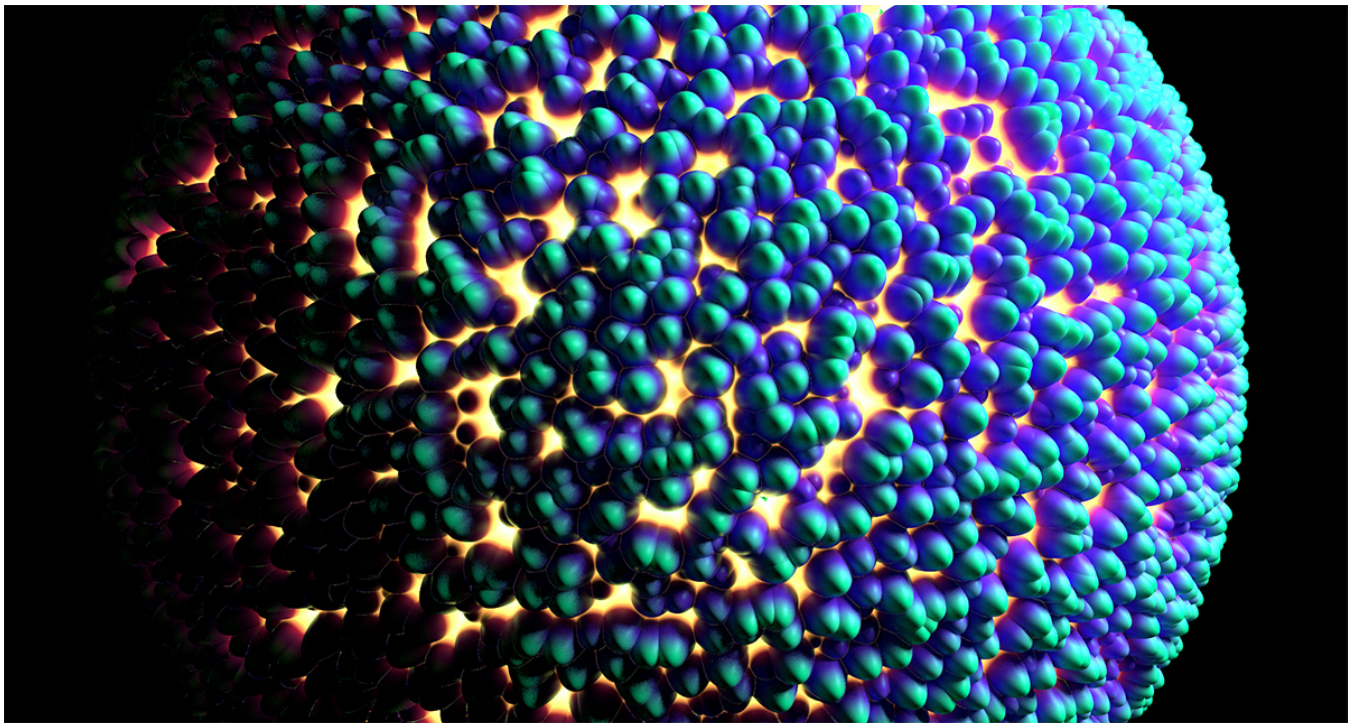
Creating Custom OSL HyperShade Nodes



Sample Render using the custom Voronoi 3D Texture Node.

The following package contains all the material described in this tutorial: Custom_Hypershade_Nodes.zip.

Shader development in *3Delight for Maya* is quite simple. With OSL shaders, the process gets simpler yet, as *3Delight for Maya* can automatically register as *Maya* shading nodes user-provided OSL shaders. *3Delight* supports all the required functions to properly run OSL shaders including all the most advanced *closures,* refer to Performance Analysis for more informations*.*

This tutorial explains how to create your own *HyperShade* node. As an example, we show how to develop a simple voronoi noise *pattern* as a *Maya 2D Texture.* For more informations about procedural textures we recommend this modern classic: Texture & Modeling: a procedural apprach.

