OSL shaders in 3dsMax 2019

3dsMax 2019 from Autodesk introduced a subsystem for using and writing OSL shaders, mainly by the introduction of one key component in the material editor: the **OSL Map**. This feature is actually very well designed for a first release, and it has received good reviews. Some bugs remain, and the user interface is a bit rough in places, but it is definitely useful. There is still no support for OSL native surface shaders, volume shaders or displacement shaders, because the **closure** concept is not supported. Only generic OSL **shader** functions can be programmed and used as nodes in the shader network. The lack of closure support means that the full potential of OSL is not yet available to the shader writer through this particular tool, but a lot can be achieved by using the OSL Map for various texture-mapped parameters of the existing materials and modifiers. An OSL Map can have arbitrary input and output connections, including multiple outputs, and we recommend that you use the **Slate Material Editor** rather than the Compact Material Editor to handle them.

One nifty thing about the OSL Map is that it doesn’t necessarily require a renderer with OSL shader support. The **Arnold** renderer available in 3dsMax 2019 has native support for OSL, but the simple (and much faster) **Scanline** renderer does not. However, when you use an OSL Map with the Scanline renderer, the shader is executed separately from the renderer, by 3dsMax itself. This also makes it possible for 3dsMax to show a reasonably accurate baked preview of OSL maps in the interactive viewports.

We recommend you use the Arnold renderer for this exercise, but it’s not absolutely necessary. In fact, Arnold currently implements the slightly outdated OSL version 1.8, while the 3dsMax internal OSL shader execution engine implements at least some features of OSL 1.9. The changes are few and small, but they happened in the noise functions, which are important to this course. Therefore, you might want to use the scanline renderer for at least some of your experiments. If nothing else, it’s a lot faster, so you won’t have to wait as long for preview renderings while you are writing your shader. On the other hand, the scanline renderer does a very poor job with automatic derivatives, which are important for antialiasing, and Arnold allows you to open an **ActiveShade** rendering window where the image is updated automatically when you change anything in your scene. On a modern computer, an Arnold rendering of reasonable resolution for a simple test scene with only a couple of objects doesn’t take more than a few seconds. Either renderer has its advantages and disadvantages for OSL programming. Take your pick, but don’t hesitate to change your mind later.

In this exercise, we ask you to write at least two OSL shaders to use as maps on a simple object. The most obvious place to put an OSL Map with a procedural shader is of course the diffuse surface color, but you may want to try also modifying the opacity, specularity, or any other mappable parameter of whatever material you choose. Additionally, you should at least consider writing a shader for the surface **displacement**, because a procedural displacement shader is a very powerful and useful modeling method. It’s not good for everything, but try writing one, and consider making a displacement shader part of what you present as the result of this exercise.

The main point is this: don’t just blindly hack away with no plan and accept where that random walk takes you. Try to set a goal for your experiments in terms of what visual result you are striving for, and try to attain that goal. Feel free to redefine your goal as you go along and discover that it was too ambitious or too easy, or if you suddenly have an idea for something else and more interesting, but at least try to have a plan with what you are doing.

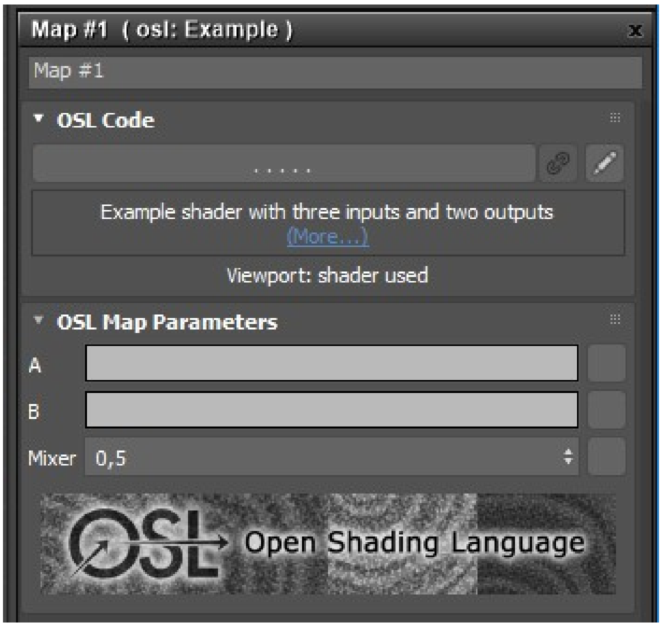
Getting started with OSL Map in Arnold

The renderer determines which materialas are available to you, so you need to decide on the renderer first of all. In the **Render Setup** dialog, choose Arnold as the renderer for the two modes **Production** and **ActiveShade**.

