Procedurals in Katana

RenderMan Procedurals give you the opportunity to dynamically create geometry and then pass it to RenderMan while rendering. By passing geometry in to RenderMan at the very last second, you can save the time and memory it would take to generate and display the geometry in Katana, Maya, or other application.

Procedurals are most commonly used in fur/hair rendering, crowd rendering, dynamically grown vegetation, or procedurally created environments. They can be used for anything.

While procedurals have many advantages, one of the disadvantages derives from its strength. Because the geometry is generated only within RenderMan, Katana doesn't know anything about the geometry and can't display it. If the application you are using to create the procedural has some sort of static representation, such as an Alembic file, we recommend that you use the Katana Viewer proxy feature so that you can see something other than a bounding box in the Viewer. See the Katana Help for how to use Viewer proxies.

Setting Up a Procedural in the Node Graph

There are three main steps that you must do in order to tell Katana to properly pass a procedural to RenderMan.

- Set up a Scene Graph location of type "renderer procedural".
- Set the path of the shared library that contains the procedural code that RenderMan will call.
- Set up any additional arguments that the procedural requires in order to do its work.

Below each of the steps is detailed.

Step 1: Set up a Scene Graph location of type "renderer procedural"

Within the Scene Graph, the "type" of each location controls how the Attributes for that location are passed to RenderMan. For RenderMan to call the Procedural at runtime, you must add a Scene Graph location of type "renderer procedural".

The easiest way for you to do this is to use the *PrimitiveCreate* node and set the type to "renderer procedural". Specify the name as well. In the example below, I have set it to /root/world/geo/procedurals/myprocedural

💿 Parameters	Catalog							(•
🔻 PrimitiveCreate	e_rendererP	rocedural 🔳			₽	Ŧ	۶		٩
-	name 🛇	/root/world/g	eo/procedurals/n	nyprocedural	 				•
type				👻 renderer procedural					

Step 2: Set the path of the shared library that contains the procedural code that RenderMan will call

Developers of the application that you created your procedural geometry within will provide you with a RenderMan Procedural library. Consult with the documentation for that application or consult your pipeline team for where you can find this path. Below, I have specified a path to an imaginary library. Use an *AttributeSet* node to do this. The attribute name to use is: rendererProcedural.procedural

O Parameters Cat	alog	. 8		
< ⊳				
▼ AttributeSet_proce	duralLocation AttributeSet	• • • • • •		
🔹 mode р	aths	-		
🤋 🔻 paths (1) 🕲		Add Locations 👻 🔍		
🗉 🗿 /root/world/ge	o/procedurals/myprocedural	▼ 🗄		
action <u>C</u>	create/Override	.		
attributeName	enderer Procedural.procedural			
attributeType s	tring	–		
groupInherit Y	és	T		
⑦ ▼ stringValue		1 × 1		
/path/to/my/Procedural.so				
Drop Attributes Here				

Step 3: Set up any additional arguments that the procedural requires in order to do its work

Each procedural will have a unique set of parameters that must be passed to it in order for the procedural to generate the correct geometry for the frame you are rendering. These parameters are likely to be similar to what you would specify to a command-line utility.

Below I have specified some imaginary attributes using an *AttributeSet* node that the imaginary Procedural.so needs in order to execute properly. The attribute name to use is: rendererProcedural.args.extraArgs

Parameters Catalog				
AttributeSet_proceduralArgs AttributeSet	0 ¢ T	۶ ۶ ۹		
mode paths		~		
⑦ ▼ paths (1) ⑤	Add Lo	ocations 👻 🔍		
🗉 💿 /root/world/geo/procedurals/myprocedural		▼ 🗄		
② action Create/Override		T		
attributeName rendererProcedural.args.extraArgs				
attributeType string		T		
groupInherit Yes		T		
© ▼ stringValue		1 x 1		
-myparam1 value1 -myparam2 value2 -myparam3 value3				
Drop Attributes Here				

After you complete each of these steps above, you will end up with a Node Graph, Scene Graph, and Attributes similar to the example below.

