Gaffer Node Reference

image engine
version 0.87.0, 2014

Gaffer

Backdrop

A utility node which allows the positioning of other nodes on a coloured backdrop with optional text. Selecting a backdrop in the ui selects all the nodes positioned on it, and moving it moves them with it.

Plugs:

description

`Gaffer::StringPlug`

Text describing the contents of the backdrop - this will be displayed below the title.

scale

`Gaffer::FloatPlug`

Controls the size of the backdrop text.

title

`Gaffer::StringPlug`

The title for the backdrop - this will be displayed at the top of the backdrop.

Box

!!!*EMPTY*!!!

ContextVariablesComputeNode

!!!*EMPTY*!!!

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

variables

`Gaffer::CompoundDataPlug`

!!!*EMPTY*!!!

ExecutableOpHolder

!!!*EMPTY*!!!

Plugs:

despatcherParameters

`Gaffer::CompoundPlug`

!!!*EMPTY*!!!

requirement

`Gaffer::Plug`

!!!*EMPTY*!!!

requirements

`Gaffer::ArrayPlug`

!!!*EMPTY*!!!

Expression

!!!*EMPTY*!!!

Plugs:

engine

`Gaffer::StringPlug`

!!!*EMPTY*!!!

expression

`Gaffer::StringPlug`

!!!*EMPTY*!!!

ObjectReader

!!!*EMPTY*!!!

Plugs:

fileName

`Gaffer::StringPlug`

!!!*EMPTY*!!!

out

`Gaffer::ObjectPlug`

!!!*EMPTY*!!!

parameters

`Gaffer::CompoundPlug`

!!!*EMPTY*!!!

ObjectWriter

!!!*EMPTY*!!!

Plugs:

despatcherParameters

`Gaffer::CompoundPlug`

!!!*EMPTY*!!!

fileName

`Gaffer::StringPlug`

!!!*EMPTY*!!!

in

`Gaffer::ObjectPlug`

!!!*EMPTY*!!!

parameters

`Gaffer::CompoundPlug`

!!!*EMPTY*!!!

requirement

`Gaffer::Plug`

!!!*EMPTY*!!!

requirements

`Gaffer::ArrayPlug`

!!!*EMPTY*!!!

requirement0

`Gaffer::Plug`

!!!*EMPTY*!!!

OpHolder

!!!*EMPTY*!!!

Plugs:

ParameterisedHolderDependencyNode

!!!*EMPTY*!!!

Plugs:

Preferences

!!!*EMPTY*!!!

ProceduralHolder

!!!*EMPTY*!!!

Plugs:

output

`Gaffer::ObjectPlug`

!!!*EMPTY*!!!

Random

!!!*EMPTY*!!!

Plugs:

baseColor

`Gaffer::Color3fPlug`

!!!*EMPTY*!!!

contextEntry

`Gaffer::StringPlug`

!!!*EMPTY*!!!

floatRange

`Gaffer::V2fPlug`

!!!*EMPTY*!!!

hue

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

outColor

`Gaffer::Color3fPlug`

!!!*EMPTY*!!!

outFloat

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

saturation

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

seed

`Gaffer::IntPlug`

!!!*EMPTY*!!!

value

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

Reference

!!!*EMPTY*!!!

Plugs:

fileName

`Gaffer::StringPlug`

!!!*EMPTY*!!!

ScriptNode

!!!*EMPTY*!!!

Plugs:

fileName

`Gaffer::StringPlug`

!!!*EMPTY*!!!

frameRange

`Gaffer::CompoundPlug`

!!!*EMPTY*!!!

unsavedChanges

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

variables

`Gaffer::CompoundDataPlug`

!!!*EMPTY*!!!

SwitchComputeNode

!!!*EMPTY*!!!

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

index

`Gaffer::IntPlug`

!!!*EMPTY*!!!

TimeWarpComputeNode

!!!*EMPTY*!!!

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

offset

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

speed

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

GafferArnold

ArnoldAttributes

The base type for nodes that apply attributes to the scene.

Plugs:

attributes

`Gaffer::CompoundDataPlug`

The attributes to be applied - arbitrary numbers of user defined attributes may be added as children of this plug via the user interface, or using the CompoundDataPlug API via python.

cameraVisibility.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

shadowVisibility.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

reflectedVisibility.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

refractedVisibility.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

diffuseVisibility.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

glossyVisibility.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

subdivIterations.value

`Gaffer::IntPlug`

!!!*EMPTY*!!!

subdivPixelError.value

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

subdivAdaptiveMetric.value

`Gaffer::StringPlug`

!!!*EMPTY*!!!

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

filter

`Gaffer::IntPlug`

The filter used to control which parts of the scene are processed. A Filter node should be connected here.

in

`GafferScene::ScenePlug`

The input scene.

out

`GafferScene::ScenePlug`

The processed output scene.