Content:

Using the example package

Download and decompress Custom_Hypershade_Nodes.zip. To use the package files, you can either copy them to specific locations of your *3Delight* installation, or define environment variables to indicate their locations.

Copying example files to your 3Delight installation

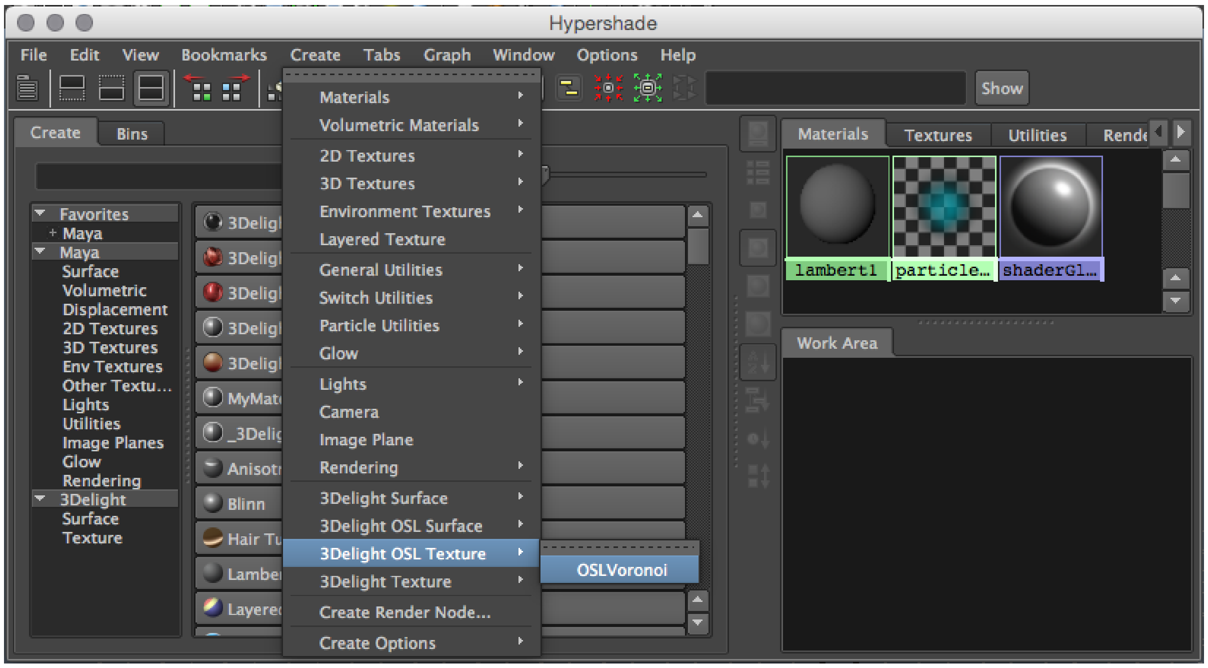
- Copy the `customShadingNodes` directory itself into the `maya` folder of your *3Delight* installation - for instance, place the directory copy into `C:\Program Files\3Delight\maya\` on a default Windows installation of *3Delight Studio Pro,* so to have it in `C:\Program Files\3Delight\maya\customShadingNodes`
- **Optional** - copy the files from the icons directory into the icons folder of your *3Delight* installation. Make sure to choose the subdirectory that corresponds to the *Maya* version you are using. For instance, on a default Windows installation of *3Delight Studio Pro*, for *Maya 2015*, the correct destination location is `C:\Program Files\3Delight\maya\2015\icons`.

