The Silo Documentation

Release 0.1.2

Florian Einfalt

Aug 31, 2017

Contents

Contents:

CHAPTER 1

Installation

To install `the_silo`, type:

```
$ pip install the_silo
```

Open Nuke's `init.py` file and add:

```
nuke.pluginAddPath('/path/to/your/local/python/site-packages')

import the_silo
the_silo.init()
```

Open Nuke's `menu.py` file and add:

```
the_silo.build()
```

CHAPTER 2

Getting Started

There are currently two categories of tools:

- Gizmos
- Scripts

Getting started is easy. After you have installed The Silo, all available tools will be added to a menu called `The Silo` and sub-menus called `Gizmos` and `Scripts`. That's it. :)

CHAPTER 3

Gizmos

Chromatic Abberations

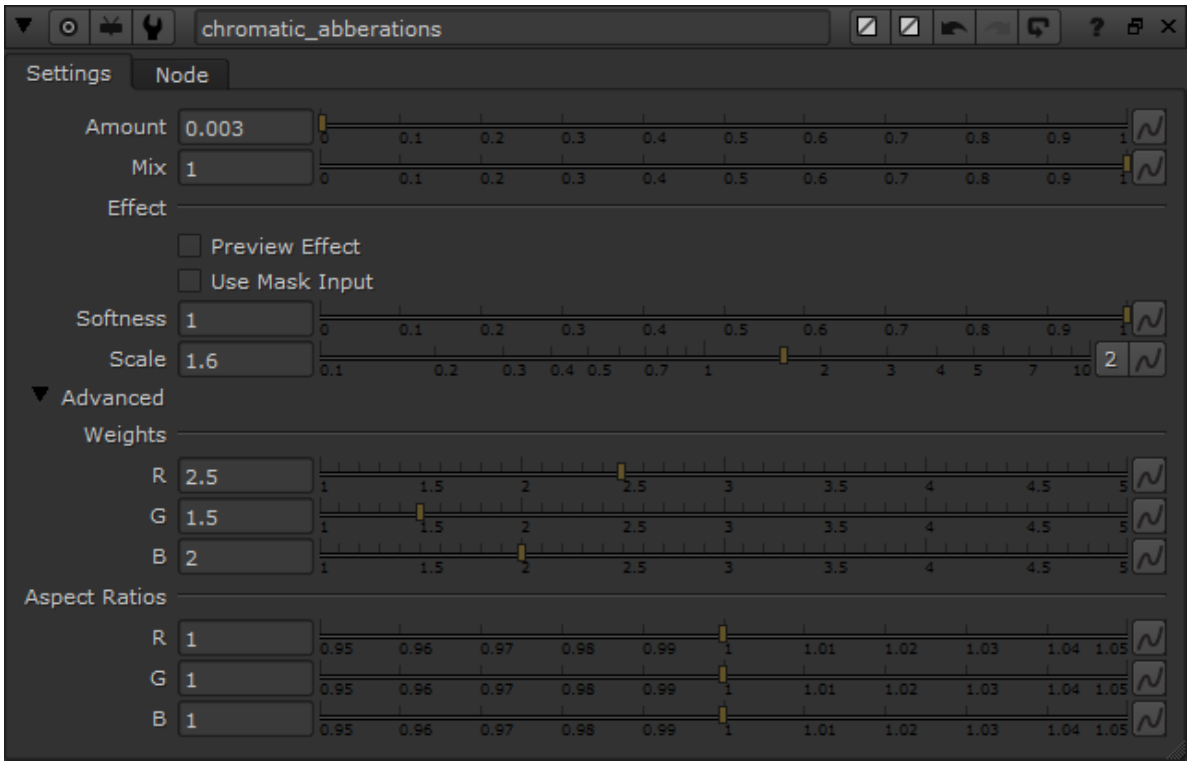
chromatic_abberations()

Simulate chromatic abberation lens effects.

The effect is masked by a built-in radial that can be adjusted in softness and scale in the `Effect` section of the UI. By default the mask will ensure that the effect is more pronounced towards the edges of the image than in the centre. This can be overridden by plugging in a custom mask and switching `Effect > Use Mask Input` on.

The standard weights will produce magenta/green chromatic abberations but can be tweaked in the `Advanced > Weights` section of the UI.