ArnoldLight

Creates a scene with a single light in it.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs an empty scene.

name

`Gaffer::StringPlug`

The name of the object in the output scene.

out

`GafferScene::ScenePlug`

The output scene.

parameters

`Gaffer::CompoundPlug`

The parameters of the light shader - these will vary based on the light type.

transform

`Gaffer::TransformPlug`

The transform applied to the object.

translate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

rotate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

scale

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

ArnoldOptions

The base type for nodes that apply options to the scene.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

in

`GafferScene::ScenePlug`

The input scene.

options

`Gaffer::CompoundDataPlug`

The options to be applied - arbitrary numbers of user defined options may be added as children of this plug via the user interface, or using the CompoundDataPlug API via python.

aaSamples.value

`Gaffer::IntPlug`

!!!*EMPTY*!!!

giDiffuseSamples.value

`Gaffer::IntPlug`

!!!*EMPTY*!!!

giGlossySamples.value

`Gaffer::IntPlug`

!!!*EMPTY*!!!

giRefractionSamples.value

`Gaffer::IntPlug`

!!!*EMPTY*!!!

ignoreTextures.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

ignoreShaders.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

ignoreAtmosphere.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

ignoreLights.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

ignoreShadows.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

ignoreSubdivision.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

ignoreDisplacement.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

ignoreBump.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

ignoreMotionBlur.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

ignoreSSS.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

textureSearchPath.value

`Gaffer::StringPlug`

!!!*EMPTY*!!!

proceduralSearchPath.value

`Gaffer::StringPlug`

!!!*EMPTY*!!!

shaderSearchPath.value

`Gaffer::StringPlug`

!!!*EMPTY*!!!

errorColorBadTexture.value

`Gaffer::Color3fPlug`

!!!*EMPTY*!!!

errorColorBadMesh.value

`Gaffer::Color3fPlug`

!!!*EMPTY*!!!

errorColorBadPixel.value

`Gaffer::Color3fPlug`

!!!*EMPTY*!!!

errorColorBadShader.value

`Gaffer::Color3fPlug`

!!!*EMPTY*!!!

out

`GafferScene::ScenePlug`

The processed output scene.

ArnoldRender

A base class for nodes which can render scenes.

Plugs:

despatcherParameters

`Gaffer::CompoundPlug`

!!!*EMPTY*!!!

fileName

`Gaffer::StringPlug`

!!!*EMPTY*!!!

in

`GafferScene::ScenePlug`

The scene to be rendered.

mode

`Gaffer::StringPlug`

!!!*EMPTY*!!!

requirement

`Gaffer::Plug`

!!!*EMPTY*!!!

requirements

`Gaffer::ArrayPlug`

!!!*EMPTY*!!!

requirement0

`Gaffer::Plug`

!!!*EMPTY*!!!

verbosity

`Gaffer::IntPlug`

!!!*EMPTY*!!!

ArnoldShader

The base type for all nodes which create shaders. Use the ShaderAssignment node to assign them to objects in the scene.

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

name

`Gaffer::StringPlug`

The name of the shader being represented. This should be considered read-only. Use the Shader.loadShader() method to load a shader.

parameters

`Gaffer::CompoundPlug`

Where the parameters for the shader are represented.

type

`Gaffer::StringPlug`

!!!*EMPTY*!!!

GafferImage

Clamp

!!!*EMPTY*!!!

Plugs:

channels

`GafferImage::ChannelMaskPlug`

!!!*EMPTY*!!!

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

in

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

max

`Gaffer::Color4fPlug`

!!!*EMPTY*!!!

maxClampTo

`Gaffer::Color4fPlug`

!!!*EMPTY*!!!

maxClampToEnabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

maxEnabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

min

`Gaffer::Color4fPlug`

!!!*EMPTY*!!!

minClampTo

`Gaffer::Color4fPlug`

!!!*EMPTY*!!!

minClampToEnabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

minEnabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

out

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

Constant

!!!*EMPTY*!!!

Plugs:

color

`Gaffer::Color4fPlug`

!!!*EMPTY*!!!

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

format

`GafferImage::FormatPlug`

!!!*EMPTY*!!!

out

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

Display

!!!*EMPTY*!!!

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

out

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

port

`Gaffer::IntPlug`

!!!*EMPTY*!!!

Grade

!!!*EMPTY*!!!

Plugs:

blackClamp

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

blackPoint

`Gaffer::Color3fPlug`

!!!*EMPTY*!!!

channels

`GafferImage::ChannelMaskPlug`

!!!*EMPTY*!!!

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

gain

`Gaffer::Color3fPlug`

!!!*EMPTY*!!!

gamma

`Gaffer::Color3fPlug`

!!!*EMPTY*!!!

in

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

lift

`Gaffer::Color3fPlug`

!!!*EMPTY*!!!

multiply

`Gaffer::Color3fPlug`

!!!*EMPTY*!!!

offset

`Gaffer::Color3fPlug`

!!!*EMPTY*!!!

out

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

whiteClamp

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

whitePoint

`Gaffer::Color3fPlug`

!!!*EMPTY*!!!