Or, defining environment variables

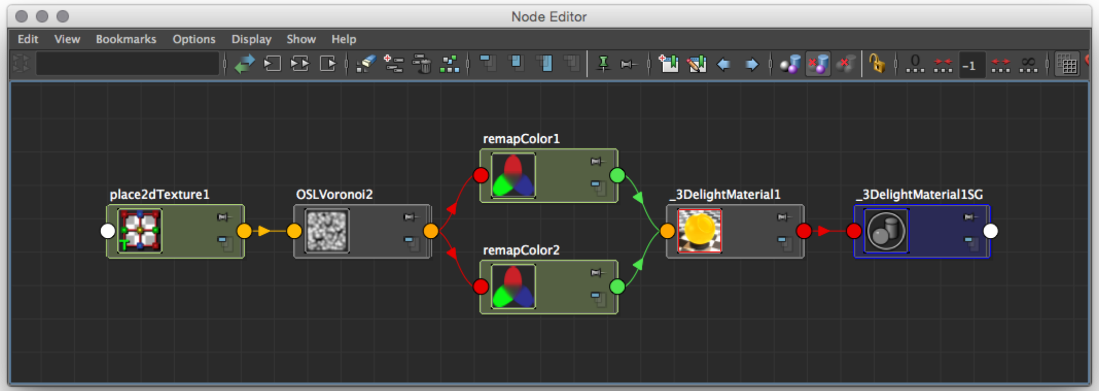
The `XBMLANGPATH` definition is optional. On Linux, paths set in `XBMLANGPATH` must end with `/%B` .

Environment Variable	Value
`_3DFM_USER_OSL_PATH`	Path to the `customShadingNodes` directory of the downloaded package.
`XBMLANGPATH`	Path to the `icons` directory of the downloaded package.

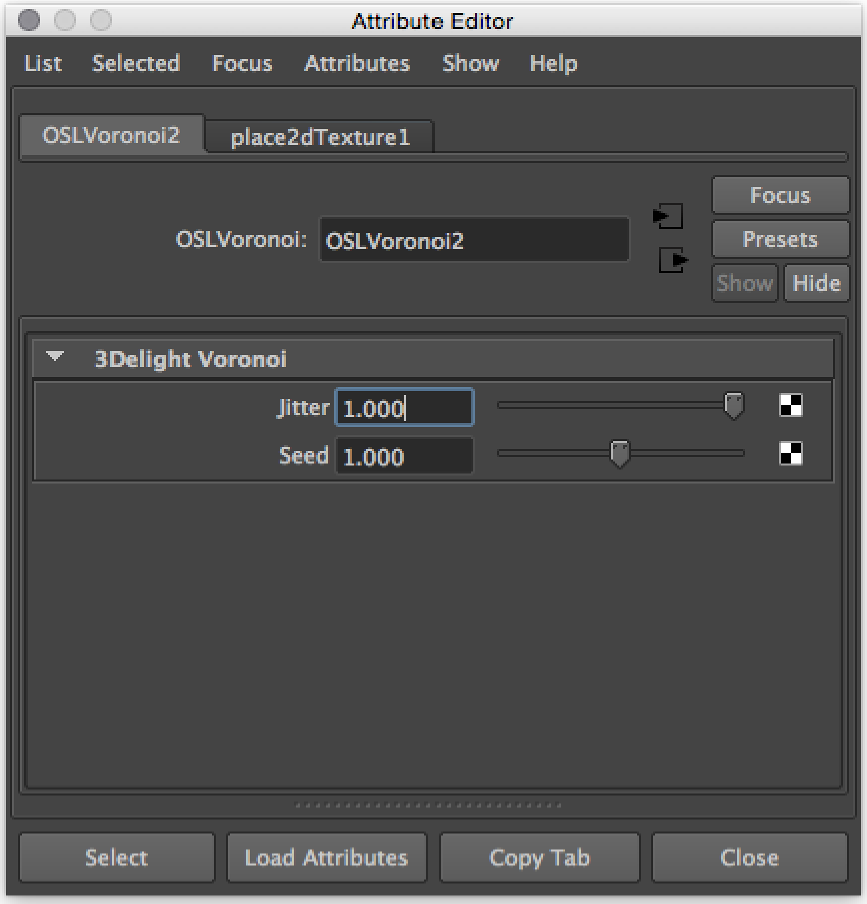
Lauch Maya and go to the *Hypershade* editor. The `OSLVoronoi` node should appear listed under *3Delight Texture* in the *Create* tab, and under *3Delight OSL Texture* in the *Create* menu. Note that the Maya script editor will report the creation of the new OSL nodes.



