

# V-Ray Proxy

This page describes the new V-Ray Proxy in Maya.

## Overview

V-RayProxy loads geometry from file on disk at render time only. The geometry is not present in the scene, and does not use up any resources, except for the viewport preview. This allows rendering scenes with many millions of faces, more than Maya itself can handle.

The V-Ray Proxy in Maya is a V-RayProxy node that loads the geometry from a file on disk. The node loads alembic ( *.abc* ) and V-Ray Proxy ( *.vmesh* ) files. Any mesh can be exported to an *.abc* or *.vmesh* file from Maya and then loaded as a proxy.

Meshes are exported to a special *.vmesh* file format. This file contains all geometric information for a mesh such as vertices and face topology as well as texture channels, face material IDs, smoothing groups, and normals. In short, everything that is needed to render the mesh is included in the file. In addition, the mesh is preprocessed and subdivided into chunks for easier access. The file also contains a simplified version of the mesh used for preview purposes in the viewports. More information about the *.vmesh* file format is available in the V-Ray SDK documentation that comes with the V-Ray for Maya installation.

This page describes working with the new and improved V-RayProxy node in V-Ray 5 that's simpler, faster and offers new and powerful features.

Scenes using the old V-Ray Proxy can be easily converted to the new V-RayProxy with the built-in converter found in the **V-Ray menu > Tools > Convert to New V-Ray Proxy Node**.

After loading the file, the V-Ray Proxy attributes are accessible in the Attribute Editor.

To load files with V-Ray Proxy, see the [Import V-Ray Proxy](#) page.

To export meshes to V-Ray Proxy, see the [Export Mesh to V-Ray Proxy](#) page.

For more information on how to work with the V-Ray Proxy, see the [V-Ray Proxy courseware](#) page.

## Basic Parameters

To load files with V-Ray Proxy, see the [Import V-Ray Proxy](#) page.

**File name** – The source *.vmesh* file. Animated proxies can be stored either in one single file, or as a sequence of files with one file per frame. In the latter case, you can use the string `<frame0n>` in the file name to have it replaced with the current frame number at render time, where *n* is an integer number specifying the number of digits. For example, if you enter *my\_proxy\_<frame04>.vmesh* as the file name, this will be expanded to *my\_proxy\_0000.vmesh* for frame 0, *my\_proxy\_0001.vmesh* for frame 1, and so on. If you use a sequence of files, the **Playback type** option is ignored, as V-Ray doesn't know how many frames are in the animation. The **Playback speed** parameter may also work unreliably with sequences of files.

**Geometry to load** – Select the viewport preview mode from the drop down menu. This does not affect the final render. This parameter is Keyable and can be used as a driven key as **Geom Type** in the Channel Box.

**None** – No preview. This mode does not read any information from the proxy file and is useful for loading many and massive proxies for final scene assembly.

**Bounding box** – The mesh is represented as a box scaled to fit the object in viewport.

**Preview Geometry** – The mesh is represented as a low poly proxy preview as defined in the *.vmesh* file. A preview is generated on the fly for *.abc* files.

**Full Geometry** – The full geometry of the mesh is visible in the viewport. The new V-Ray Proxy uses an optimized preview based on Maya GPU Cache objects in this mode.

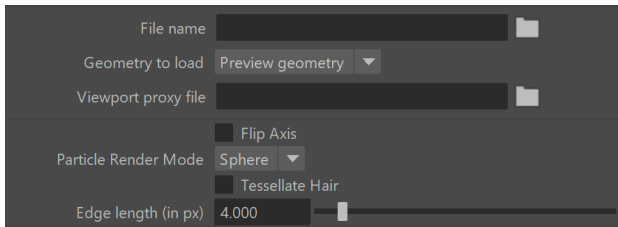
**Viewport proxy file** – Allows you to load a *.vmesh* file that represents the actual V-RayProxy in the viewport.

**Flip Axis** – Switches the proxy's vertical axis between Y and Z. It allows aligning vertical axis of the proxy with the vertical axis in the scene in cases where the proxy was not exported from Maya or when loading Alembic files.

**Particle Render Mode** – Controls how the particles in the proxy are rendered in the viewports. The available options are **Point** and **Sphere**.

**Tessellate Hair** – When enabled, tessellates hair strands for files containing hair.

**Edge length (in px)** – Specifies hair edge length in pixels for smoothing hair strands.





## Object Hierarchy

The proxy file structure is represented in a tree view with its full hierarchy. You can select individual elements and assign visibility or material overrides. Double click on the hierarchy level labels to expand/collapse the hierarchy.

