PrmanObjectStatements

The PrmanObjectStatements is assigned to a specific location or group of locations and translates into prman as setting the state for an `AttributeBegin` `/AttributeEnd` block. This node allows you to do things like assign a subdivision approximation to objects to render smooth surfaces, apply a displacement bound for rendering displacements, and changing visibility flags and memberships.
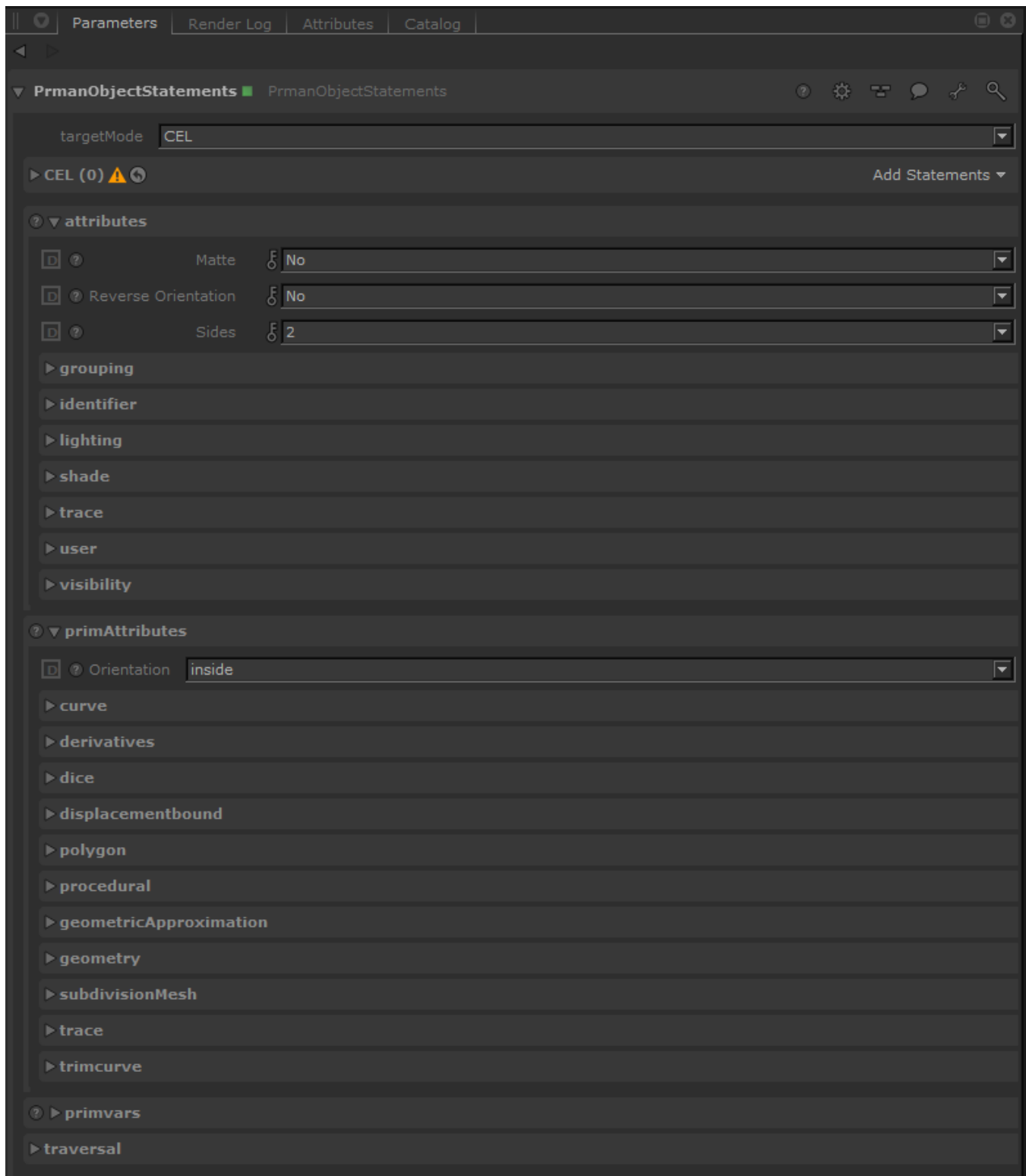
In RenderMan, there are different ways to choose how your object and its instances will render. These are conveniently divided in the node:

Section	Description
`attributes`	Attributes which can be varied across instances (e.g. `lighting subset` or `camera visibility`)
`primAttributes`	Per-primitive attributes which are considered "prototype" properties (referred to in the table below as "Primitive Variables")

`primvars`	Constant per-primitive attributes which are part of the primitive definition. These cannot be varied per-instance.
`traversal`	RfK plugin controls for scene traversal.