The Voronoi node viewed in the Create *menu of the* HyperShade



The Voronoi node viewed in the Node Editor*. The `place2dTexture` node is automatically created and connected.*



OSLVoronoi: OSLVoronoi2

3Delight Voronoi

Jitter 1.000

Seed 1.000

Voronoi noise connected to the 3Delight Material.

Components of a custom shading node

There is only one required component to define a custom shading node: **a compiled OSL shader**. Optionally, it is possible to add icons to have a better visual representation of the node inside the *Hypershade* and the *Outliner.*

Components location

The following table shows the default location where *3Delight for Maya* will look for components, and the environment variable that allows you to specify another location for them.

Component	**Environment Variable**	**Default Location**
Compiled OSL shaders	`_3DFM_USER_OSL_PATH`	`C:\Program Files\3Delight\maya\customShadingNodes`
Icons	`XBMLANGPATH`	`C:\Program Files\3Delight\maya\2015\icons` (varies according to the Maya version being used)

The compiled shader

This is a standard *Open Shading Language* surface shader, complied with `oslc` (provided with the *3Delight Studio Pro* package). The shader source code can be annotated to provide indications to *3Delight for Maya* about the *Maya* shading node classification and its appearance in the *Attribute Editor*.

While annotations are optional, it is highly recommended to provide the *shader annotation* that specifies the type ID.

When no type ID is provided as annotation, *3Delight for Maya* will attempt to generate one in the range reserved for studio internal node types Because this range is not large enough, it is possible that two different OSL shader names will result in identical type IDs; this will cause problems when reading a scene in *Maya Binary* format. For this reason, using the automatically generated IDs is not recommended for usage outside of prototyping purposes, refer to mayaid.autodesk.io to obtain a reserved block of IDs.

Expand the *annotations* section below and refer to the `maya_typeID` shader annotation for more details on how to define your type ID.

Annotations in the OSL source code

The annotations are defined between double square brackets. Multiple, comma-separated annotations, can be used in one set of double square brackets.

Shader annotation

This annotation is provided between the shader name and its parameter list. For instance:

```
surface OSLVoronoi [[ string maya_type = "texture", string maya_typeID = "0x00" ]] ( ... )
```

The supported shader annotations are:

`string maya_typeID`

Specifies the *type ID* used for the node registration. This is a integer that *Maya* uses to identify the node type, most notably when saving a scene in the *Maya Binary* format. Each OSL shader that defines a given shading node type should be assigned a unique type ID. You can chose a value between `0x0000` and `0x007F`, or between `0x7F01` and `0x7FFF` for your shading node type.

The IDs from `0x0000` to `0x7FFF` are reserved for node types that are used internally in a studio. *3Delight for Maya* will generate a type ID between `0x0080` and `0x7F00` if no `maya_typeID` annotation is provided. You can also request your own reserved node ID range to *Autodesk* , for free. This is also the recommended solution if you intend to share your nodes with users outside your studio.

You may use any ID between `0x0000` and `0x7FFF` if you always provide the `maya_typeID` annotation for every custom OSL shader you define. In this case *3Delight for Maya* will never need to generate a type ID.

`string maya_type`

Specifies the *Maya shading node classification*. The classification affects where the node is presented in the *Hypershade* tree lister and menus. Some classification types will also change the node creation mechanism to automatically create and connect related nodes - for instance, creating a *surface shader* will also create and connect a *shading group*.

The currently supported types are:

`texture`

The shading node will be classified as a *texture* node. The node will be classified as a *2D texture* node, for which *Maya* automatically create and connect a *place2DTexture* node, if:

- it contains a `float[2]` parameter that has the `string maya_name = "uv"` annotation, and
- it contains a `float[2]` parameter that has the `string maya_name = "uvFilter"` annotation.