**Viewport Picking** – This option is available when **Geometry to load** is set to **Bounding boxes**. It allows you to select parts of the proxy directly from the viewport and mark them in the Object Hierarchy.

**Filter** – A field allowing you to filter the nodes from the hierarchy list. Wildcards can be used in this field. Note that your filter input will not transfer if you copy the tab.



**Override Shader** – Allows overriding the material for the currently selected node in the tree. You can also enable an override for the selected material




via its  icon. When an override is enabled, it is indicated by  icon.


**Material Override** – Connect a material to override the currently selected material node. The override can be removed with the bin


icon ().


**Override Visibility** – Allows overriding the visibility for the currently selected node in the tree. You can also enable an override for the selected object via


its  icon. The override can be removed with the bin icon (.

There are 3 states of the visibility override:  default (inherit),  force visible and  force invisible.

**Default** – In this state , the selected object in the hierarchy will inherit the visibility override from its parent(s). The parent objects in the hierarchy may have the same default state, which means that visibility is not overridden.

**Force Visible** – In this state , the selected object in the hierarchy will always be visible. This can be used to force an object visible, while the parent objects are forced invisible. The state is inherited by the object's children in the hierarchy.

**Force Invisible** – In this state , the selected object in the hierarchy will always be invisible. This can be used to force an object invisible, while the parent objects are forced visible. The state is inherited by the object's children in the hierarchy.

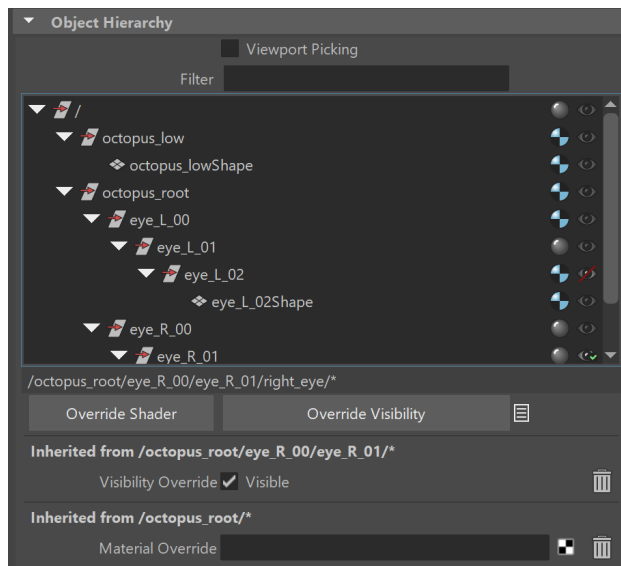
**Import/Export material rules** ( button) – Right-click the button to select import or export of material override rules to/from the current proxy. The syntax is the same as the **Material assignments file**, utilizing the new **scene Material** tag. *For more control over the command, you can use the [scripting version to import/export material rules](#).*

The **import** option reads the file and applies the rules in the Attribute Editor. Importing material assignment rules only adds to the existing ones, overriding only the matching ones. Import can be undone.

The **export** option writes all material overrides from the tree hierarchy as well as any custom overrides. Exporting the rules does not remove them from the scene.

Old material assignment rules (that don't use the **sceneMaterial** tag), cannot be imported from this button and only show a warning. Newly exported rules can be imported or used as **Material assignment file** together with a *.vrscene* file with materials.

The new VRayProxy offers rule-based visibility and material overrides. Note that the rule-based (advanced) overrides are executed **prior to** the hierarchy overrides.



## Advanced Overrides

The general workflow for applying advanced overrides is to write out the name of the object or, in one case - material, that needs its visibility or shader overridden between **wildcards(\*)** in the **Rule**. Pressing **Enter** applies the override. Alternatively, selecting the object from the hierarchy and middle mouse dragging its name onto the Rule also applies the override.

To apply an override to multiple objects, typing any value between wildcards in the Rule - number, symbol or word, is valid and applies the override to objects that contain that value. See [the example for applying an override to multiple objects](#).

Keep in mind that **V-Ray Proxy files (.vrmesh)** contain in their hierarchy objects and materials, while **Alembic files (.abc)** contain only objects. This is important as the Shader Override, in the case of V-Ray Proxy files, is applied to the materials in the hierarchy. In all other cases, the override is applied to the objects. See [the Shader Overrides example](#) for more information.