The settings are listed in the tables below. Note that the tables do not currently line up exactly with the PrmanObjectStatements UI presentation of the parameters.

Attributes

Common

Name	Type	Default	Description
bake: activeudims	*string*	*empty*	Active udims for Integrator baking of texture atlases. Lists and ranges of udims can be provided in the string, for example: Attribute "bake" "string activeudims" "1001,1003,1005-1006". The default is "" which means all udims get baked.
bake: udimresolutions	*string*	*empty*	Udim texture resolutions for Integrator baking of texture atlases. Lists and ranges of udims can be provided in the string, for example: Attribute "bake" "string udimresolutions" "1001 512 512 1002,1003 64 32 1005-1006 16 16" If the string is empty (the default) then the Display Format resolution will be used.
grouping: id	*int*	0	Grouping id used for raytrace disambiguation purposes. Value range: 0 to 15.
grouping: membership	*string*	*empty*	Grouping membership used for linking.
identifier: id	*int*	0	Lower 32 bits of 64 bit object id.
identifier: id2	*int*	0	Upper 32 bits of 64 bit object id.
identifier: lpegroup	*string*	*empty*	Grouping membership used for light path expressions.
identifier: name	*string*	*empty*	Object instance name.
stats: identifier	*string*	*empty*	Identifier string used by the stats system when referring to this instance.

Shading

Name	Type	Default	Description
curves: widthaffectscurvature	*int*	1	Indicates whether curve width should be used when computing curve curvature. If turned off, only the curvature along the curve spline will be calculated.
lightfilter:subset	*string*	*empty*	Active light filter groups in lighting.
lighting:excludesubset	*string*	*empty*	Exclude light groups from lighting.
lighting:mute	*int*	0	Mutes light emission.
lighting:subset	*string*	*empty*	Light groups active in lighting.
Ri:Matte	*int*	0	Indicates matte object.
Ri:ReverseOrientation	*int*	0	Flip shading normals.
Ri:Sides	*int*	2	Geometry can be one or two sided.
shade: indexofrefraction	*float*	-1	Override material IOR for nested dielectrics.
shade:minsamples	*int*	1	Min Pixel Samples. For fine-tuning adaptive sampling.
shade: relativepixelvariance	*float*	1	PixelVariance multiplier for camera visible objects. For fine-tuning adaptive sampling.
trace:holdout	*int*	0	Indicates holdout object.
trace:intersectpriority	*int*	0	Raytrace intersection priority for nested dielectrics. Value range: 0 to 31.
trace:maxdiffusedepth	*int*	1	Maximum diffuse light bounces.
trace: maxspeculardepth	*int*	2	Maximum specular light bounces.

trace: reflectexcludesubset	*string*	*empty*	Exclude object groups from relections.
trace:reflectsubset	*string*	*empty*	Object groups visible to reflections.
trace: shadowexcludesubset	*string*	*empty*	Exclude object groups from casting shadows.
trace:shadowsubset	*string*	*empty*	Object groups active in shadows.
trace: transmitexcludesubset	*string*	*empty*	Exclude object groups from refractions.
trace:transmitsubset	*string*	*empty*	Object groups visible to refractions.
visibility:camera	*int*	1	Indicates if object is visible to camera rays.
visibility:indirect	*int*	1	Indicates if object is visible to indirect (reflection and refraction) rays.
visibility:transmission	*int*	1	Indicates if object is visible to shadow rays.

Link

PrimAttributes

> ⚠ These may also be referred to as "prototype attributes" for clarity versus Instance Attributes which may be varied per instance.

Common

Name	Type	Default	Description
identifier:object	*string*	*empty*	Object shape name.
stats:prototypeIdentifier	*string*	*empty*	Identifier string used by the stats system when referring to this prototype.