ImageContextVariables

!!!*EMPTY*!!!

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

in

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

out

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

variables

`Gaffer::CompoundDataPlug`

!!!*EMPTY*!!!

ImageReader

!!!*EMPTY*!!!

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

fileName

`Gaffer::StringPlug`

!!!*EMPTY*!!!

out

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

ImageStats

!!!*EMPTY*!!!

Plugs:

average

`Gaffer::Color4fPlug`

!!!*EMPTY*!!!

channels

`GafferImage::ChannelMaskPlug`

!!!*EMPTY*!!!

in

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

max

`Gaffer::Color4fPlug`

!!!*EMPTY*!!!

min

`Gaffer::Color4fPlug`

!!!*EMPTY*!!!

regionOfInterest

`Gaffer::Box2iPlug`

!!!*EMPTY*!!!

ImageSwitch

!!!*EMPTY*!!!

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

in

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

index

`Gaffer::IntPlug`

!!!*EMPTY*!!!

out

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

ImageTimeWarp

!!!*EMPTY*!!!

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

in

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

offset

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

out

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

speed

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

ImageTransform

!!!*EMPTY*!!!

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

filter

`GafferImage::FilterPlug`

!!!*EMPTY*!!!

in

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

out

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

transform

`Gaffer::Transform2DPlug`

!!!*EMPTY*!!!

translate

`Gaffer::V2fPlug`

!!!*EMPTY*!!!

x

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

y

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

rotate

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

scale

`Gaffer::V2fPlug`

!!!*EMPTY*!!!

x

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

y

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

pivot

`Gaffer::V2fPlug`

!!!*EMPTY*!!!

x

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

y

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

ImageWriter

!!!*EMPTY*!!!

Plugs:

channels

`GafferImage::ChannelMaskPlug`

!!!*EMPTY*!!!

despatcherParameters

`Gaffer::CompoundPlug`

!!!*EMPTY*!!!

fileName

`Gaffer::StringPlug`

!!!*EMPTY*!!!

in

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

requirement

`Gaffer::Plug`

!!!*EMPTY*!!!

requirements

`Gaffer::ArrayPlug`

!!!*EMPTY*!!!

writeMode

`Gaffer::IntPlug`

!!!*EMPTY*!!!

Merge

!!!*EMPTY*!!!

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

in

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

in1

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

operation

`Gaffer::IntPlug`

!!!*EMPTY*!!!

out

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

ObjectToImage

!!!*EMPTY*!!!

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

object

`Gaffer::ObjectPlug`

!!!*EMPTY*!!!

out

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

OpenColorIO

!!!*EMPTY*!!!

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

in

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

inputSpace

`Gaffer::StringPlug`

!!!*EMPTY*!!!

out

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

outputSpace

`Gaffer::StringPlug`

!!!*EMPTY*!!!

Reformat

!!!*EMPTY*!!!

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

filter

`GafferImage::FilterPlug`

!!!*EMPTY*!!!

format

`GafferImage::FormatPlug`

!!!*EMPTY*!!!

in

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

out

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

RemoveChannels

!!!*EMPTY*!!!

Plugs:

channels

`GafferImage::ChannelMaskPlug`

!!!*EMPTY*!!!

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

in

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

mode

`Gaffer::IntPlug`

!!!*EMPTY*!!!

out

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

Select

!!!*EMPTY*!!!

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

in

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

in1

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

out

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

select

`Gaffer::IntPlug`

!!!*EMPTY*!!!

GafferOSL

OSLImage

Executes OSL shaders to perform image processing.

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

in

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

out

`GafferImage::ImagePlug`

!!!*EMPTY*!!!

shader

`Gaffer::Plug`

The shader to be executed - connect the output from an OSL network here.

OSLObject

Executes OSL shaders to perform object processing.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

filter

`Gaffer::IntPlug`

The filter used to control which parts of the scene are processed. A Filter node should be connected here.

in

`GafferScene::ScenePlug`

The input scene.

out

`GafferScene::ScenePlug`

The processed output scene.

shader

`Gaffer::Plug`

The shader to be executed - connect the output from an OSL network here.

OSLShader

Represents OSL shaders.

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

name

`Gaffer::StringPlug`

The name of the shader being represented. This should be considered read-only. Use the Shader.loadShader() method to load a shader.

parameters

`Gaffer::CompoundPlug`

Where the parameters for the shader are represented.

type

`Gaffer::StringPlug`

!!!*EMPTY*!!!

GafferRenderMan

InteractiveRenderManRender

A base class for nodes which can render scenes interactively, updating the render to reflect changes to the node graph.

Plugs:

in

`GafferScene::ScenePlug`

The scene to be rendered.

state

`Gaffer::IntPlug`

The interactive state.

updateLights

`Gaffer::BoolPlug`

When on, changes to lights are reflected in the interactive render.

updateShaders

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

RenderManAttributes

The base type for nodes that apply attributes to the scene.