If the target object is located inside a group, its location is typed out similarly to a file directory or a URL - each level separated by a slash (/). For more information, see [the Visibility override example](#).

The list of Advanced Overrides is executed from the bottom to the top. Advanced Overrides are executed prior to the hierarchy overrides. This allows adding a more general override with a wildcard, then refining it using the hierarchy.

### Visibility Overrides (wildcard)

Overrides a selected object's visibility.

**Add New Item** –Adds a new visibility override rule.

**Rule** – A field to add the name or path of an object from the file's hierarchy. The object matched by the rule will receive the override.

**Visible** – Specifies whether visibility should be on or off.



## Shader Overrides (wildcard)

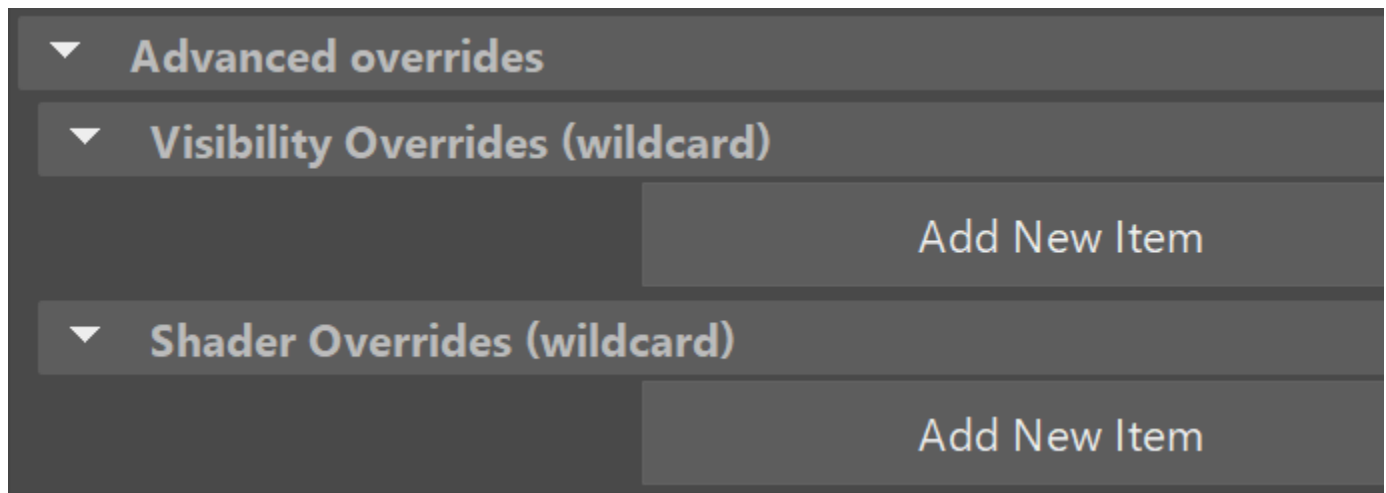
Overrides a selected object's shader (for V-Ray Proxy files) or a material in the scene (for Alembic files).

**Add New Item** –Adds a new shader override rule.

**Rule** – A field to add the name or path of an object from the file's hierarchy. The object matched by the rule will receive the override.

**Shader** – Connect a scene material to assign it to the objects matched by the rule.

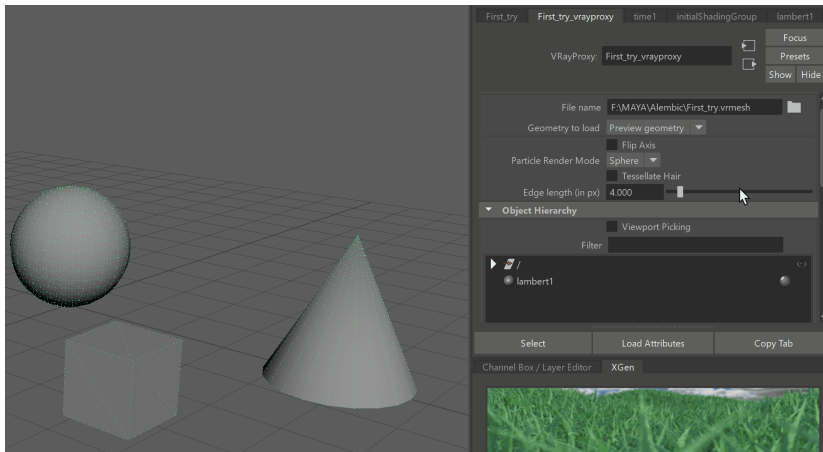
The rule can be removed with the bin icon () , and the Up/Down buttons () allow re-ordering the list.