Shading

Name	Type	Default	Description
derivatives: extrapolate	*integer*	1	Extrapolated smooth normals across grid boundaries.
displacement: ignorereferenceinstance	*int*	0	Indicates if displacement shading should ignore properties of the reference instance.
displacementbound: CoordinateSystem	*string*	object	The name of the coordinate system that the displacement bound is measured in.
displacementbound: offscreen	*int*	0	Apply displacementbound to offscreen geometry when dicing. Geometry that is entirely offscreen, but displaces into view may be prone to undertessellation because of the offscreen dicing strategy. This can be corrected by enabling this control and setting a reasonable displacementbound. Doing so will incur a performance penalty commensurate with the size of the displacement bound.
displacementbound: sphere	*float*	0	Maximum displacement distance used to compute object bounds. This value should be as tight as possible.
Ri: Orientation	*string*	outside	Geometry winding order that determines surface normal. This attribute effects lighting and displacement. Supported values: inside, outside.
trace: autobias	*integer*	1	Enable automatic raytrace bias.
trace:bias	*float*	0.01	Manual raytrace bias in object space. Small values should be used to avoid artifacts such as shadow acne.
trace: displacements	*integer*	1	Enable displacement shading.
trace: sssautobias	*integer*	1	Enable automatic raytrace bias for subsurface scattering rays.
trace:sssbias	*float*	1e-4f	Manual raytrace bias (in object space) for subsurface scattering rays. Small values should be used to avoid artifacts such as surface acne.

Dicing

Name	Type	Default	Description
dice: micropolygonlength	float	1	Micropolygon distance in raster space for "instanceprojection" dicing. Values are expressed in pixel size.
dice: offscreenmultiplier	float	-1	Dicing rate multiplier for objects outside the viewing frustum. If a value less than zero is specified, the value from the Option with the same name will be used instead. Value range: -1 or more.
dice: offscreenstrategy	string	viewfrustumdistance	Dicing method of objects outside the viewing frustum. Supported values: viewfrustumdistance, worlddistance, objectdistance, planarprojection.
dice: rasterorient	integer	1	Turning this off enables non-oriented dicing, a mode of dicing that computes micropolygon sizes using non-oriented raster space rather than screen-aligned raster space. Non-oriented dicing is useful for displacement around edges where projected micropolygon distance can be arbitrarily small.
dice: referencecamera	string	empty	Specify the camera used for dicing. If no reference camera is specified, RenderMan will use the primary camera.
dice: referenceinstance	string	empty	Specify the reference instance used for dicing and displacement shading. The reference instance is used to drive dicing and displacement based on its position, scale, user attributes, and scoped coordinate systems. Reference instances are specified by its identifier:name attribute (instance name). If no reference instance is specified, RenderMan will automatically pick the nearest instance inside the primary camera's view frustum.
dice: strategy	string	instanceprojection	Dicing method of objects within the viewing frustum. Supported values: instanceprojection, worlddistance, objectdistance.
dice: worlddistancelength	float	-1	Micropolygon distance in world space for "worlddistance" dicing or object space for "objectdistance" dicing.
Ri: GeometricApproximationFocusFactor	float	0	Allows the renderer to use more coarse dicing for blurry objects due to depth of field.

Points

Name	Type	Default	Description
falloffpower	float	0	For use with points, if not supplied, or set to zero, the points will have a hard edge. A value of 1 is a "reasonable" value that emulates the usual cosine based falloff; this will likely be the goto value for most people doing volumetric particle effects. Values between 0 and 1 makes the falloff faster, eroding the point faster - point has "less presence". Values higher than 1 (up to infinity) makes the falloff slower to the point of being non-existent.

Curves

Name	Type	Default	Description
curve: opacitysamples	int	0	number of opacity/presence shading samples along a curve for opacity caching, if equal to zero then the value is determined using the dicing oracle and the length of each curve.
curve: widthaffectscurvature	int	1	When true the curve width of round curves is taken into account in the computation of the tube curvature, otherwise only the curvature along the curve is. This control does not affect curve ribbons. Value range: true/false.

Volume

Name	Type	Default	Description
dice: minlength	float	-1	Volume minimum dice length. Negative indicates to automatically compute this value.
dice: minlengthspace	string	empty	Coordinate space of dice:minlength.
Ri: Bound	float[6]	0 0 0 0 0 0	Volume bounds.
volume: aggregate	string	globalVolumeAggregate	The name of the volume aggregate to which a volume belongs. If non-empty, the volume will only be rendered if the integrator explicitly uses that named aggregate, or if a surface shader such as PxrSurface binds the named aggregate to its interior.