Plugs:

attributes

`Gaffer::CompoundDataPlug`

The attributes to be applied - arbitrary numbers of user defined attributes may be added as children of this plug via the user interface, or using the CompoundDataPlug API via python.

cameraVisibility.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

cameraHitMode.value

`Gaffer::StringPlug`

!!!*EMPTY*!!!

transmissionVisibility.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

transmissionHitMode.value

`Gaffer::StringPlug`

!!!*EMPTY*!!!

diffuseVisibility.value

Gaffer::BoolPlug

!!!*EMPTY*!!!

diffuseHitMode.value

Gaffer::StringPlug

!!!*EMPTY*!!!

specularVisibility.value

Gaffer::BoolPlug

!!!*EMPTY*!!!

specularHitMode.value

Gaffer::StringPlug

!!!*EMPTY*!!!

photonVisibility.value

Gaffer::BoolPlug

!!!*EMPTY*!!!

photonHitMode.value

Gaffer::StringPlug

!!!*EMPTY*!!!

shadingRate.value

Gaffer::FloatPlug

!!!*EMPTY*!!!

relativeShadingRate.value

Gaffer::FloatPlug

!!!*EMPTY*!!!

matte.value

Gaffer::BoolPlug

!!!*EMPTY*!!!

displacementBound.value

Gaffer::FloatPlug

!!!*EMPTY*!!!

maxDiffuseDepth.value

Gaffer::IntPlug

!!!*EMPTY*!!!

maxSpecularDepth.value

Gaffer::IntPlug

!!!*EMPTY*!!!

traceDisplacements.value

Gaffer::BoolPlug

!!!*EMPTY*!!!

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

filter

`Gaffer::IntPlug`

The filter used to control which parts of the scene are processed. A Filter node should be connected here.

in

`GafferScene::ScenePlug`

The input scene.

out

`GafferScene::ScenePlug`

The processed output scene.

RenderManLight

Creates a scene with a single light in it.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs an empty scene.

name

`Gaffer::StringPlug`

The name of the object in the output scene.

out

`GafferScene::ScenePlug`

The output scene.

parameters

`Gaffer::CompoundPlug`

The parameters of the light shader - these will vary based on the light type.

transform

`Gaffer::TransformPlug`

The transform applied to the object.

translate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

rotate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

scale

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

RenderManOptions

The base type for nodes that apply options to the scene.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

in

`GafferScene::ScenePlug`

The input scene.

options

`Gaffer::CompoundDataPlug`

The options to be applied - arbitrary numbers of user defined options may be added as children of this plug via the user interface, or using the CompoundDataPlug API via python.

pixelSamples.value

`Gaffer::V2iPlug`

!!!*EMPTY*!!!

hider.value

`Gaffer::StringPlug`

!!!*EMPTY*!!!

hiderDepthFilter.value

`Gaffer::StringPlug`

!!!*EMPTY*!!!

hiderJitter.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

hiderSampleMotion.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

hiderExtremeMotionDOF.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

hiderProgressive.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

statisticsLevel.value

`Gaffer::IntPlug`

!!!*EMPTY*!!!

statisticsFileName.value

`Gaffer::StringPlug`

!!!*EMPTY*!!!

statisticsProgress.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

shaderSearchPath.value

`Gaffer::StringPlug`

!!!*EMPTY*!!!

textureSearchPath.value

`Gaffer::StringPlug`

!!!*EMPTY*!!!

displaySearchPath.value

`Gaffer::StringPlug`

!!!*EMPTY*!!!

archiveSearchPath.value

`Gaffer::StringPlug`

!!!*EMPTY*!!!

proceduralSearchPath.value

`Gaffer::StringPlug`

!!!*EMPTY*!!!

out

`GafferScene::ScenePlug`

The processed output scene.

RenderManRender

A base class for nodes which can render scenes.

Plugs:

despatcherParameters

`Gaffer::CompoundPlug`

!!!*EMPTY*!!!

in

`GafferScene::ScenePlug`

The scene to be rendered.

mode

`Gaffer::StringPlug`

!!!*EMPTY*!!!

requirement

`Gaffer::Plug`

!!!*EMPTY*!!!

requirements

`Gaffer::ArrayPlug`

!!!*EMPTY*!!!

requirement0

`Gaffer::Plug`

!!!*EMPTY*!!!

ribFileName

`Gaffer::StringPlug`

!!!*EMPTY*!!!

RenderManShader

Loads shaders for use in RenderMan renderers. Use the ShaderAssignment node to assign shaders to objects in the scene.

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

name

`Gaffer::StringPlug`

The name of the shader being represented. This should be considered read-only. Use the Shader.loadShader() method to load a shader.

out

`Gaffer::Plug`

!!!*EMPTY*!!!

parameters

`Gaffer::CompoundPlug`

Where the parameters for the shader are represented.

type

`Gaffer::StringPlug`

!!!*EMPTY*!!!