## Example: Applying Visibility Overrides

- In the first example: typing **/group2/group1/\*** matches all objects contained inside *group1*, which is contained inside *group2*. This format is similar to a file directory or a URL. Only typing **\*group1\*** in the Rule also applies the override to those objects. When applied, the override is turned on by default, that can be toggled through the **Visible** option.
- In the second example: the original rule is deleted using the bin icon and the desired object is selected from the hierarchy. Its path is then highlighted and dragged, using the middle mouse button onto the Rule.

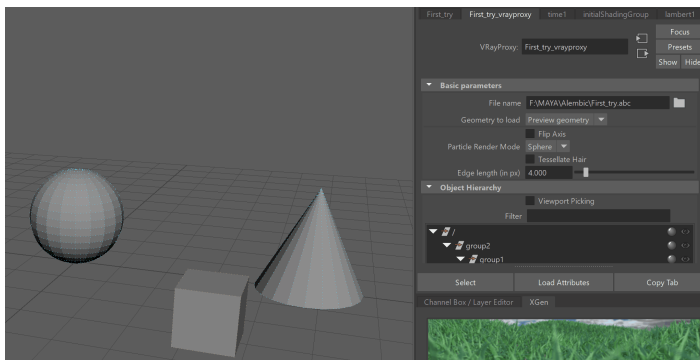
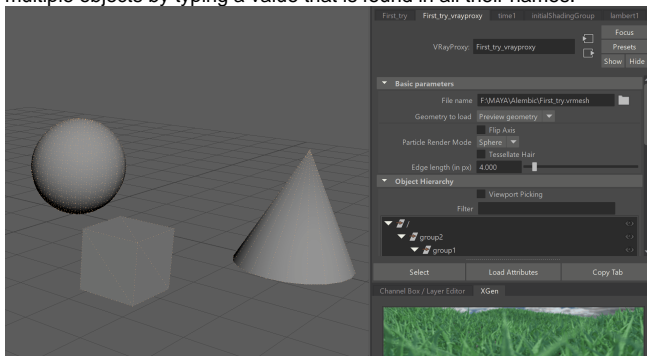
Pressing Enter confirms the input and applies the Visibility Override to the new object.



## Example: Applying Shader Overrides

This example shows the application of a **Shader Override** to a **V-Ray Proxy file (.vrmesh)**. The Shader override applies a new material on top of the original lambert. This is because V-Ray Proxy files contain materials and objects in their hierarchy and it is possible for the shader to be overridden directly.

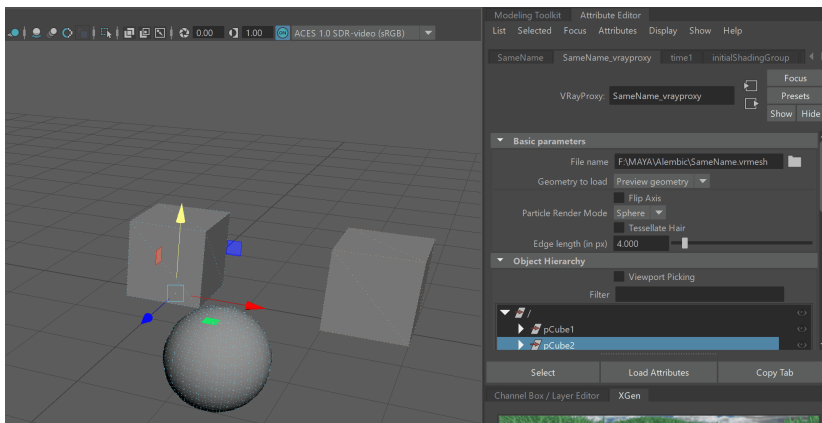
This example shows the application of a **Shader Override** to an **Alembic file (.abc)**. Alembic files contain only objects in their hierarchy, so the override is applied to the selected cube shape. The override can also be applied to multiple objects by typing a value that is found in all their names.



---

## Example: Applying an Override to Objects with Similar Names

In this example, typing **\*cube\*** (the word cube between wildcards) in the Rule targets all objects in the hierarchy containing that word. This applies the Visibility Override to both of them. This input is applicable to both **.vrmesh** and **.abc** files, and to both Visibility and Shader override types. The keyword does not have to be a word - it can be a symbol or number as well. As long as all desired objects contain that value inside the wildcards, the shader is applied to them.