By default, the aspect ratio is `1.0` on all channels, this can be tweaked in the `Advanced > Aspect Ratios` section of the UI to get a horizontally/vertically "squished" effect.



Global:

Parameters

- **Amount** (*float*) – Overall strength of the effect, i.e. separation of RGB channels, default: `0.003`
- **Mix** (*float*) – Mix or opacity value, default: `1.0`

Switches:

Parameters

- **Preview_Effect** (*bool*) – Display the effect only, default: `False`
- **Use_Mask_Input** (*bool*) – Use node's mask input instead of built-in radial mask, default: `False`

Mask:

Parameters

- **Softness** (*float*) – Softness of falloff of built-in radial mask, default: `1.0`
- **Scale** (*float*) – Scale of built-in radial mask, default: `2.0`

Advanced > Weights:

Parameters

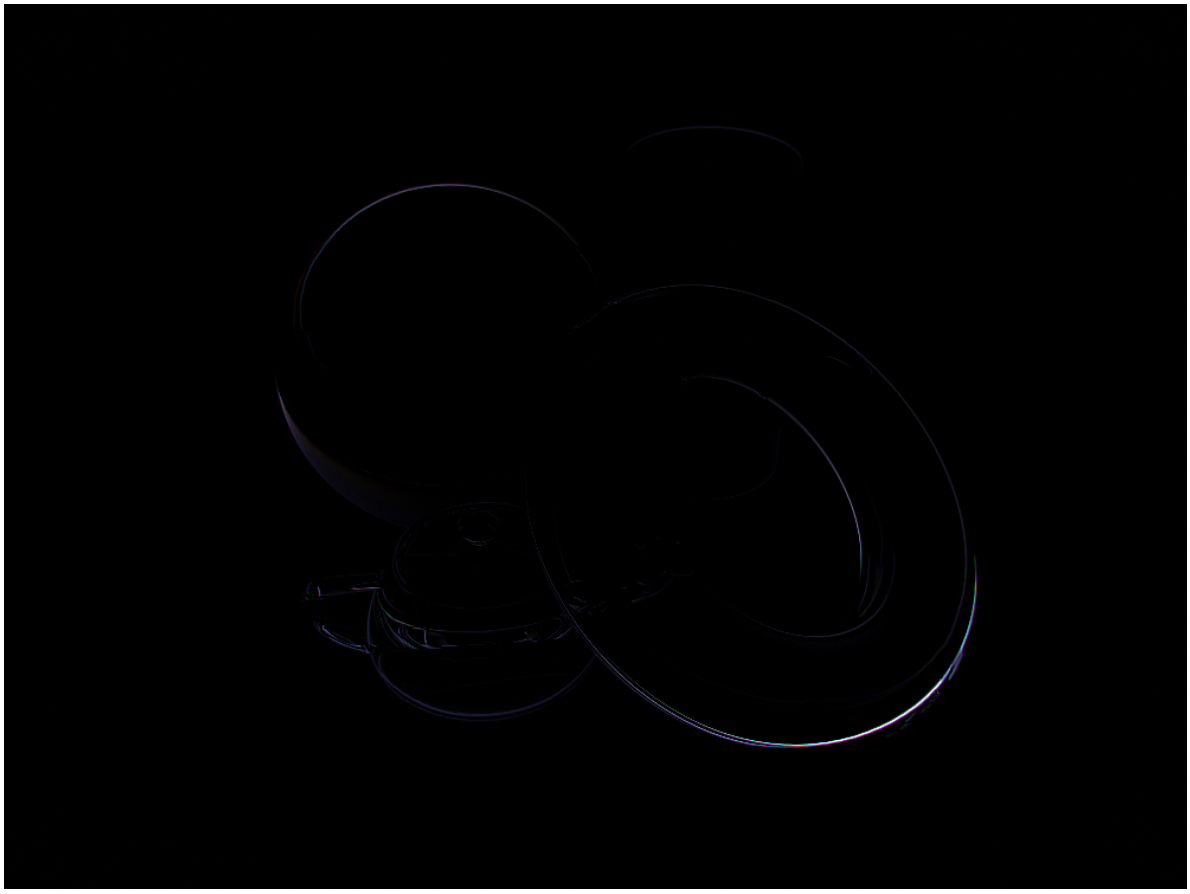
- **Weight_R** (*float*) – Relative strength of the effect on the R channel, default: `2.5`
- **Weight_G** (*float*) – Relative strength of the effect on the G channel, default: `1.5`
- **Weight_B** (*float*) – Relative strength of the effect on the B channel, default: `2`

