsewhere in the image and may improve your results. Note convergence rates vary within volumes as well, and this means some regions can converge reasonably well with a small number of volume samples, but others might need more help.

- minSamples enforces a lower limit to sample, only valid with Single Scattering
- maxSamples are the maximum limit per ray inside the Volume

Multiscatter Optimization (Valid for PxrPathTracer)

On/Off, the default is off.

Extinction Multiplier

This parameter is used during *Multiscatter* volume rendering. The value must be above 0.0 and defaults to 1.0 which is no modification. Values below 1 reduce the extinction coefficients while values above 1.0 increase them.

Contribution Multiplier

This multiplier is used during *Multiscatter* volume rendering. This parameter multiplies the direct lighting contribution for multi scatter events. Values must be above 0.0 and defaults to 1.0 which is no modification. A value below 1.0 reduces the contribution of direct lighting while values above 1.0 increase them (this may make volumes appear softer and more translucent.)
The above parameter for "Light Source" is useful for many applications when using a volume with emission properties. But there may be times you need to control the influence of the light from the volume. This requires the use of a light filter. The PxrVolume material doesn't have a way to use a light filter so you can apply the PxrMeshlight material to the volume object along with the PxrVolume for shading.

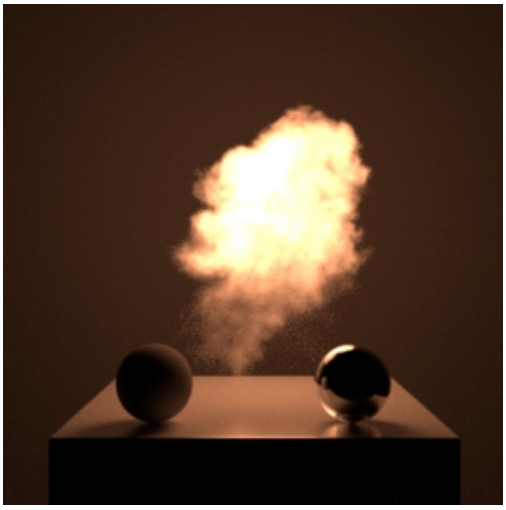
In doing so this will light the scene but allow you to connect and use a light filter through the mesh light controls. But there is a caveat to this workflow that requires extra consideration in your setup.

When you apply the mesh light to the volume you may notice an increase in illumination and/or additional noise. This is because of the double contribution of the volume emission and the mesh light itself. Here are the current steps to reduce this issue.

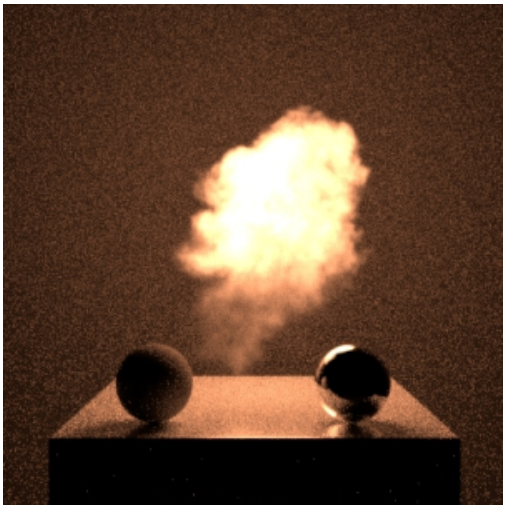
> ⚠ If you do not need a light filter then this work is unnecessary, simple use the "Light Source" parameter.

1. Duplicate your original volume
    a. On the duplicate, assign a duplicate PxrVolume with the correct pattern connections to the original patterns you may be using to drive emission, density, etc
    b. Assign a PxrMeshlight to the volume using the DCC application options (SG Node, Material Builder, etc)
    c. Hide this volume from indirect visibility, we only want the light contribution.
2. In the original volume, disconnect the emission, this will now be lit by the duplicate volume's mesh light source. This prevents double light contribution
3. Apply a light filter as needed to the duplicate volume with meshlight

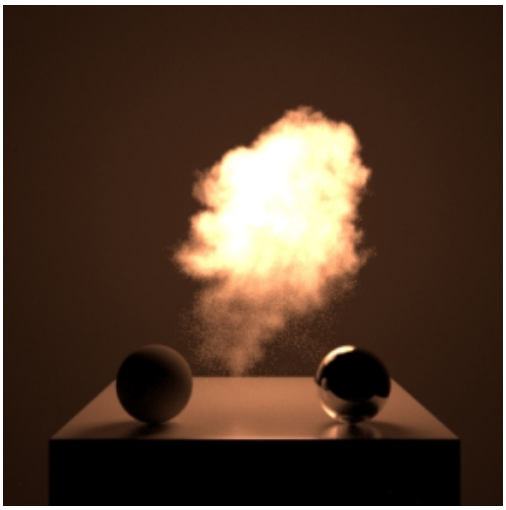
Examples:



Using Light Source



Adding the mesh light increases light in the scene and noise, this is the *incorrect* workflow



This image is using the *above workflow* to create a well-converged render while allowing the use of a light filter

PxrVolume also writes to the U2 user lobe as "DiffuseAlbedo"