GafferScene

AimConstraint

The base type for scene processors which use a Filter node to control which part of the scene is affected.

Plugs:

aim

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

filter

`Gaffer::IntPlug`

The filter used to control which parts of the scene are processed. A Filter node should be connected here.

in

`GafferScene::ScenePlug`

The input scene.

out

`GafferScene::ScenePlug`

The processed output scene.

target

`Gaffer::StringPlug`

!!!*EMPTY*!!!

targetMode

`Gaffer::IntPlug`

!!!*EMPTY*!!!

up

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

AlembicSource

The base type for all nodes which are capable of generating a hierarchical scene.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs an empty scene.

fileName

`Gaffer::StringPlug`

!!!*EMPTY*!!!

out

`GafferScene::ScenePlug`

The output scene.

refreshCount

`Gaffer::IntPlug`

!!!*EMPTY*!!!

AttributeCache

The base type for scene processors which use a Filter node to control which part of the scene is affected.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

fileName

`Gaffer::StringPlug`

!!!*EMPTY*!!!

filter

`Gaffer::IntPlug`

The filter used to control which parts of the scene are processed. A Filter node should be connected here.

in

`GafferScene::ScenePlug`

The input scene.

out

`GafferScene::ScenePlug`

The processed output scene.

Camera

A node which produces scenes with exactly one object in them.

Plugs:

clippingPlanes

`Gaffer::V2fPlug`

!!!*EMPTY*!!!

x

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

y

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs an empty scene.

fieldOfView

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

name

`Gaffer::StringPlug`

The name of the object in the output scene.

out

`GafferScene::ScenePlug`

The output scene.

projection

`Gaffer::StringPlug`

!!!*EMPTY*!!!

transform

`Gaffer::TransformPlug`

The transform applied to the object.

translate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

rotate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

scale

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

Cube

A node which produces scenes containing a cube.

Plugs:

dimensions

`Gaffer::V3fPlug`

Controls size of the cube.

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs an empty scene.

name

`Gaffer::StringPlug`

The name of the object in the output scene.

out

`GafferScene::ScenePlug`

The output scene.

transform

`Gaffer::TransformPlug`

The transform applied to the object.

translate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

rotate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

scale

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

CustomAttributes

Applies arbitrary user-defined attributes to locations in the scene. Note that for most common cases the StandardAttributes, OpenGLAttributes, RenderManAttributes, and ArnoldAttributes nodes should be used in preference - they provide predefined sets of attributes with customised user interfaces. The CustomAttributes node is of most use when needing to set a custom attribute not supported by the specialised nodes.

Plugs:

attributes

`Gaffer::CompoundDataPlug`

The attributes to be applied - arbitrary numbers of user defined attributes may be added as children of this plug via the user interface, or using the CompoundDataPlug API via python.

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

filter

`Gaffer::IntPlug`

The filter used to control which parts of the scene are processed. A Filter node should be connected here.

in

`GafferScene::ScenePlug`

The input scene.

out

`GafferScene::ScenePlug`

The processed output scene.

CustomOptions

Applies arbitrary user-defined options to the root of the scene. Note that for most common cases the StandardOptions, OpenGLOptions, RenderManOptions, and ArnoldOptions nodes should be used in preference - they provide predefined sets of options with customised user interfaces. The CustomOptions node is of most use when needing to set a custom option not supported by the specialised nodes.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

in

`GafferScene::ScenePlug`

The input scene.

options

`Gaffer::CompoundDataPlug`

The options to be applied - arbitrary numbers of user defined options may be added as children of this plug via the user interface, or using the CompoundDataPlug API via python.

out

`GafferScene::ScenePlug`

The processed output scene.

DeleteAttributes

The base type for scene processors which use a Filter node to control which part of the scene is affected.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

filter

`Gaffer::IntPlug`

The filter used to control which parts of the scene are processed. A Filter node should be connected here.

in

`GafferScene::ScenePlug`

The input scene.

invertNames

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

names

`Gaffer::StringPlug`

!!!*EMPTY*!!!

out

`GafferScene::ScenePlug`

The processed output scene.

DeletePrimitiveVariables

The base type for scene processors which use a Filter node to control which part of the scene is affected.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

filter

`Gaffer::IntPlug`

The filter used to control which parts of the scene are processed. A Filter node should be connected here.

in

`GafferScene::ScenePlug`

The input scene.

invertNames

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

names

`Gaffer::StringPlug`

!!!*EMPTY*!!!

out

`GafferScene::ScenePlug`

The processed output scene.

Displays

The base type for all nodes which take an input scene and process it in some way.

Plugs:

displays

`Gaffer::CompoundPlug`

!!!*EMPTY*!!!

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

in

`GafferScene::ScenePlug`

The input scene.

out

`GafferScene::ScenePlug`

The processed output scene.

