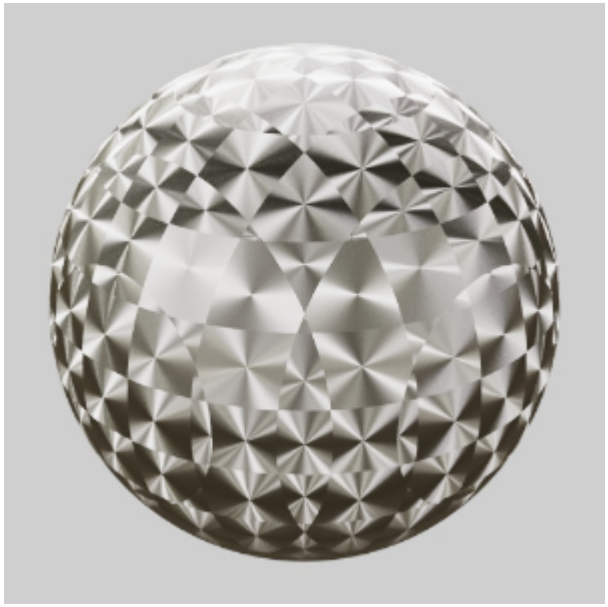
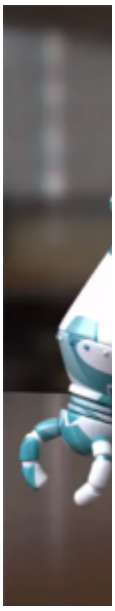
OSL Shaders





The OSL shaders used in the above examples can be found here.

It is possible to create your own shaders using Open Shading Language. OSL shaders can be used to implement anything from simple texture patterns to full materials using closures. They can be placed in the plugin search path, and will then be automatically loaded like other types of shaders. Once loaded, they can be inspected, instantiated, and linked in the same way as C++ shaders. The `osl` node provides an alternate way to load OSL shaders, which can be used to write shader code for a specific material. When setting the shader name or code, the parameters from the OSL shader automatically appear on the node to be set or linked.

More information about using OSL shaders can be found here.

A video that shows how to use the OSL shader can be found here.

An example .osl shader and .mtd file can be found here.

Limitations

- The shaders must have unique names. If the name conflicts with an existing shader, the OSL shader won't load. You should see a warning in the log in this case.
- The name of the output attribute is not the same as the shader code.
- Unable to do #includes in OSL shader code.
- No support for exposing input attributes of type Array and Matrix.

Multiple Outputs

Shader outputs can be linked to inputs of other shader nodes in an .ass file or programmatically via the *AiNodeLinkOutput()* function by optionally specifying which output is desired when linking. When no output is specified, the node's default output parameter will be chosen, preserving Arnold's existing behavior. Standard OSL language rules apply when specifying multiple OSL shader outputs. An example OSL shader with three color outputs coul d look like the following snippet and each output could be connected separately :

```
shader test_shader
 (
    output color result = color(1, 1, 1),
    output color good = color(0, 0, 1),
    output color bad = color(1, 0, 0)
 )
 {
 }
```

An example .ass file using multiple outputs can be downloaded here.

Installation

Just like any other third-party shader libraries, OSL shaders placed in the shader search path are automatically registered as Arnold shader nodes. The OSL shader parameters are converted to Arnold parameters. Once loaded, they can be inspected, instantiated and linked in the same way as C++ shaders.

To render OSL shaders directly with Arnold, you must do the following:

- Put the OSL shader (.osl and .mtd) in a folder, and set ARNOLD_PLUGIN_PATH to point to this folder. For example:

  ```
  ARNOLD_PLUGIN_PATH=C:\shaders\osl
  ```

- Arnold will automatically compile the .osl file and produce a dll (.oso) file.

To render OSL shaders directly in C4DtoA, you must do the following:

- Copy the **.osl** or **.oso** file to the ***$C4D_INSTALL/plugins/C4DtoA/shaders*** folder. All plugins located in this folder are read automatically by the plug-in. Meta data (**.mtd**) file placed next to the OSL shader is also loaded. If the shader ships with C4D UI resource files (**.res, .h, .str**) you have to copy them to the ***$C4D_INSTALL/plugins/C4DtoA/res*** folder.

⚠ Make sure you have write permission in the folder where the .osl file is located, because Arnold will compile and write the .oso file there.