Create a simple object, like a sphere. Open the Material Editor (M), and pick the Slate Material Editor rather than the Compact Material Editor . If you open the wrong one, you can change it through the Modes menu at the top left of the material editor window.

Right-click in the central pane of the Slate Material Editor, and create a new Material of type Materials → Arnold → Surface → Standard Surface. If you feel intimidated by the huge amount of maps, you can start out simple by instead using a material of type Materials → Arnold → Surface → Lambert, but that is at the other extreme: it supports pure diffuse reflection only and is too limited for most applications.

Now, right-click again on the background pane of the material editor, and create a new Map of type Maps → General → OSL Map. Note that the Maps → OSL subsection contains some other OSL examples and utilities, but they are all variations on the general OSL Map, only with different source code.


Map #1 (osl: Example)

Map #1

Example shader with three inputs and two outputs (More...)

Viewport: shader used

Mixer 0,5

Double click on the title bar for the newly created OSL Map pane, to get its parameter dialog into the rollout pane at the right hand side of the material editor. Click the pencil icon at the top right to open a text editor where you can edit the OSL source code.

Drag a connection from the output of the OSL Map to the Diffuse Color input of the Standard material, and drag a connection from the material output to the object in your scene, or assign it to the object by selecting the object and clicking Assign to Selection in the command pane at the top of the material editor window.

Click Show Shaded Material in Viewport to get a rough but useful preiew of your material in the viewport.

Now it’s time to start editing the OSL code. Let’s cut down on the inputs and the outputs and write a shader with a single `float` input and one `color` output, using Perlin noise in a very straightforward manner:

```
shader Example (
float size = 1.0,
output color Out = 0.0
)
{
    Out = noise(P/size);
}
```

Hit the Recompile button, and note that the interactive pane in the material editor and the settings rollout for the OSL Map changes accordingly.

Make a test render. Now you’re all set for engaging in your own OSL programming experiments!

Displacement maps in Arnold

Displacement mapping in Arnold is performed by a separate modifier. In the Modify panel (top right, second tab), find the modifier **Arnold Properties**. In the rollout **Displacement**, you find the map slot for displacement mapping. Check the box **Enable**, Drag and drop a map connection from the material editor to the box that says **No Map** to pick a map, and check the box **Use Map**. Now, because this displacement is an object-space modifer, you need to have enough polygons for the displacement to look nice. If your object has too few vertices to displace, you can use the Subdivision rollout, right beneath the Displacement rollout, to apply a subdivision and/or smoothing to your object.

It would be nice if this subdivision could be dynamic in some manner, to avoid tesselating more than necessary and to spend the effort where it’s needed instead of blindly increasing the amount of vertices and polygons everywhere on the object. Unfortunately, there seems to be no good way of achieving this in the current version of the modifier. There are controls in the **Adaptive** section of the Subdivision rollout that seem to be meant for this purpose, but they are unresponsive. If you find a way to get this working, please let us know, because dynamic adaptive tesselation is really helpful when working with displacement shaders. This is the first release of the OSL subsystem in 3dsMax, and adaptive tesselation is hopefully going to be properly implemented in a future release of the software. For now, you need to make sure to create enough polygons in your objects for your displacement shaders to work as intended.

That’s it. You’re now ready to start experimenting on your own!

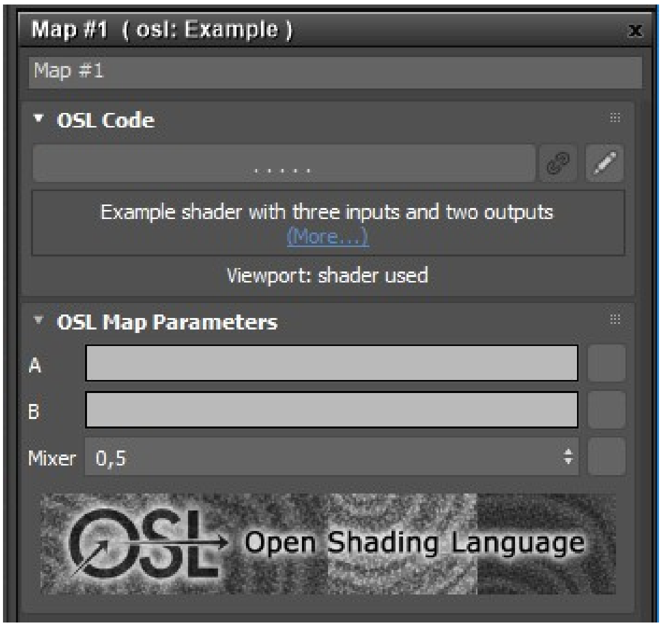
Getting started with OSL Map in the Scanline renderer

The renderer determines which materialas are available to you, so yo need to decide on the renderer first of all. In the **Render Setup** dialog, choose **Scanline Renderer** as the renderer for the mode **Production**.