PrimitiveCreate_rendererProcedural AttributeSet_proceduralLocation AttributeSet_proceduralArgs AttributeSet_proceduralArgs AttributeSet_proceduralArgs Name Type AttributeSet_proceduralArgs Name Type a AttributeSet_proceduralArgs Name Type a MatributeSet_proceduralArgs Name Type a AttributeSet_proceduralArgs AttributeSet_proceduralArgs AttributeSet_proceduralArgs AttributeSet_proceduralArgs AttributeSet_proceduralArgs AttributeSet_proceduralArgs AttributeSet_proceduralArgs AttributeSet_proceduralArgs AttributeSet_procedural AttributeSet_proc					
AttributeSet_proceduralLocation AttributeSet_proceduralArgs AttributeSet_proceduralS AttributeSet_proceduralS AttributeSet_proceduralS AttributeS AttributeSet_proceduralS AttributeSet_proceduralS AttributeSet_proceduralS AttributeSet_proceduralS AttributeSet_proceduralS AttributeSet_proceduralS AttributeSet_procedural AttributeSet_procedura	PrimitiveCreate_rendererProcedural				
AttributeSet_proceduralLocation AttributeSet_proceduralArgs AttributeSet_proceduralArgs AttributeSet_proceduralArgs AttributeSet_proceduralArgs AttributeSet_proceduralArgs Name Type Foroot group Foroot grou					
AttributeSet_proceduralArgs Scene Graph Project Settings Python Monitor Scene Graph Project Settings Python Monitor AttributeSet_proceduralArgs Name Type Foroot group geo group geo group geo group forocedurals renderer procedural I on Attributes Render Log Viewer froot/world/geo/procedurals/myprocedural renderer procedural root/world/geo/procedurals/myprocedural renderer procedural procedural renderer Procedural procedural /path/to/my/Procedural.so extraArgs -myparam1 value1 -myparam2 value2 -myparam3 value3 useInfiniteBounds Yes	AttributeSet_proceduralLoc	cation			
 AttributeSet_proceduralArgs Scene Graph Project Settings Python Monitor AttributeSet_proceduralArgs AttributeSet_proceduralArgs Name Type root group y or ot group group procedurals group (root/world/geo/procedurals/myprocedural renderer procedural renderer Procedural procedural procedural<!--</th--><th>Α</th><th></th>	Α				
Scene Graph Project Settings Python Monitor Mame Type Name Type root group geo group group group geo group group group myprocedural renderer procedural (root/world/geo/procedurals/myprocedural materialAssign materialAssign renderer procedural procedural procedural (root/world/geo/procedurals/myprocedural renderer procedural (root/world/geo/procedurals/myprocedural renderer procedural (root/world/geo/procedurals/myprocedural renderer procedural (renderer procedural renderer procedural (renderer procedural renderer procedural (rendererProcedural rendererProcedural (rendererProcedural (r	AttributeSet_proceduralA	rgs 📕			
Scene Graph Project Settings Python Monitor Mame Type root group geo group geo group group group geod group renderer procedural renderer procedural renderer procedural materialAssign rendererProcedural	· · · · · · · · · · · · · · · · · · ·				
Name Type Image: I	Scene Graph Project Settings	Python Monitor			
Name Type * root group * geo group * geo group * procedurals group * myprocedural renderer procedural * renderer procedural * renderer procedural * renderer procedural * renderer procedural * renderer Procedural * renderer Procedural * renderer Procedural * renderer Procedural * renderer Procedural * renderer Procedural * renderer Procedural * renderer Procedural * renderer Procedural * renderer Procedural * renderer Procedural * renderer Procedural * uselnfiniteBounds Yes	🚫 < ▷ 🔳 AttributeSet_procedura	alArgs 🗖			
 root group group geo group procedurals group Procedurals group Attributes Render Log Viewer Attributes Render Log Viewer froot/world/geo/procedurals/myprocedural renderer procedural renderer procedural procedural procedural procedural group group group group group group group group group procedurals/myprocedural group group	Name	Туре			