---

## Animation Parameters

**Playback speed** – A multiplier for the speed of the animation. Putting negative numbers here makes the animation play backwards. This option may not work very well for sequences of **.vrmesh** files.

**Use alembic animation offset** – When enabled, automatically uses the frame offset from the alembic file. When disabled, the frame offset is set manually.

**Start offset** – Offsets the beginning of the animation by the given number of frames. You can use positive as well as negative values here.

**Playback type** – Specifies a playback mode. This option is ignored if you use a sequence of separate **.vrmesh** files.

**Loop** – The animation is looped by skipping to the first frame once it has finished.

**Play once** – The animation is played once.

**Ping-pong** – The animation is looped by playing it backwards once the last frame has been reached and then playing it forward again when the first frame is reached.

**Still** – The animation is not played. Instead, just the **Start offset** frame is shown.

**Sequence override** – Allows you to manually override the start frame and the animation length to read (in frames) for animations stored in sequences of **.vrmesh** files (where each frame of the animation is stored in a separate proxy file). Importing proxy sequences requires a special format that specifies how to match the frame padding in the file name. See the **File name** section of the [Import V-Ray Proxy](#) page for more details.

**Sequence start** – Specifies the first frame of the animation.

**Sequence length** – Specifies the length of the animation to be played.

Playback speed	1.000
	<input type="checkbox"/> Use alembic animation offset
Start offset	0.000
Playback type	Loop ▼
	<input type="checkbox"/> Sequence override
Sequence start	0
Sequence length	0

## Alembic Layers

The Alembic layers feature allows modification of a selected alembic proxy by adding layer files on top of it. Each layer transforms the proxy with a different set of properties. These transformations include adding properties to a shape (e.g. UV mapping of an object that has no UVs), overriding properties, adding new objects, pruning objects or properties, replacing an object's (or properties') hierarchy with a new one, etc.

You can combine different cache files. Whenever the same object is found in a layer, all of its properties are considered.

**Add New Item** – Adds a new layer to the stack.

**Enabled** – Each layer's effect can be enabled or disabled by the checkbox.

**File** – Loads an `.abc` file that can be used as a supplement or an override on top of other alembics. For example, you can add an alembic file that contains UVs to another alembic containing a polymesh without UVs.

**Browse** – Allows you to browse and open an `.abc` file.

**Delete** – Removes the layer from the stack.

	Add New Item
--	--------------

## Alembic proxy parameters

**Starting object path** – Allows to specify a starting path in the Alembic file; only objects below that path are rendered. The path may start with `ABC/` or it may be omitted.

**Recompute bounding box** – Enabling this check box forces V-Ray to recompute the bounding box for the geometry before rendering. When this is off, V-Ray uses the bounding box specified in the Alembic file. However, sometimes these bounding boxes are not correct, and in that case the rendering is also incorrect. To avoid such issues, enable this option. Note that this option might slow down the rendering.

**Instancing** – When enabled, if there are multiple instances of an object it is loaded once and the memory is reused.

**Compute normals** – This option allows you to force smooth normals on the geometry in case they were not originally smoothed.

**Smooth angle** – When the **Compute normals** option is enabled, this specifies the angle below which normals are smoothed.

**Flip normals** – Reverses the direction of the normals.

**Preview faces** – Allows you to specify how many faces are going to be used to show a preview of the proxy in the view port.

**Preview hairs** – Allows you to specify how many splines are going to be used to show a preview of the proxy hair in the view port.

