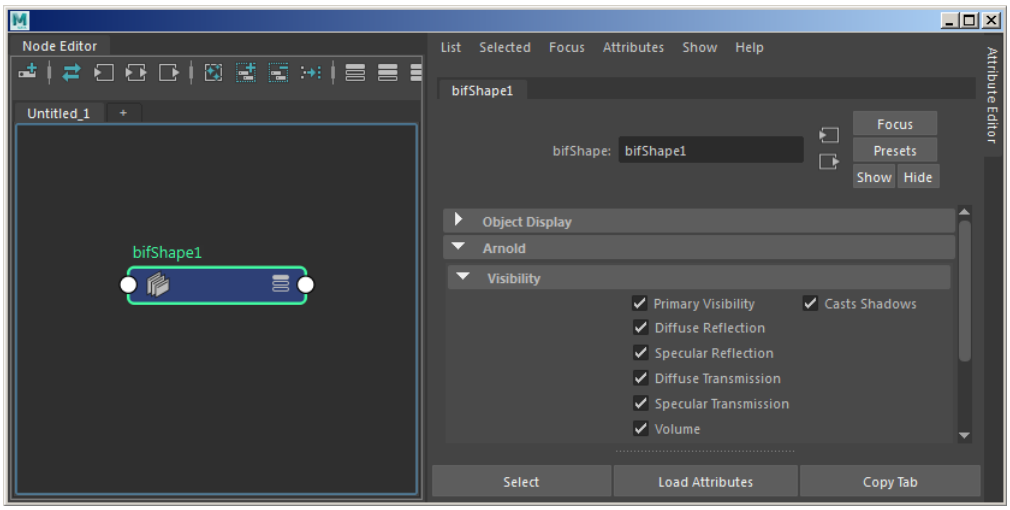
Bifrost Graph



Arnold *visibility* attributes found under the *bifShape* node

- The native Bifrost volume does not currently work with GPU. However, volume data exported as VDB should work.
- The custom property name can be anything as long as you pass the same name to the *aiUserDataColor* node in the material network being used. Using “my_color” is fine.
- The *color_jitter* shader does not currently work with Bifrost.
- The random color used here does not currently display in the viewport and will only show when rendering.

When a shader is overridden, e.g. a shader is applied to the bifShape or bifrostGraphShape in Maya, it gets applied to all of the Arnold shapes generated by arnold_bifrost. However, if the shader to be applied is not a volume shader, it should not apply it to volumes so that in-graph volume shaders still get applied (or the default generic volume shader that *arnold_bifrost* generates gets applied). If you create your own custom volume shader and assign it directly to the bifShape or bifrostGraphShape, *arnold_bifrost* will not recognize it. The solution, in that case, is to just apply the custom shader in-graph.

Bifrost Graph Editor

The following Arnold nodes are available in the *Bifrost Graph Editor*. Click on the link to go to the relevant Arnold attributes.

Bifrost Node
Arnold_ray_bitmask
set_Arnold_geo_settings
set_Arnold_mesh_settings
set_Arnold_points_settings
set_Arnold_strands_settings
set_Arnold_volume_settings

Exporting to Ass / Kick workflow

When exporting to a .ass file for use with kick, you must run kick with the location of the Arnold-Bifrost procedural using the -l parameter, for example:

```
C:\solidangle\mtoadeploy\2018\bin\kick.exe volume_scene.ass -l "C:\Program Files\Autodesk\Bifrost\Maya2018\2.0.1.0\bifrost\arnold-5.3.0.0"
```

Note that when using Bifrost Graph Volumes, to export ass and kick you need to not expand procedurals. In MtoA, this means in the .ass export options *Expand Procedurals* should be disabled.

User Data Support

Arnold-Bifrost will attempt to translate every Bifrost property to Arnold. For built-in properties that correspond to a known Arnold built-in parameter, they will be translated as such and not as user data; however, all other properties encountered will be translated as user data so that they are made available to shaders and operators to use as they see fit.

The type and scope of user data in Arnold depends on the type of object being rendered, e.g. *points* only support constant and uniform user data, while *polymesh* supports constant, uniform, varying and indexed user data. The following table shows which scopes are supported for which object types, and which types are supported overall:

Object Type	constant	uniform	varying	indexed
points	Yes	Yes		
curves (Bifrost strands)	Yes	Yes	Yes (per-segment, also auto-converted from per-control-point)	Yes (requires index/range property)
polymesh (Bifrost mesh)	Yes	Yes	Yes	
instance	Yes			
volume (Bifrost multiresolution tile tree)	Yes			
implicit (Bifrost signed distance field)	Yes			

The following are the types of Bifrost data supported, and their converted equivalent in Arnold. Note that in some cases the data is truncated or loses precision to fit the Arnold type, such as double-to-float conversions.