`lens`

The shading node will be classified as a *lens* shader. Lens shaders can be connected to a camera's lens shader *3Delight extension attribute*.

surface

The shading node will be classified as a surface shader. It will be connected to a shading group upon creation.

Parameter Annotations

Parameter annotations are provided between a parameter's default value and the comma that ends its declaration. For instance:

```
float i_jitter = 1.0 [[ string maya_name = "jitter" ]],
```

The supported parameter annotations are:

`string maya_type`

Specifies the type of the *Maya* attribute related to the shader parameter. For now, only `bool` is supported to display an integer parameter as a checkbox.

`string maya_name`

Specifies the name of the *Maya* attribute related to the shader parameter.

`string maya_label`

> Specifies the label to use for the *Maya* attribute in the various *Maya* editors (*Attribute Editor*, *Node Editor*, *Channel Box*, etc.).

`string maya_group`

> Specifies the label of a *Frame Layout* into which the *Maya* attribute will be displayed in the *Attribute Editor*.

`float maya_min`

> Specifies the soft minimum value for the *Maya* attribute attribute related to the shader parameter.

`float maya_max`

> Specifies the soft maximum value for the *Maya* attribute attribute related to the shader parameter.

`int maya_hidden`

> When set to 1, the Maya attribute will not be shown in the *Attribute Editor*. It will still be visible in the other Maya editors (*Node Editor*, *Channel Box*, etc.).

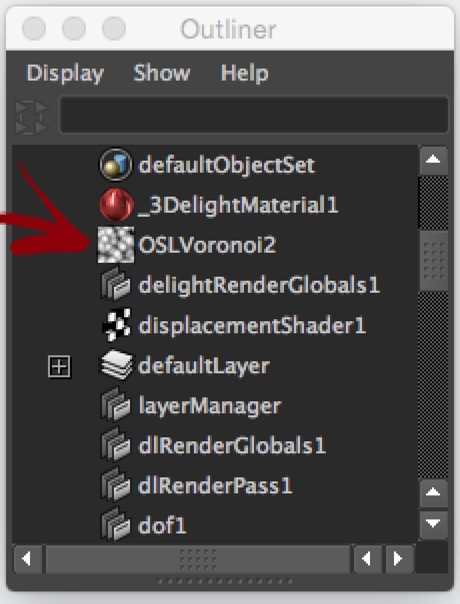
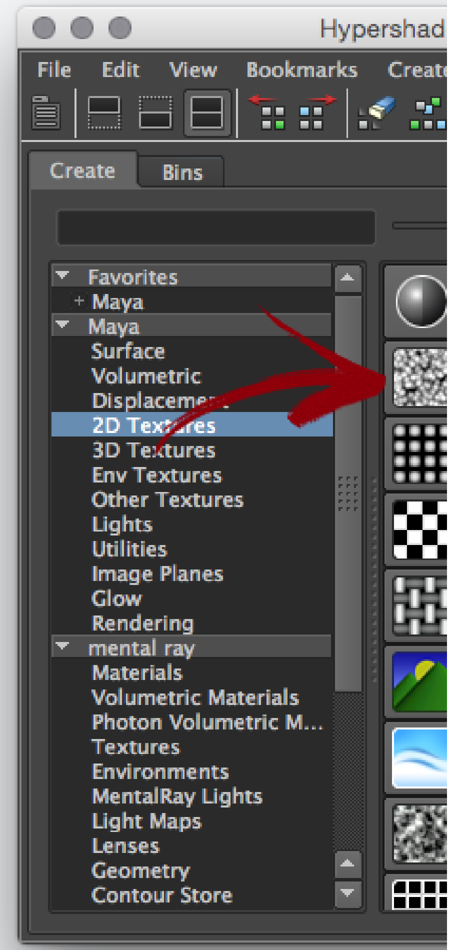
Annotated OSL shader example