volume: aggregaterespectvisibility	int	empty	If set to 1, standard visibility attributes will be respected by the volume if it is part of an aggregate. If set to 0, the visibility attributes will be ignored and only the integrator visibility settings will apply. Value range: true/false.
volume: dsominmax	integer	empty	Currently only used for aggregate volumes, and only for volumes that use an ImplicitField DSO. If set to 1, the DSO may be able to use stored information from the file directly to compute the minimum and maximum values within the volume. This may allow the renderer to avoid shading the volume up front, greatly decreasing time to first pixel. This can only be enabled if the density signal from the volume is used directly, or if the density signal is modulated only by the DSO itself. Any shading modifications of the density signal requires setting this parameter to off. Value range: true/false.
volume: dsovelocity	integer	0	Used only for aggregate volumes that use an ImplicitField DSO. If set to 1, the DSO can provide velocity bounds to the renderer directly via the BoxMotion call. This can allow the renderer to avoid upfront shading of the volume's velocity data when using Eulerian velocity, improving render time and time to first pixel greatly. This can only be enabled if the velocity signal from the DSO is used directly. Value range: true/false.
volume: fps	float	1	The frames per second for volumetric velocity data. The velocity data will be divided by this quantity to derive the velocity for the frame.
volume: shutteroffset	float	1	The shutter offset used to interpret volumetric velocity. A value of 1 will use the current position of the object and the position of the object on the next frame as the time interval to use for motion blur. A value of -1 will use the position of the object on the previous frame and the current position of the object as the time. A value of 0 will generate an interval which begins halfway through the previous frame and ends halfway into the next frame.
volume: temporalmethod	integer	0	Method of generating temporal data for volume rendering. Supported values: Eulerian (0), Reves (1).
volume: velocityshuttercorrection	int	0	If set to 1, volumetric velocity data for Eulerian motion blur will be assumed to be relative to the entire frame, and will be corrected to match the current shutter interval. If set to 0, no correction will be made. Value range: true/false.

SubdivisionMesh

Name	Type	Default	Description
dice:pretessellate	integer	1	Pre-tessellate subdivision geometry to polygons.
dice:watertight	integer	0	Tessellate geometry with no holes. Watertight geometry requires less raytrace bias.
shade:faceset	integer[n]	empty	Active geometry face indexes.
stitchbound:CoordinateSystem	string	empty	
stitchbound:sphere	float	0	

NuPatch

Name	Type	Default	Description
trimcurve:sense	string	inside	Supported values: inside, outside.

PolygonMesh

Name	Type	Default	Description
polygon:concave	integer	1	Allow concave polygons.
polygon: smoothdisplacement	integer	0	Output smoothed (per-vertex) normals as Ndsp primvar, if polygon:smoothnormals hasn't already inserted smooth normals.
polygon:smoothnormals	integer	0	Smooth (per-vertex) normals if not provided.

Procedural

Name	Type	Default	Description
procedural: immediatesubdivide	integer	0	
procedural:reentrant	integer	0	

Link

Primvars

Name	Type	Default	Description
falloffpower	*float*	0	Falloff power for point primitives.