Group

The base type for all nodes which take an input scene and process it in some way.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

in

`GafferScene::ScenePlug`

The input scene.

name

`Gaffer::StringPlug`

!!!*EMPTY*!!!

out

`GafferScene::ScenePlug`

The processed output scene.

transform

`Gaffer::TransformPlug`

!!!*EMPTY*!!!

translate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

rotate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

scale

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

Instancer

The base type for all nodes which take an input scene and process it in some way.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

in

`GafferScene::ScenePlug`

The input scene.

instance

`GafferScene::ScenePlug`

!!!*EMPTY*!!!

name

`Gaffer::StringPlug`

!!!*EMPTY*!!!

out

`GafferScene::ScenePlug`

The processed output scene.

parent

`Gaffer::StringPlug`

!!!*EMPTY*!!!

Isolate

The base type for scene processors which use a Filter node to control which part of the scene is affected.

Plugs:

adjustBounds

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

filter

`Gaffer::IntPlug`

The filter used to control which parts of the scene are processed. A Filter node should be connected here.

in

`GafferScene::ScenePlug`

The input scene.

out

`GafferScene::ScenePlug`

The processed output scene.

MapOffset

Adds an offset to object texture coordinates. Provides a convenient way of looking at specific texture UDIMs.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

filter

`Gaffer::IntPlug`

The filter used to control which parts of the scene are processed. A Filter node should be connected here.

in

`GafferScene::ScenePlug`

The input scene.

offset

`Gaffer::V2fPlug`

An offset added to the texture coordinates. Note that moving the texture coordinates in the positive direction will move the texture in the negative direction.

x

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

y

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

out

`GafferScene::ScenePlug`

The processed output scene.

sName

`Gaffer::StringPlug`

The name of the primitive variable holding the s coordinate.

tName

`Gaffer::StringPlug`

The name of the primitive variable holding the t coordinate.

udim

`Gaffer::IntPlug`

A specific UDIM to offset the texture coordinates to. The UDIM is converted to an offset which is added to the offset above.

MapProjection

The base type for scene processors which use a Filter node to control which part of the scene is affected.

Plugs:

camera

`Gaffer::StringPlug`

!!!*EMPTY*!!!

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

filter

`Gaffer::IntPlug`

The filter used to control which parts of the scene are processed. A Filter node should be connected here.

in

`GafferScene::ScenePlug`

The input scene.

out

`GafferScene::ScenePlug`

The processed output scene.

sName

`Gaffer::StringPlug`

!!!*EMPTY*!!!

tName

`Gaffer::StringPlug`

!!!*EMPTY*!!!

MeshType

The base type for scene processors which use a Filter node to control which part of the scene is affected.

Plugs:

calculatePolygonNormals

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

filter

`Gaffer::IntPlug`

The filter used to control which parts of the scene are processed. A Filter node should be connected here.

in

`GafferScene::ScenePlug`

The input scene.

meshType

`Gaffer::StringPlug`

!!!*EMPTY*!!!

out

`GafferScene::ScenePlug`

The processed output scene.

overwriteExistingNormals

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

ObjectToScene

A node which produces scenes with exactly one object in them.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs an empty scene.

name

`Gaffer::StringPlug`

The name of the object in the output scene.

object

`Gaffer::ObjectPlug`

!!!*EMPTY*!!!

out

`GafferScene::ScenePlug`

The output scene.

transform

`Gaffer::TransformPlug`

The transform applied to the object.

translate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

rotate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

scale

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

OpenGLAttributes

The base type for nodes that apply attributes to the scene.

Plugs:

attributes

`Gaffer::CompoundDataPlug`

The attributes to be applied - arbitrary numbers of user defined attributes may be added as children of this plug via the user interface, or using the CompoundDataPlug API via python.

primitiveSolid.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

primitiveWireframe.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

primitiveWireframeColor.value

`Gaffer::Color4fPlug`

!!!*EMPTY*!!!

primitiveWireframeWidth.value

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

primitiveOutline.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

primitiveOutlineColor.value

`Gaffer::Color4fPlug`

!!!*EMPTY*!!!

primitiveOutlineWidth.value

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

primitivePoint.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

primitivePointColor.value

`Gaffer::Color4fPlug`

!!!*EMPTY*!!!

primitivePointWidth.value

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

primitiveBound.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

primitiveBoundColor.value

`Gaffer::Color4fPlug`

!!!*EMPTY*!!!

pointsPrimitiveUseGLPoints.value

`Gaffer::StringPlug`

!!!*EMPTY*!!!

pointsPrimitiveGLPointWidth.value

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

curvesPrimitiveUseGLLines.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

curvesPrimitiveGLLineWidth.value

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

curvesPrimitiveIgnoreBasis.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

filter

`Gaffer::IntPlug`

The filter used to control which parts of the scene are processed. A Filter node should be connected here.

in

`GafferScene::ScenePlug`

The input scene.

out

`GafferScene::ScenePlug`

The processed output scene.