**Hair width multiplier** – This multiplier allows you to control the hair width during rendering.

**Preview particles** – Allows you to specify how many particles to show in the viewports.

**Particle width multiplier** – Controls the size of the particles when rendering.

**Velocity multiplier** – Controls the length of the motion blur when rendering.

**Subdivide all meshes** – Subdivide the meshes before rendering using the OpenSubdiv library.

**Subdivision level** – The number of subdivision levels.

**Subdivide UVs** – Allows you to choose whether or not the UVs of the object are subdivided at the borders.

**Preserve geometry borders** – Specifies whether geometry borders are preserved or not.

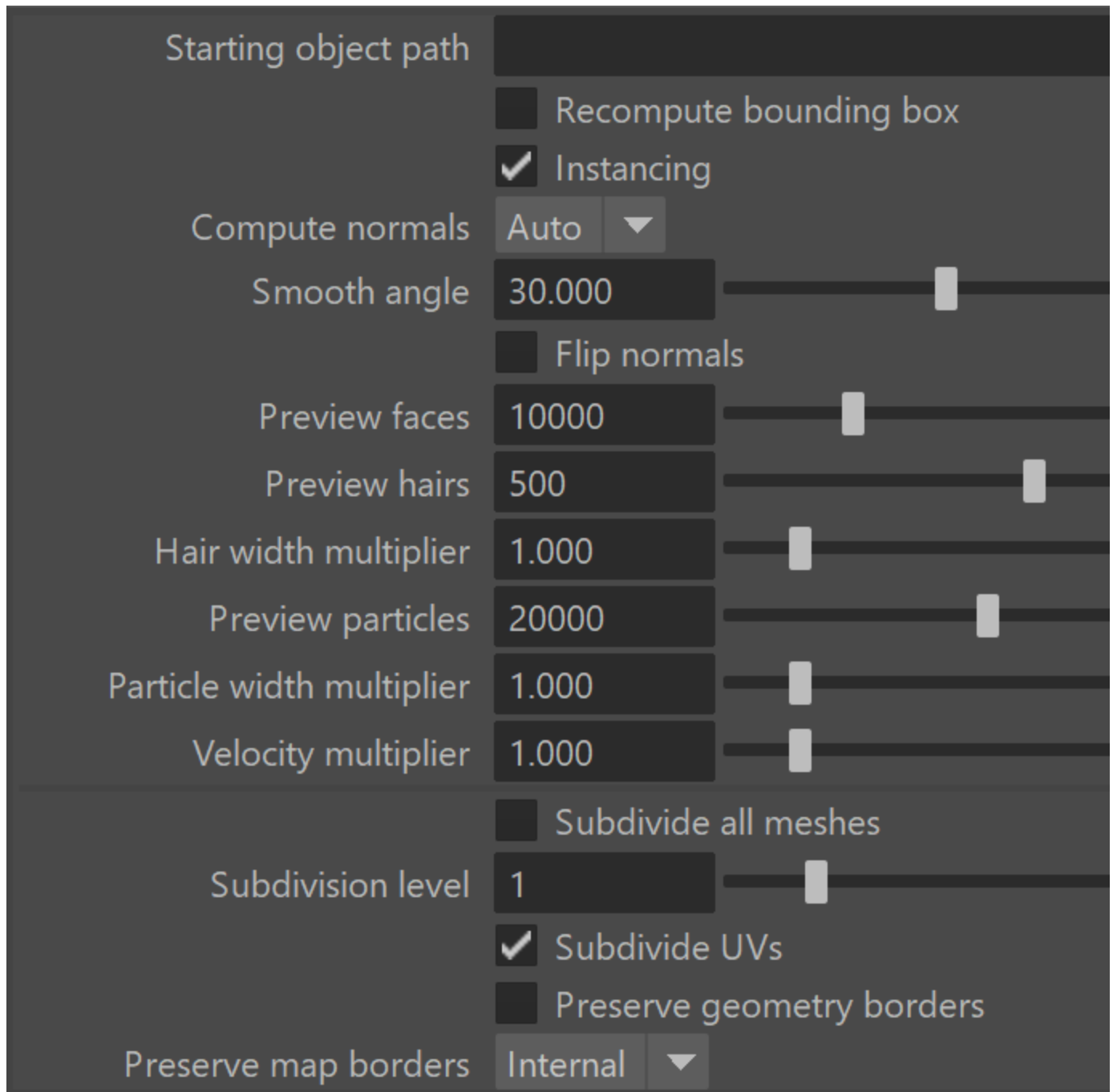
**Preserve map borders** – Specifies how to handle subdivisions of UV coordinates at UV seams when Subdivide UVs is enabled. The possible values are:

**None** – UVs are always subdivided regardless of whether they are on a UV seam or not.

**Internal** – Only preserve UVs if they are on an internal UV seam.

**All** – Does not subdivide UVs on UV seams.





## Assigning Random Colors per Alembic Particle

These example workflows show how to assign random colors to alembic particles.