Traversal

As RfK traverses the Katana scene graph it will process locations in parallel and pass them to RenderMan for rendering. By default each new child location retrieved from Katana will spawn a new thread for processing. This default behavior is unlikely to give the optimal performance for any given scene, however PrmanObjectStatements includes settings for tuning the efficiency and the level of parallelism for a particular pipeline (e.g. your studio's standard Op Chain) or various types of scenes (e.g. many instances, many textures, many primitives, or perhaps all three).
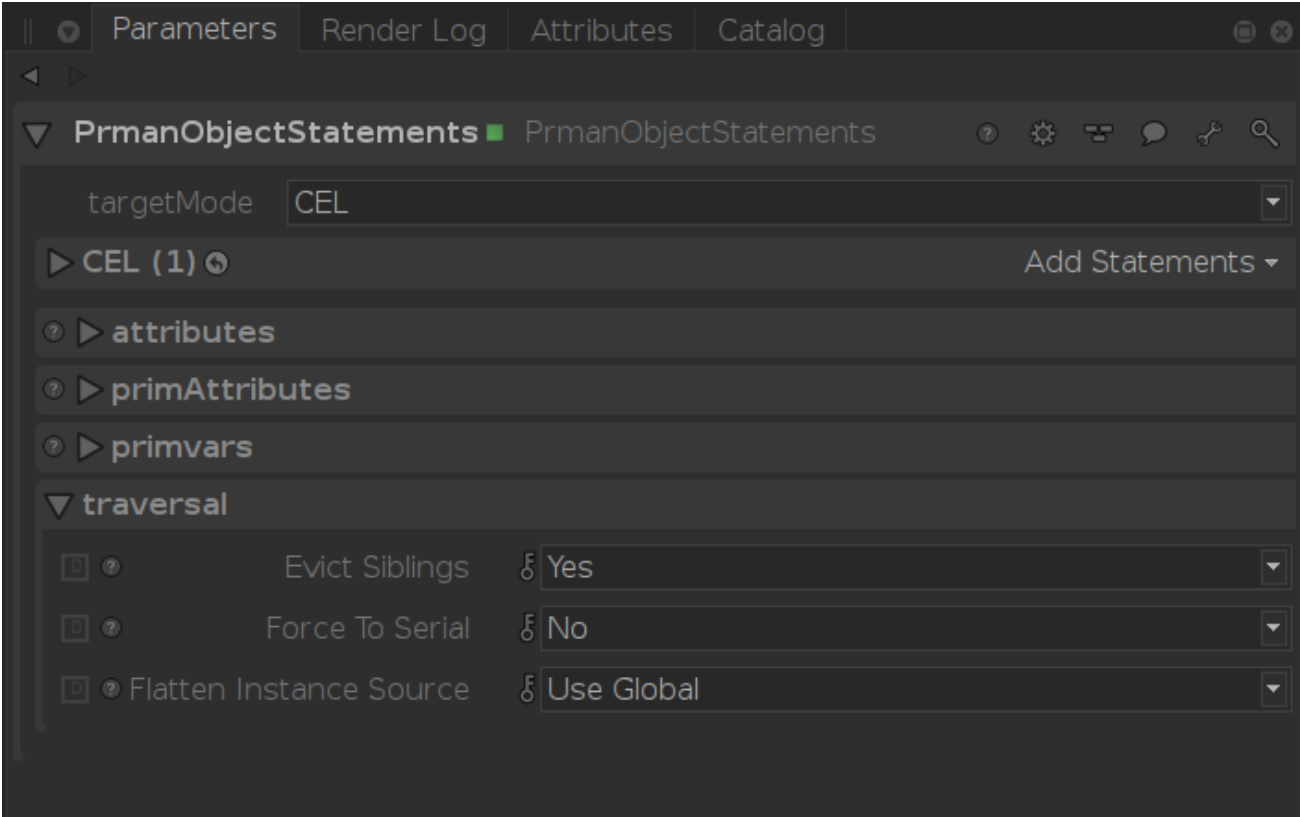
Name	Type	Default	Description
forceSerial	*int*	0	Force traversal to serial processing for this and child locations.
evictHere	*int*	1	Prevent sibling eviction at this location.
flattenInstanceSource	*int*	2	Flatten hierarchical instance source locations.

Force Serial

The first tool is an attribute to control the depth of parallelism in your scene: `forceSerial`.

For example, take a scene:

```
/root/world/geo/pine_tree
/root/world/geo/elm_tree
/root/world/geo/oak_tree
```

with each tree location containing branches, leaves, roots, etc.

Ideally you traverse this in parallel at a high-enough level that it's not trying to do every leaf or every branch on a different thread. We do this in RfK by setting "prmanStatements.traversal.forceSerial" to `True` at the location where we want to stop processing in parallel. When that attribute is encountered during traversal RfK will switch to serial at that location and all child locations will be run in a single thread.

In the above example, setting "forceSerial" to true on "/root/world/geo/*_tree" means that each tree will be on a different thread, but all the contents of each tree will be serially processed in their respective threads. You'll need to evaluate your scene setup to see at what level it makes sense to force serial processing.

Each thread has it's own Geolib runtime containing per-thread caches of cooked locations. As a result all locations needed for any Op would to be cooked per-thread. Continuing with the above example: if you have three threads each looking at different locations under "/root/world/geo", you'll see "/root" cooked three times, "/root/world" cooked three times and "/root/world/geo" cooked three times. If you are running serial, or forcing the parallel traversal to one thread at "/root" then you'll only have one Geolib runtime cooking the location and so would see only a single print from that Op at "/root"

Evict Here

The second tool we have for parallel traversal efficiency is the `evictHere` setting.

By default siblings of a Katana scenegraph location will be evicted from a thread's runtime memory when a location is processed for conversion to the Rix scenegraph. Setting this flag false prevents sibling eviction at this location in the event it is known that those siblings are best kept in the cache for subsequent access.

Flatten Instance Source

The last tool is the `flattenInstanceSource` setting.

Rather than changing the behavior to RfK's parallel traversal of the Katana scene, this setting improves RenderMan's traversal when building instances. By default, RenderMan flattens an instance source into separate object prototypes for each geometric primitive in the group. This can get expensive as the number of instances increases or as the number of primitives inside the instance source increases. Setting this option to `Yes` will tell RenderMan to keep this instance source location as a single object prototype.

This setting will override the global setting for prmanGlobalStatements.plugin.flattenInstanceSources - which can enable this setting on all instance source locations.