Create a simple object, like a sphere. Open the Material Editor (M), and pick the Slate Material Editor rather than the Compact Material Editor . If you open the wrong one, you can change it through the Modes menu at the top left of the material editor window.

Right-click in the central pane of the Slate Material Editor, and create a new Material of type Materials → Scanline → Standard.

Now, right-click again on the background pane of the material editor, and create a new Map of type Maps → General → OSL Map. Note that the Maps → OSL subsection contains some other OSL examples and utilities, but they are all variations on the general OSL Map, only with different source code.



Map #1 (osl: Example)

Map #1

Example shader with three inputs and two outputs (More...)

Viewport: shader used

Mixer 0,5

Double click on the title bar for the newly created OSL Map pane, to get its parameter dialog into the rollout pane at the right hand side of the material editor. Click the pencil icon at the top right to open a text editor where you can edit the OSL source code.

Drag a connection from the output of the OSL Map to the Diffuse Color input of the Standard material, and drag a connection from the material output to the object in your scene, or assign it to the object by selecting the object and clicking Assign to Selection in the command pane at the top of the material editor window.

Click Show Shaded Material in Viewport to get a rough but useful preiew of your material in the viewport.

Now it’s time to start editing the OSL code. Let’s cut down on the inputs and the outputs and write a shader with a single `float` input and one `color` output, using Perlin noise in a very straightforward manner:

```
shader Example (
float size = 1.0,
output color Out = 0.0
)
{
    Out = noise(P/size);
}
```

Hit the Recompile button, and note that the interactive pane in the material editor and the settings rollout for the OSL Map changes accordingly.

Make a test render. Now you’re all set for engaging in your own OSL programming experiments!

Displacement maps in Scanline Renderer

There is a Displacement map slot in the Standard material but, for some reason, it doesn’t work with the Scanline renderer unless you also add a **Displace** modifier to the modifer stack. The displacement map shouldn’t actually be set in that Displace modifier, though. Instead, you just place the modifier in the stack for the object you wish to perform displacement mapping, and then use the **Displacement** slot in the Standard material to assign the map. This is strange and counter-intuitive, but it seems to be the only way to get displacement mapping to work with the Scanline renderer in the current release. (A bit surprisingly, the Arnold Properties modifier works for this purpose with the Scanline renderer as well, but this is an undocumented and possibly unintended feature, and it’s probably unwise to rely on it.)

In the Modify panel (top right, second tab), add the modifier **Displace** to your object from the list of modifiers. In the **Parameters** rollout, under **Image**, you find a map slot labeled **Map:** for displacement mapping**,** but leave that at its default state, **None**. Instead, drag a map connection in the material editor to the **Displacement** map slot of the Standard material. Now, because this displacement is performed by an object-space modifer, you also need to have enough polygons for the displacement to look nice. If your object has too few and too large polygons, you need to edit it to have more segments, or add a **Subdivide** or **TurboSmooth** modifier in the modifer stack, between your object and the Displace modifier. Take care not to add too much detail, or your object will take very long to generate and to modify with the displacement map, possibly even exhausting the memory available to 3dsMax and causing the program to crash.

It would be nice if this subdivision could be dynamic in some manner, to avoid tesselating more than necessary and to spend the effort where it’s needed instead of blindly increasing the amount of vertices and polygons everywhere on the object. Unfortunately, the Displace modifier doesn’t support this. (There is an **Adaptive** section the Displacement section of the **Arnold Properties** modifier, so adaptive tesselation was at least planned for Arnold. Unfortunately, those controls seem to be unresponsive in the current release.) If you find a way to get this working, please let us know, because dynamic adaptive tesselation is really helpful when working with displacement shaders. This is the first release of the OSL subsystem in 3dsMax, and adaptive tesselation is hopefully going to be properly implemented in a future release of the software, though possibly not for the outdated Scanline renderer.

That’s it. You’re now ready to start experimenting on your own!