Volumes

The rendering of effects like fog, smoke, clouds, cloudy water, and even glass is complicated because light no longer just interacts with the surface of a material; it can be both attenuated and scattered inside and within the **participating media**. **Volume rendering** is required to handle the complex light transport.

When dealing with volumetric effects that are **homogeneous** (the participating media is unchanging inside the volume, i.e. unclouded glass or water), a dedicated volume shader like PxrVolume or a surface shader that supports some volumetric effects like PxrSurface can be used on closed geometry of any type - for example a subdivision mesh. However, if the effects are **heterogeneous** (like fog or clouds), then a special type of geometry is typically needed in order to provide values that change over a three dimensional domain. RenderMan provides a volume primitive (RiVolume) which supports this need.

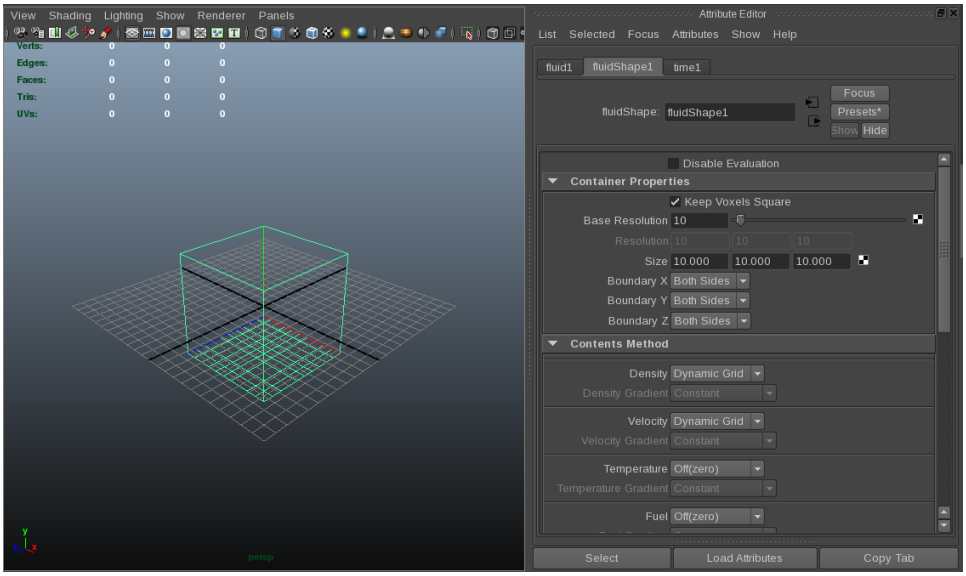
The volume primitive is simply a rectangular axis-aligned box which can return a value for an arbitrary variable at a three dimension coordinate inside the box. This differs from a parametric surface, which can typically only respond to queries on the surface parameterized by two dimensional coordinates. For volume rendering of participating media such as smoke, the most typically used variable is the density of the volume. For more complicated situations like fire, densities like fuel, temperature, or velocity may be part of the variables associated with the volume primitive.

Volume Description

There are two descriptions of the volume primitive typically used with RenderMan.

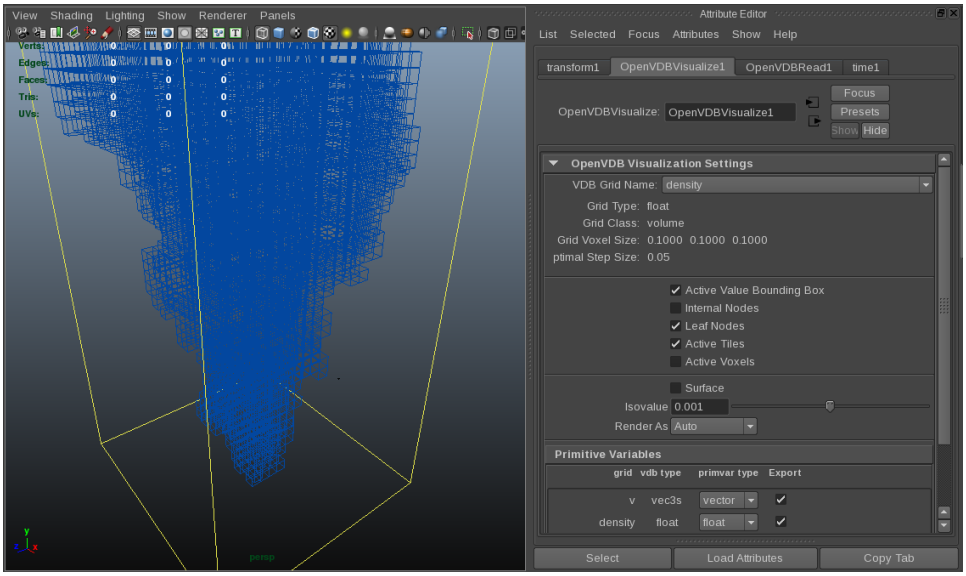
- The simplest is a box subdivided in three dimensions into a regular lattice of voxels, with values provided for arbitrary variables at every corner of the lattice. A lookup of a variable within the box perform trilinear or tricubic interpolation of the nearest lattice points. This representation, while simple, may lead to unwieldy RIB files for dense volumes.

  This approach is used by RenderMan for Maya to render Maya fluids. In the Maya user interface shown below, the bounds of the box are directly manipulated in the viewer on the left. In the attribute editor, the resolution describes the number of subdivisions of the box, which ultimately affect the accuracy of the volume simulation and rendering; in this case, the box is divided into 10x10x10 voxels. The variables associated with the volume are described in the Contents Method pulldown; here, the variables "Density" and "Velocity" are enabled, while "Temperature" and "Fuel" are disabled.



- In situations where the volume is already described via an external resource such as an **OpenVDB** file on disk, the volume primitive supports arbitrary C++ **volume field plugins** which can interface directly with the external resource in order to compute the value of an arbitrary variable within the extents of the box. RenderMan provides a plugin called impl_openvdb which can use VDB files directly. In this case, the volume description simply involves the extents of the box, the OpenVDB file, and the grids from the OpenVDB file which correspond to useful primitive variables.

  Below is an illustration of using the OpenVDB Viewer node supplied with RenderMan for Maya. An OpenVDB file has been loaded from disk and the contents of the density grid visualized in Maya. The Primitive Variables tab describes the mapping from the OpenVDB grids to the variables that will be made available to RenderMan.



Volume Shading

With the geometry description of a volume being relatively simple, setting the shading parameters of the volume is critical to the overall look. For volume effects like fog, smoke, and clouds, RenderMan provides a dedicated shader called PxrVolume. After performing a volumetric simulation, a typical workflow for rendering involves setting up the geometry description as above, deciding which variables in the geometry description need to be mapped either directly to inputs of PxrVolume, or remapped using intermediate Pattern nodes, and finally tweaking the settings in PxrVolume to control the overall volumetric look. Note that scattering in the volume is controlled by the Max Path Length parameter in the chosen integrator and not the individual diffuse and specular trace depths.

The "dice" "minlength" attribute should be set to -1 to provide a hint about what the minlength should be based on the voxel data. This may become the default behavior in the future.

Output

For debugging purposes, in the Renderman.ini file you can find /prman/deepcomp/flagvolumes and Option "deep" "int flagvolumes" controls now interpret the value 1 as an "auto" mode. This is on by default, but off if generating any deep images in DTEX format. Use 2 to force it always on. This only affects samples generated by volumes.