OSL Shader Source (OSLVoronoi.osl)

```
/** Returns the U comp of the default UV set */
float GetS()
{
        float st[2];
        if( getattribute("st", st) )
        {
                return st[0];
        }
        else
        {
                return u;
        }
}

/** Returns the T comp of the default UV set */
float GetT()
{
        float st[2];
        if( getattribute("st", st) )
        {
                return st[1];
        }
        else
        {
                return v;
        }
}


surface OSLVoronoi [[ string maya_type = "texture" ]] (
                float i_jitter = 1.0 [[
                        string maya_name = "jitter",
                        string maya_label = "Jitter",
                        float maya_min = 0.0,
                        float maya_max = 1.0,
                        string maya_group = "3Delight Voronoi" ]],
                float i_seed = 1.0 [[
                        string maya_name = "seed",
                        string maya_label = "Seed",
                        float maya_min = 0.0,
                        float maya_max = 2.0,
                        string maya_group = "3Delight Voronoi" ]],
                float i_uvCoord[2] = {GetS(), GetT()} [[
                        string maya_name = "uv",
                        int maya_hidden = 1,
                        int skip_init = 1 ]],
                float i_uvFilterSize[2] = {0, 0} [[
                        string maya_name = "uvFilterSize",
                        int maya_hidden = 1,
                        int skip_init = 1 ]],

                output color outColor = 0.0 [[
                        int maya_hidden = 1 ]],
                output float outDistance = 0.0 [[
                        int maya_hidden = 1 ]] )
{
        float edgeDist;
        float outside;

        /*
                i_uvCoord is multiplied by 10 to produce a visible result with default
                place2DTexture. This is similar to setting RepeatU / V to 10.
        */
        point pp = point(i_uvCoord[0] * 10, i_uvCoord[1] * 10, i_seed);

        point position;

        point thiscell = point( floor(pp[0])+.5, floor(pp[1])+.5, floor(pp[2])+.5);
        float f1 = 1000;
        for (int i = -1;  i <= 1;  i += 1)
        {
                for (int j = -1;  j <= 1;  j += 1)
                {
                        for (int k = -1;  k <= 1;  k += 1)
                        {
                                point testcell = thiscell + vector(i,j,k);
                                point pos =
                                        testcell + i_jitter * ((vector)cellnoise(testcell) - 0.5);
                                vector offset = pos - pp;
                                float dist = dot(offset, offset); /* actually dist^2 */
                                if (dist < f1)
                                {
                                        f1 = dist;
                                        position = pos;
                                }
                        }
                }
        }

        outDistance = sqrt(1 - f1);
        outColor = outDistance;
}
```

The icons

You can add icons to both the *Outliner* and *Hypershade* (this applies to both texture nodes and shader nodes). The table below details the convention for creating the icons for our Voronoi Noise.

	Outliner	HyperShade - Node
	 Outliner Display Show Help defaultObjectSet _3DelightMaterial1 OSLVoronoi2 delightRenderGlobals1 displacementShader1 defaultLayer layerManager dlRenderGlobals1 dlRenderPass1 dof1	 Hypershad File Edit View Bookmarks Creat Create Bins Favorites Maya Maya Surface Volumetric Displacem 2D Textures 3D Textures Env Textures Other Textures Lights Utilities Image Planes Glow Rendering mental ray Materials Volumetric Materials Photon Volumetric M... Textures Environments MentalRay Lights Light Maps Lenses Geometry Contour Store
Icon Resolution	20 x 20 pixels	32 x 32 pixels
Format	Transparent 24 bits PNG	Transparent 24 bits PNG
Naming Convention	"out_" + <node_type> + ".png"	"render_" + <node_type> + ".png"
Example	out_3DelightVoronoi.png	render_3DelightVoronoi.png Note the transparent corners of the icon matching *Maya* built-in 2D Texture nodes.