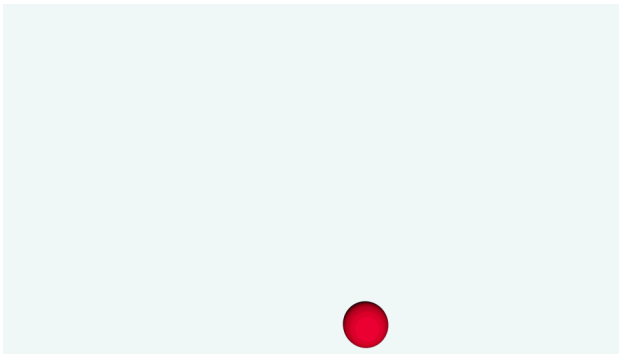
By applying a [V-Ray MultiSubTex](#) to a V-Ray Material, you can assign different colors per alembic particle. Here is the workflow:

- Import the alembic particles as a V-Ray Proxy.
- Create a V-Ray Material.
- Add a **V-Ray MultiSubTex** texture as the Diffuse Color of the V-Ray Material.
- Add multiple layers to it. Assign different colors/textures to each layer.

- Make sure to set the MultiSubTex's Get ID option to **Random by Instance ID**.
- Apply the V-Ray Material to the alembic particles.

By applying a **Ramp** texture to a V-Ray Material, you can create a range of colors to apply to the alembic particles. Here is the workflow:

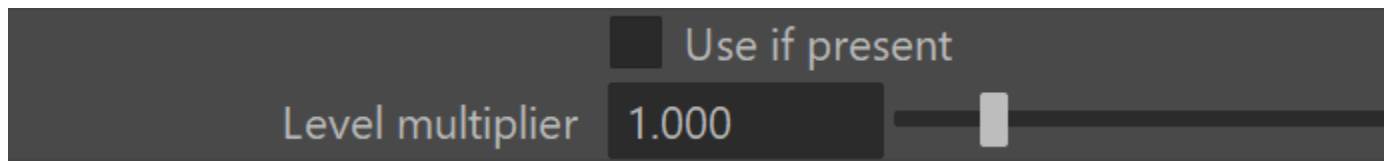
- Import the alembic particles as a V-Ray Proxy.
- Create a V-Ray Material.
- Add a Ramp texture as the Diffuse Color of the V-Ray Material.
- Adjust the ramp's settings to your liking. It can be gradient or not, adjust the colors and orientation.
- To apply the ramp's colors per particle, you need to create a **SamplerInfo node**. Select it and in the Attributes Editor, go to **Attributes > V-Ray > Additional Attributes**. This option adds V-Ray attributes to the node.
- Connect the **VRay Random by ID** attribute from the SamplerInfo node with the proper **UV coordinate** attribute of the Ramp texture. Which UV coordinate you connect depends on the Ramp Type - if it is a V Ramp, U Ramp, UV Ramp etc. - connect the corresponding coordinate attribute.
- Apply the V-Ray material to the alembic particles.



## Point Cloud

**Use if present** – Enables the usage of point cloud data if available.

**Level multiplier** – Determines the way point cloud levels are loaded. A value of 1.0 means that the level to load is determined exactly by the distance from the camera to the object. A value smaller than 1.0 means that the level is of greater detail than required by distance. Values greater than 1.0 mean that the resolution of the level is smaller than the one determined by distance. A value of 0.0 means that no point cloud level is loaded and the original mesh is rendered instead.



## VRayMesh file info

This rollout provides information on the loaded VRayMesh including:

- Mesh filename, i.e. a path to the loaded file
- Animation contained in the proxy (when animation is in one file, not as a sequence of files)
- Number of voxels
- Number of preview faces, hair geometries and particle count.

- Bounding box information
- UV sets and their names
- Color sets and their names
- Shaders sets and their full names

This can be useful in several situations. For example, when using the [VRayUserColor](#) node to read UV sets and color sets, the set names can be found in this rollout. Additionally, when using animation overrides, the rollout shows what frame of the proxy animation is rendered at a given timeline frame.

```
Mesh filename: F:/Proxies/EC - B&BITALIA.BENCHES.BULL.vrmesh
Number of frames: 1
Number of voxels: 5
Preview has 12592 triangles, 0 hairs, 0 particles
Bounding box: [0.000, 0.000, 0.000] - [75.269, 15.846, 19.580]
UV sets: 0
Color sets: 0
```

## Material Assignment Overrides

When loading a proxy file in a VRayProxy node, materials stored in `.vrscene` files can be assigned onto different objects from the proxy file. This requires a pair of a `.vrscene` file containing the materials and an XML file containing the "rules" for applying them to the objects in the proxy file.

**Material assignments file** – Specifies the location of an `.xml` file that contains the "rules" for material assignment. For alembic files, the "rules" are based on object paths, for `.vrmesh` files, they are based on material names. The wildcards `*` are allowed in the `.xml` file.