 world group geo group procedurals group myprocedural renderer procedural Attributes Render Log Viewer Attributes Render Log Viewer /root/world/geo/procedurals/myprocedural /root/world/geo/procedurals/myprocedural renderer procedural materialAssign prmanStatements rendererProcedural procedural /path/to/my/Procedural.so args extraArgs -myparam1 value1 -myparam2 value2 -myparam3 value3 uselnfiniteBounds Yes 		group			
Procedurals group myprocedural renderer procedural Attributes Render Log Viewer froot/world/geo/procedurals/myprocedural renderer procedural materialAssign © prmanStatements rendererProcedural procedural /path/to/my/Procedural.so rendererProcedural extraArgs -myparam1 value1 -myparam2 value2 -myparam3 value3		group group			
Attributes Render Log Viewer * renderer procedurals/myprocedural * renderer procedural materialAssign * rendererProcedural * rendererProcedural * rendererProcedural * rendererProcedural * rendererProcedural * rendererProcedural * uselnfiniteBounds Yes		group			
 Attributes Render Log Viewer /root/world/geo/procedurals/myprocedural renderer procedural materialAssign prmanStatements rendererProcedural procedural /path/to/my/Procedural.so extraArgs -myparam1 value1 -myparam2 value2 -myparam3 value3 useInfiniteBounds Yes 	🗠 myprocedural	renderer procedural			
 Attributes Render Log Viewer /root/world/geo/procedurals/myprocedural renderer procedural materialAssign prmanStatements rendererProcedural procedural /path/to/my/Procedural.so args extraArgs -myparam1 value1 -myparam2 value2 -myparam3 value3 useInfiniteBounds Yes 					
/root/world/geo/procedurals/myprocedural renderer procedural materialAssign prmanStatements rendererProcedural procedural /path/to/my/Procedural.so rargs extraArgs myparam1 value1 -myparam2 value2 -myparam3 value3 useInfiniteBounds Yes	Attributes Render Log Viewer				
 renderer procedural materialAssign prmanStatements rendererProcedural procedural /path/to/my/Procedural.so args extraArgs -myparam1 value1 -myparam2 value2 -myparam3 value3 uselnfiniteBounds Yes 	/root/world/geo/procedurals/myproce	dural			
 materialAssign prmanStatements rendererProcedural procedural /path/to/my/Procedural.so args extraArgs -myparam1 value1 -myparam2 value2 -myparam3 value3 useInfiniteBounds Yes 	▼renderer procedural				
 prmanStatements rendererProcedural procedural /path/to/my/Procedural.so args extraArgs -myparam1 value1 -myparam2 value2 -myparam3 value3 useInfiniteBounds Yes 	💿 materialAssign 🛇				
 rendererProcedural procedural /path/to/my/Procedural.so args extraArgs -myparam1 value1 -myparam2 value2 -myparam3 value3 useInfiniteBounds Yes 	▶ prmanStatements				
procedural /path/to/my/Procedural.so srgs extraArgs -myparam1 value1 -myparam2 value2 -myparam3 value3 useInfiniteBounds	🔲 🔻 rendererProcedural				
 args extraArgs -myparam1 value1 -myparam2 value2 -myparam3 value3 useInfiniteBounds Yes 	procedural /path/to/my/Proce	edural.so			
<pre>extraArgs -myparam1 value1 -myparam2 value2 -myparam3 value3</pre> I useInfiniteBounds Yes	🛯 🔻 args				
🗈 useInfiniteBounds Yes	🕒 extraArgs -myparam1 value1 -myparam2 value2 -myparam3 value3				
	🖸 useInfiniteBounds Yes				
D includeCameraInfo None	IncludeCameraInfo None				
□ ▷ tabs	▶ tabs				
Deviewer	▶ viewer				
	□ ► xform				
	□ ►xform				
L ⊳xform					

Going Beyond the Basics

The information below only provides you with the most basic way to insert a RenderMan Procedural into your Katana scene. There are other more advanced options that you or your pipeline team can pursue. For each of the methods below, your mileage may vary. Take a look at the Katana documentation available from the Help menu within Katana for more details on each of the methods below.

- Create + use Args files for renderer procedurals to make it easier to specify the parameters that the procedural needs to do its work.
 Create a custom node that wraps the procedural the ultimate in user-friendliness, but requires a developer.