OpenGLRender

A base class for nodes which can render scenes.

Plugs:

despatcherParameters

`Gaffer::CompoundPlug`

!!!*EMPTY*!!!

in

`GafferScene::ScenePlug`

The scene to be rendered.

requirement

`Gaffer::Plug`

!!!*EMPTY*!!!

requirements

`Gaffer::ArrayPlug`

!!!*EMPTY*!!!

requirement0

`Gaffer::Plug`

!!!*EMPTY*!!!

OpenGLShader

The base type for all nodes which create shaders. Use the ShaderAssignment node to assign them to objects in the scene.

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

name

`Gaffer::StringPlug`

The name of the shader being represented. This should be considered read-only. Use the Shader.loadShader() method to load a shader.

out

`Gaffer::Plug`

!!!*EMPTY*!!!

parameters

`Gaffer::CompoundPlug`

Where the parameters for the shader are represented.

type

`Gaffer::StringPlug`

!!!*EMPTY*!!!

PathFilter

!!!*EMPTY*!!!

Plugs:

match

`Gaffer::IntPlug`

!!!*EMPTY*!!!

paths

`Gaffer::StringVectorDataPlug`

!!!*EMPTY*!!!

Plane

A node which produces scenes containing a plane.

Plugs:

dimensions

`Gaffer::V2fPlug`

Controls size of the plane in X and Y.

x

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

y

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

divisions

`Gaffer::V2iPlug`

Controls tesselation of the plane.

x

`Gaffer::IntPlug`

!!!*EMPTY*!!!

y

`Gaffer::IntPlug`

!!!*EMPTY*!!!

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs an empty scene.

name

`Gaffer::StringPlug`

The name of the object in the output scene.

out

`GafferScene::ScenePlug`

The output scene.

transform

`Gaffer::TransformPlug`

The transform applied to the object.

translate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

rotate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

scale

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

PointConstraint

The base type for scene processors which use a Filter node to control which part of the scene is affected.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

filter

`Gaffer::IntPlug`

The filter used to control which parts of the scene are processed. A Filter node should be connected here.

in

`GafferScene::ScenePlug`

The input scene.

offset

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

out

`GafferScene::ScenePlug`

The processed output scene.

target

`Gaffer::StringPlug`

!!!*EMPTY*!!!

targetMode

`Gaffer::IntPlug`

!!!*EMPTY*!!!

xEnabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

yEnabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

zEnabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

Prune

A node for removing whole branches from the scene hierarchy.

Plugs:

adjustBounds

`Gaffer::BoolPlug`

Computes new tightened bounding boxes taking into account the removed locations. This can be an expensive operation - turn on with care.

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

filter

`Gaffer::IntPlug`

The branches to prune. The specified locations and all locations below them will be removed from the scene.

in

`GafferScene::ScenePlug`

The input scene.

out

`GafferScene::ScenePlug`

The processed output scene.

SceneContextVariables

The base type for all nodes which take an input scene and process it in some way.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

in

`GafferScene::ScenePlug`

The input scene.

out

`GafferScene::ScenePlug`

The processed output scene.

variables

`Gaffer::CompoundDataPlug`

!!!*EMPTY*!!!

SceneReader

The base type for all nodes which are capable of generating a hierarchical scene.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs an empty scene.

fileName

`Gaffer::StringPlug`

!!!*EMPTY*!!!

out

`GafferScene::ScenePlug`

The output scene.

refreshCount

`Gaffer::IntPlug`

!!!*EMPTY*!!!

SceneSwitch

The base type for all nodes which take an input scene and process it in some way.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

in

`GafferScene::ScenePlug`

The input scene.

index

`Gaffer::IntPlug`

!!!*EMPTY*!!!

out

`GafferScene::ScenePlug`

The processed output scene.

SceneTimeWarp

The base type for all nodes which take an input scene and process it in some way.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

in

`GafferScene::ScenePlug`

The input scene.

offset

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

out

`GafferScene::ScenePlug`

The processed output scene.

speed

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

SceneWriter

!!!*EMPTY*!!!

Plugs:

fileName

`Gaffer::StringPlug`

!!!*EMPTY*!!!

in

`GafferScene::ScenePlug`

!!!*EMPTY*!!!

Seeds

The base type for all nodes which take an input scene and process it in some way.

Plugs:

density

`Gaffer::FloatPlug`

!!!*EMPTY*!!!

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

in

`GafferScene::ScenePlug`

The input scene.

name

`Gaffer::StringPlug`

!!!*EMPTY*!!!

out

`GafferScene::ScenePlug`

The processed output scene.

parent

`Gaffer::StringPlug`

!!!*EMPTY*!!!

pointType

`Gaffer::StringPlug`

!!!*EMPTY*!!!

ShaderAssignment

The base type for scene processors which use a Filter node to control which part of the scene is affected.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

filter

`Gaffer::IntPlug`

The filter used to control which parts of the scene are processed. A Filter node should be connected here.

in

`GafferScene::ScenePlug`

The input scene.

out

`GafferScene::ScenePlug`

The processed output scene.

shader

`Gaffer::Plug`

!!!*EMPTY*!!!