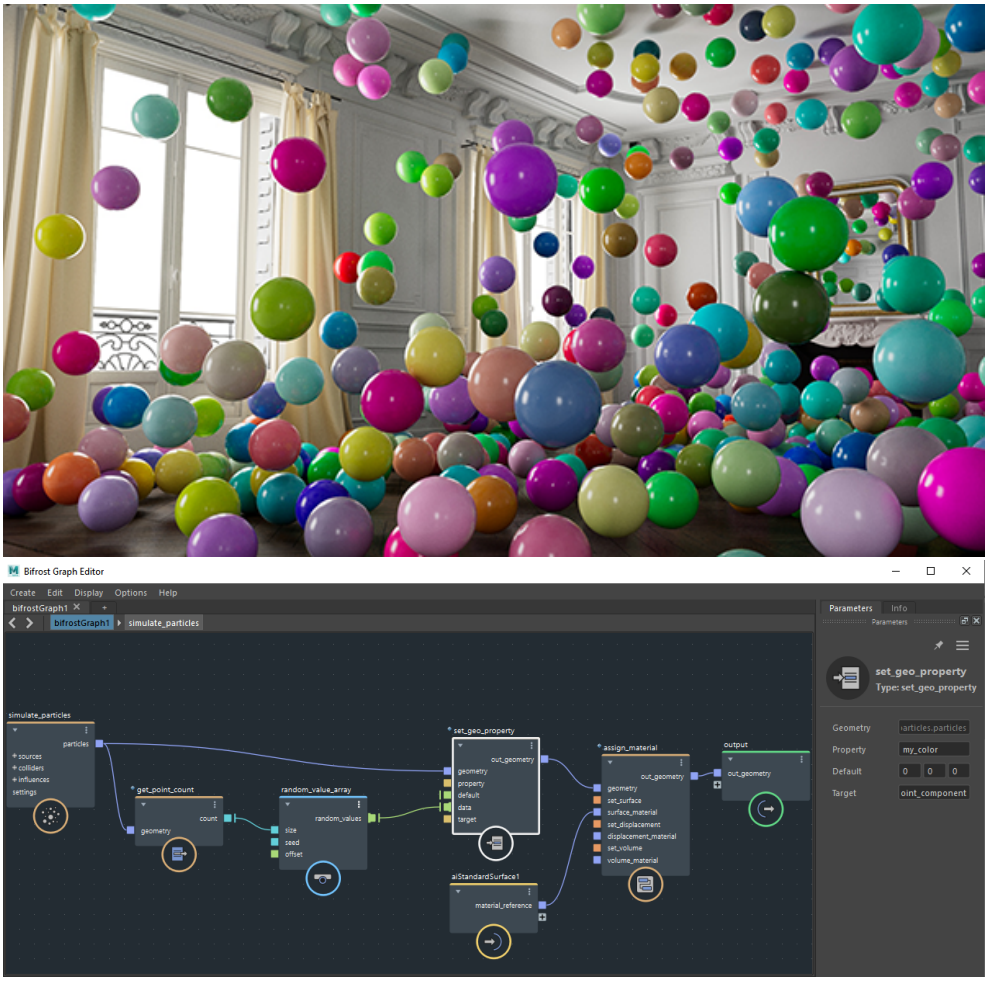
Bifrost Types	Arnold Component Type	Arnold Type
bool	bool	AI_TYPE_BOOLEAN
signed char, signed short int, signed int, signed long int	int32_t	AI_TYPE_INT
unsigned char	uint8_t	AI_TYPE_BYTE
unsigned short int, unsigned int, unsigned long int	uint32_t	AI_TYPE_UINT
float, double	float	AI_TYPE_FLOAT
string	AtString	AI_TYPE_STRING
char2, uchar2, short2, ushort2, int2, uint2, long2, ulong2, float2, double2	float	AI_TYPE_VECTOR2 (AtVector2)
char3, uchar3, short3, ushort3, int3, uint3, long3, ulong3, float3, double3	float	AI_TYPE_RGB (AtRGB)
char4, uchar4, short4, ushort4, int4, uint4, long4, ulong4, float4, double4	float	AI_TYPE_RGBA (AtRGBA)
char3x3, uchar3x3, short3x3, ushort3x3, int3x3, uint3x3, long3x3, ulong3x3, float3x3, double3x3	float	AI_TYPE_MATRIX (AtMatrix)
char4x4, uchar4x4, short4x4, ushort4x4, int4x4, uint4x4, long4x4, ulong4x4, float4x4, double4x4	float	AI_TYPE_MATRIX (AtMatrix)

- There are some Bifrost types not supported, such as the *type*2x2 or other non-square matrix variants.
- Note also that only the types with a *float* component on the Arnold side can be used with varying or indexed user data. String, integer, or boolean data cannot be interpolated, so can only be used with constant or uniform user data.

Randomize Particle Color Workflow

Below is an example of how to randomize the color of Bifrost particles.

A scene file can be downloaded here.



***random_value_array* -> new *set_geo_property (color)*. The color is a float3 or float4 property (RGB, RGBA).**