Advanced > Aspect Ratios:

Parameters

- **Aspect_R** (*float*) – Height/width ratio of the effect on the R channel, default: `1.0`
- **Aspect_G** (*float*) – Height/width ratio of the effect on the G channel, default: `1.0`
- **Aspect_B** (*float*) – Height/width ratio of the effect on the B channel, default: `1.0`

Effect example:

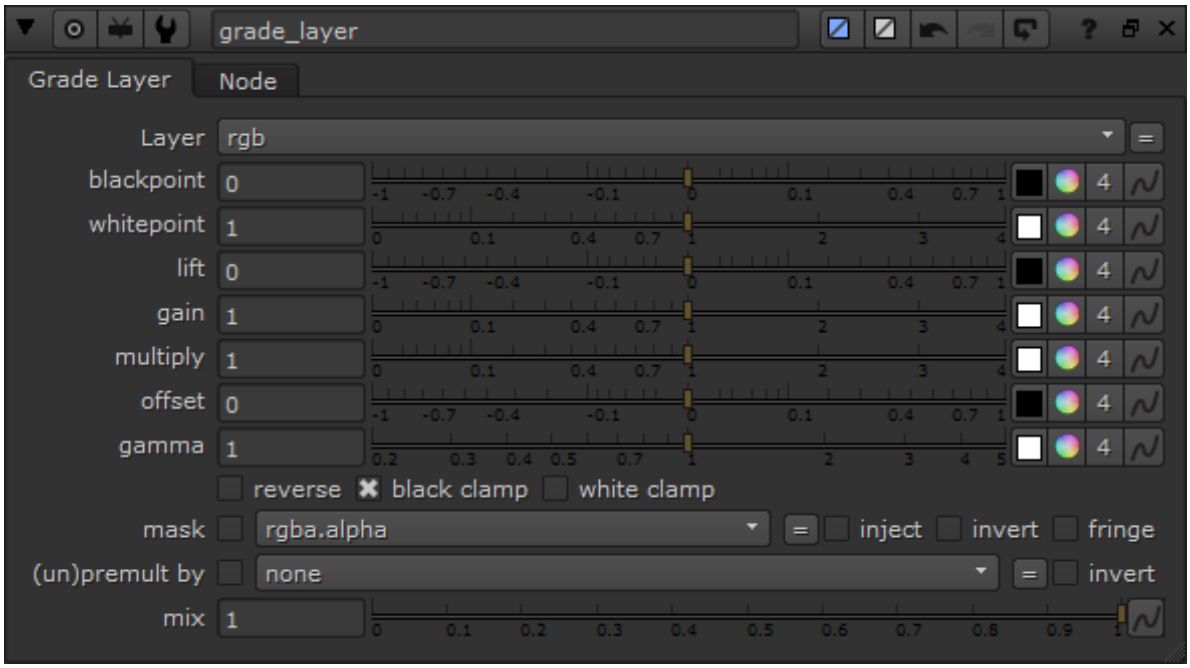


Grade Layer

grade_layer()

In multi-pass compositing, passes or layers are usually shuffled out, tweaked and re-combined to build the beauty render. For pre-comps or other simple composites however, this might be an overhead and the artist will work directly with the multichannel EXR beauty instead.

This gizmos allows the grading of a specific pass/layer, e.g. Reflection, even if passes have not been shuffled out and the result is being re-composited correctly back into the main beauty. It mirrors the Nuke Grade node, the only addition being the `Layer` drop-down menu from which the artist can select the pass/layer to grade.



