OpenVDB Implicit Field Plugin

> ⚠ This documentation is intended for developers working directly with the RenderMan interface or RIB files.

The OpenVDB Implicit Field Plugin impl_openvdb.so supports the use of OpenVDB files, either as implicit surfaces or as volumes. Two string arguments are required (a third argument is optional, as described below). The first argument is the name of the OpenVDB file. The second argument is the name of the grid which will be used as the source of the field function. This grid must be of float type, and can be either a level set (getGridClass() == openvdb::GRID_LEVEL_SET) or a fog volume (openvdb::GRID_FOG_VOLUME). If the grid is a level set, it will be converted to a fog volume where the interior voxels have a field function value of 1, the exterior voxels have a value of 0, and the narrow-band voxels have values varying linearly from 0 to 1. If the field function grid's class is not specified in the OpenVDB file, or is neither a level set nor a fog volume, an override can be supplied by appending ":levelset" or ":fogvolume" to the name of the grid.

The plugin supports rendering of OpenVDB data both as a surface (using RiBlobby) and a volume (RiVolume). Here is an example of the syntax for rendering the OpenVDB dragon dataset as a surface.

```
Blobby 1 [1004  0 0 0 2 1] [0]
    ["${RMANTREE}/lib/plugins/impl_openvdb.so/impl_openvdb.so" "dragon.vdb" "ls_dragon"]
```

And here is an example of the syntax for rendering the OpenVDB dragon dataset as a volume.

```
Volume "blobbydso:${RMANTREE}/lib/plugins/impl_openvdb.so" [-99 103.4 -41.6 49.6 -57.6 77.3]
    [0 0 0] "constant string[2] blobbydso:stringargs" ["dragon.vdb" "ls_dragon"]
```

Deformation motion blur is supported by supplying the name of a grid containing velocity data in the optional third string argument. This grid must of type op envdb::Vec3f, and must be a fog volume. (If the velocity data's grid class is set otherwise in the OpenVDB file, it can be overriden by appending ":fogvolume" to the name of the grid.) Here is an example of the syntax for rendering a OpenVDB dataset containing a volume whose field function is contained in the grid named "density", and a velocity in the grid named "velocity".

```
Volume "blobbydso:${RMANTREE}/lib/plugins/impl_openvdb.so" [-30 30 -30 30 -30 30]
    [0 0 0] "constant string[3] blobbydso:stringargs" ["sphere.vdb" "density"
    "velocity"] "constant float[2] time" [0 1] "varying vector[1] dPdtime" []
```

Primitive variables are supported and are bound to grids of the same name in the OpenVDB file. RenderMan primitive variables of type color, vector, point, and normal are bound to grids of type openvdb::Vec3fGrid while float variables are bound to grids of type openvdb::FloatGrid. In the following example, we use three grids (density, fuel, and glow) to drive the field function and two attached primvars; the result of vdb_print file.vdb is shown as well.

```
Volume "blobbydso:${RMANTREE}/lib/plugins/impl_openvdb.so" [-100 100 -100 100 -100 100]
    [0 0 0] "constant string[2] blobbydso:stringargs" ["file.vdb" "density"]
     "varying float fuel" [] "varying color glow" []

fuel         float -100 -100 -100  100 100 100  200x200x200   1.8MVox 17.1MB
density      float -100 -100 -100  100 100 100  200x200x200   1.8MVox 17.1MB
glow         vec3s -100 -100 -100  100 100 100  200x200x200   1.8MVox 51.2MB
```

Starting with RenderMan 24.2, the plugin takes additional parameters as a JSON string. The JSON string is supplied as an optional fourth string argument. If you do not want to specify a velocity grid but want to provide JSON parameters, you should leave the velocity grid argument as an empty string. The JSON string should be a top-level JSON dictionary containing one (or more) of the following keys:

- filterWidth
- velocityScale
- densityMult
- densityRolloff

These parameters are documented below. Here's an example of how the JSON argument can be passed to the DSO:

```
Volume "blobbydso:${RMANTREE}/lib/plugins/impl_openvdb.so" [-99 103.4 -41.6 49.6 -57.6 77.3] [0 0 0]
    "constant string[2] blobbydso:stringargs" [
        "dragon.vdb" "ls_dragon" ""
        "{\"filterWidth\": 0.5, \"velocityScale\": 1.0, \"densityMult\": 1.0, \"densityRolloff\": 0.0}"
    ]
```

By default, the plugin performs unfiltered lookups of voxel data. It can optionally also perform fully filtered lookups of voxel data in the OpenVDB file in order to avoid aliasing. In order to use the OpenVDB plugin's mipmapping functionality, the OpenVDB file must be processed using the vdbmake utility and the *filterWidth* should be set (the default value is 0, which disables mipmapping).

If filtering is enabled by setting the *filterWidth* to 1, the width of the filter used over OpenVDB voxels favors antialiasing over detail (the DSO picks the maximum dimension of a PRMan microvoxel); this may provide a blurry result if the PRMan microvoxels are very anisotropic due to a high relative shadingrate. Setting the *filterWidth* to a value smaller than 1 can alleviate the blurriness at the cost of potential aliasing.

By default, the velocity data is used directly by the plugin without any further transformation. Should the magnitude of the velocity vectors be inappropriate, the *velocityScale* parameter can be set in the JSON string argument. If set, the *velocityScale* will act as a multiplier on the values in the velocity grid.

The *densityMult* and *densityRolloff* parameters can be used to perform simple density manipulations. The *densityMult* scales the values in the density grid, while the *densityRolloff* parameter applies a logarithmic falloff (with higher values providing falloff). By default, the *densityMult* is set to 1 (no scaling) and the *densityRolloff* is set to 0 (no falloff). Using these parameters is more efficient than manipulating the density values in a shader.

In versions of RenderMan before 24.2, the *filterWidth* and *velocityScale* are passed to the DSO as optional float arguments. The first float argument is the *filterWidth* and the second is the *velocityScale*. Note that if you are passing a *velocityScale* as a float argument, then you will also have to pass a *filterWidth*. Also note that passing these parameters as float arguments is deprecated in RenderMan 24.2. Here's an example of how to pass the *filterWidth* and *velocityScale* as float args:

```
Blobby 1 [1004 0 2 0 3 1] [1 0.25]
    ["${RMANTREE}/lib/plugins/impl_openvdb.so" "water.vdb" "surface" "vel:fogvolume"]
```