Instancing Stand-ins

Similarly, SItoA allows for stand-ins driven by ICE attributes.

You should proceed as usual when instancing stand-ins on a pointcloud. Then, if the standin property has the string "**ArnoldProcedural**" set for the path and/or the data, then it will be exported by ICE as an Arnold procedural node. In ICE, any attribute starting with "ArnoldProcedural" is a candidate for setting the corresponding procedural parameter.

In the example shown below, we set **ArnoldProceduralDso** to point to an .ass file (depending if the point id is even or odd) and set **ArnoldProceduralData** to void. "Dso" and "Data" are the names of the parameters of the native Arnold procedural node. We can then set any of the procedural parameters, as long as we use the original name of the attribute (*kick -info procedural*) after ArnoldProcedural. For instance, ArnoldProceduralSelf_Shadows, etc., the same way we did for lights.

If Path is a valid dll (meaning that Data must be set equal to "ArnoldProcedural" to be managed by ICE), then the dll path is used, and you can just set **ArnoldProceduralData** in ICE to have the same dll using different data for different points.

If Deferred Loading is on, then the stand-in's bounding box is used (the asstoc file is not used, even though selected in the standin property). If you need to set or overwrite the min and max boundaries (for instance if using your own custom procedural), you can push two extra 3D vector attributes named **ArnoldProceduralMin** and **ArnoldProceduralMax**.

> Use a **Log Value** node in your ICE tree to force ICE to evaluate the ArnoldProcedural attributes. If you don't force the evaluation of the attributes, the attribute values won't be available at render time.
>
> Note that you can disable logging in the Log Value node, and ICE will still evaluate the attributes.

Also, while rendering, you may get some warnings from Arnold about names being reused; these warnings can be ignored.