### Material assignments file

#### Alembic Sample XML script

```
<materialAssignmentRules>
  <patternRule>
    <pattern>pCube*</pattern>
    <material>VRayMtl1@material</material>
  </patternRule>
</materialAssignmentRules>
```

The sample XML assigns the `VRayMtl1` material to all alembic objects matching the pattern `pCube*`, i.e. all objects whose name starts with `pCube`.

The new alembic proxies are imported with full path names by default, so the pattern `<pattern>pCube*</pattern>` won't work for them, but `<pattern>/pCube*</pattern>` will.

If you want to use the new proxy with your old XML file unchanged, import the alembic files with the **Use full path names** option disabled (this can be done via scripting but then you'll lose the Object Hierarchy), or try using the **Convert To New V-Ray Proxy Node** tool (it keeps the old short path names as well as the linked rules file, but again you lose the Object Hierarchy). Otherwise, the material assignment override rules will need to be adjusted to match the full path names pattern.

#### Proxy Sample XML Script

```
<materialAssignmentRules>
  <patternRule>
    <pattern>VRayMtl1</pattern>
    <material>VRayMtl1@material</material>
  </patternRule>
</materialAssignmentRules>
```

The sample XML assigns the `VRayMtl1` material to all `vrmesh` objects matching `VRayMtl1` pattern, i.e. all objects whose material name is `VRayMtl1`.

### Post-translate Python script

```
from vray.utils import *

appendSceneContent( "C:/mtlAssignments/materials.vrscene" )
```

For this feature to work, you need to specify the path to the `.vrscene` containing the materials (procedural textures or paths to bitmaps). This only works with post-translate python script. The following example script specifies the path to the `.vrscene` file and should be copied into *Render Settings > Common > MEL / Python callbacks > Post translate python script*.

### Alembic Sample XML script with sceneMaterial tag

```
<materialAssignmentRules>
  <patternRule>
    <pattern>pCube*</pattern>
    <sceneMaterial>VRayMtl@material</sceneMaterial>
  </patternRule>
</materialAssignmentRules>
```

The sample XML assigns the `VRayMtl` material to all alembic objects matching the pattern `pCube*`, i.e. all objects whose name starts with `pCube`. It uses the new `sceneMaterial` tag.

The material assignment overrides is still an experimental feature and can change at any point!

## Example: Instancing V-Ray proxy with MASH

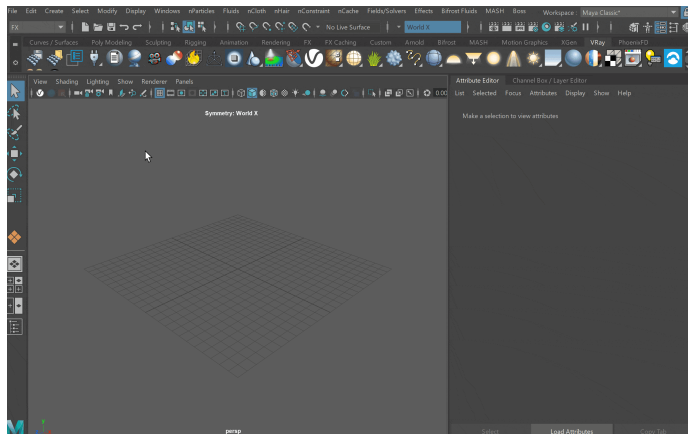
This example shows how to instance a proxy (.vrmesh) using MASH.

Load a proxy file into your scene using the **Import V-Ray Proxy** button in the toolbar.

To instance the proxy, while still selected, create a **MASH Network** (**MASH > Create MASH Network**). As a Geometry Type, select **Instancer**.

Now you can adjust the distribution.

The same steps apply when using an alembic file.



## Notes

---

- V-Ray 5.0 for Maya 2018 and above supports instancing of .vrmesh files via the Maya Instancers found in FX menu set > nParticles > Instancer.
- To create a Maya Mesh from the geometry from file (including normals, default uv set and per face shaders), you can use the -**vrayImportVRmeshGeom** command. Available flags are -frame (specifies which frame from an animation to consider), -loadHair (imports hair data) and -name (sepcifies a name for the newly created node).

### Example: Animation Frame

```
vrayImportVRmeshGeom "D:/anim_foliage.abc" -frame 13 -name "Foliage"
```

### Example: Import Hair Data

```
vrayImportVRmeshGeom "D:/hair.abc" -loadHair -name "Hair"
```

- To remove any legacy visibility or material overrides at the root level of a proxy, use the following command:

```
vrayRemoveExtraProxyRules
```