ShaderSwitch

!!!*EMPTY*!!!

Plugs:

enabled

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

in

`Gaffer::Plug`

!!!*EMPTY*!!!

index

`Gaffer::IntPlug`

!!!*EMPTY*!!!

out

`Gaffer::Plug`

!!!*EMPTY*!!!

Sphere

A node which produces scenes containing a sphere.

Plugs:

divisions

`Gaffer::V2iPlug`

Controls tesselation of the sphere when type is Mesh.

x

`Gaffer::IntPlug`

!!!*EMPTY*!!!

y

`Gaffer::IntPlug`

!!!*EMPTY*!!!

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs an empty scene.

name

`Gaffer::StringPlug`

The name of the object in the output scene.

out

`GafferScene::ScenePlug`

The output scene.

radius

`Gaffer::FloatPlug`

Radius of the sphere.

thetaMax

`Gaffer::FloatPlug`

Limits the extent of the sphere around the pole axis. Valid values are in the range [0,360].

transform

`Gaffer::TransformPlug`

The transform applied to the object.

translate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

rotate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

scale

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

type

`Gaffer::IntPlug`

The type of object to produce. May be a SpherePrimitive or a Mesh.

zMax

`Gaffer::FloatPlug`

Limits the extent of the sphere along the upper pole. Valid values are in the range [-1,1] and should always be greater than zMin.

zMin

`Gaffer::FloatPlug`

Limits the extent of the sphere along the lower pole. Valid values are in the range [-1,1] and should always be less than zMax.

StandardAttributes

The base type for nodes that apply attributes to the scene.

Plugs:

attributes

`Gaffer::CompoundDataPlug`

The attributes to be applied - arbitrary numbers of user defined attributes may be added as children of this plug via the user interface, or using the CompoundDataPlug API via python.

visibility.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

doubleSided.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

transformBlur.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

transformBlurSegments.value

`Gaffer::IntPlug`

!!!*EMPTY*!!!

deformationBlur.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

deformationBlurSegments.value

`Gaffer::IntPlug`

!!!*EMPTY*!!!

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

filter

`Gaffer::IntPlug`

The filter used to control which parts of the scene are processed. A Filter node should be connected here.

in

`GafferScene::ScenePlug`

The input scene.

out

`GafferScene::ScenePlug`

The processed output scene.

StandardOptions

The base type for nodes that apply options to the scene.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

in

`GafferScene::ScenePlug`

The input scene.

options

`Gaffer::CompoundDataPlug`

The options to be applied - arbitrary numbers of user defined options may be added as children of this plug via the user interface, or using the CompoundDataPlug API via python.

renderCamera.value

`Gaffer::StringPlug`

!!!*EMPTY*!!!

renderResolution.value

`Gaffer::V2iPlug`

!!!*EMPTY*!!!

cameraBlur.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

transformBlur.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

deformationBlur.value

`Gaffer::BoolPlug`

!!!*EMPTY*!!!

shutter.value

`Gaffer::V2fPlug`

!!!*EMPTY*!!!

out

`GafferScene::ScenePlug`

The processed output scene.

SubTree

A node for extracting a specific branch from a scene.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

in

`GafferScene::ScenePlug`

The input scene.

out

`GafferScene::ScenePlug`

The processed output scene.

root

`Gaffer::StringPlug`

The location to become the new root for the output scene. All locations below this will be kept, and all others will be discarded.

Text

A node which produces scenes with exactly one object in them.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs an empty scene.

font

`Gaffer::StringPlug`

!!!*EMPTY*!!!

name

`Gaffer::StringPlug`

The name of the object in the output scene.

out

`GafferScene::ScenePlug`

The output scene.

text

`Gaffer::StringPlug`

!!!*EMPTY*!!!

transform

`Gaffer::TransformPlug`

The transform applied to the object.

translate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

rotate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

scale

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

Transform

Modifies the transforms of all locations matched by the filter.

Plugs:

enabled

`Gaffer::BoolPlug`

The on/off state of the node. When it is off, the node outputs the input scene unchanged.

filter

`Gaffer::IntPlug`

The filter used to control which parts of the scene are processed. A Filter node should be connected here.

in

`GafferScene::ScenePlug`

The input scene.

out

`GafferScene::ScenePlug`

The processed output scene.

transform

`Gaffer::TransformPlug`

The transform to be applied.

translate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

rotate

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

scale

`Gaffer::V3fPlug`

!!!*EMPTY*!!!

UnionFilter

!!!*EMPTY*!!!

Plugs:

in

`Gaffer::ArrayPlug`

!!!*EMPTY*!!!

match

`Gaffer::IntPlug`

!!!*EMPTY*!!!

Version 0.87.0
Last updated 2014-01-04 22:13:14 GMT