Global:

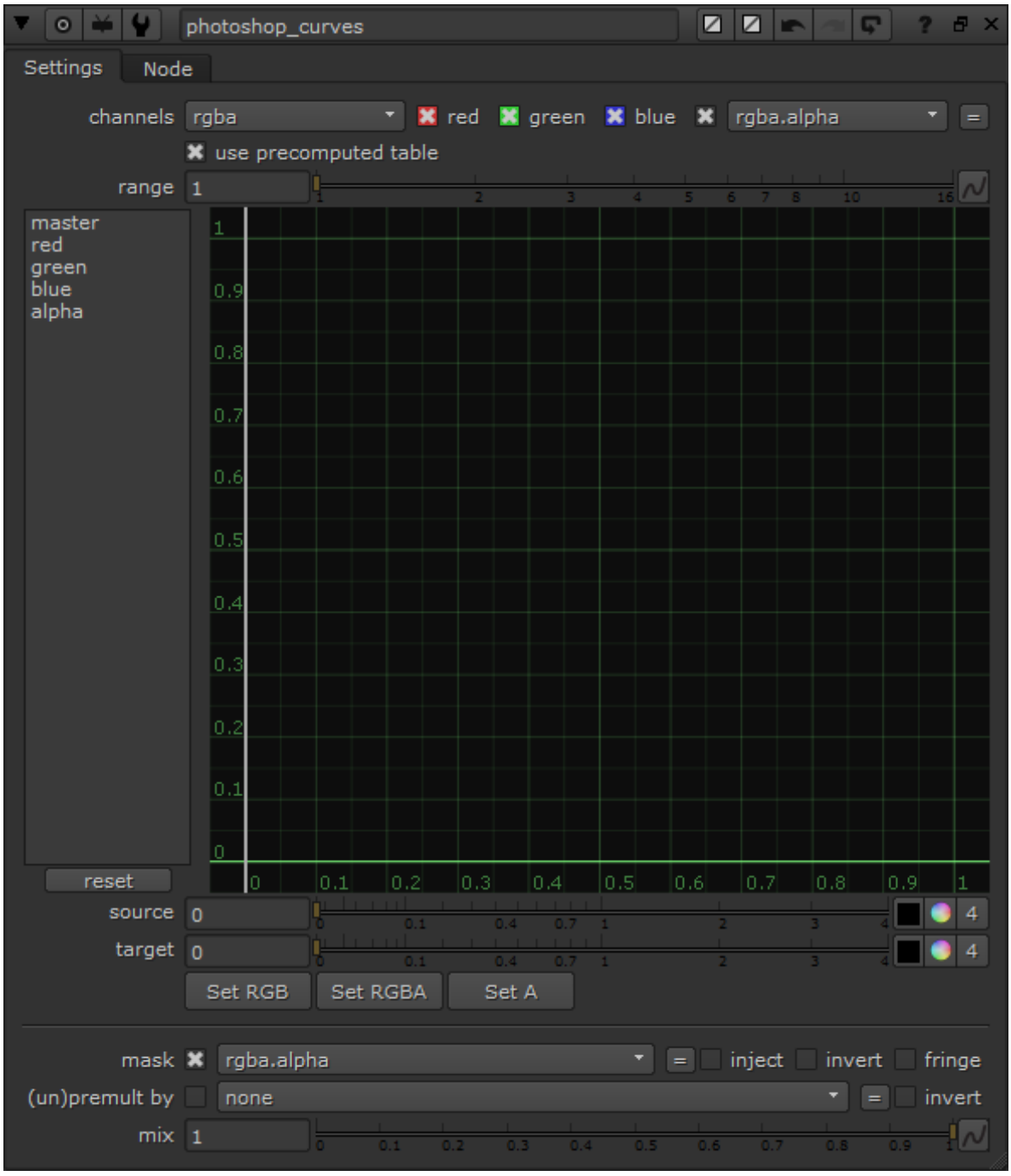
Parameters **Layer** (*str*) – Pass/Layer to be graded and re-composited, default: `rgb`

Photoshop Curves

photoshop_curves()

When co-working with stills retouching teams, mood frames and colour grades might have been done in Photoshop using curves. These curves were therefore applied in `sRGB` and not in `linear` colour space as compared to Nuke's built-in `ColorLookup`.

This gizmo wraps a standard Nuke `ColorLookup` and performs a `Colorspace` conversion to and from `sRGB` before and after so the `ColorLookup` works almost exactly like Photoshop Curves.



Refraction Diffusion

refraction_diffusion()

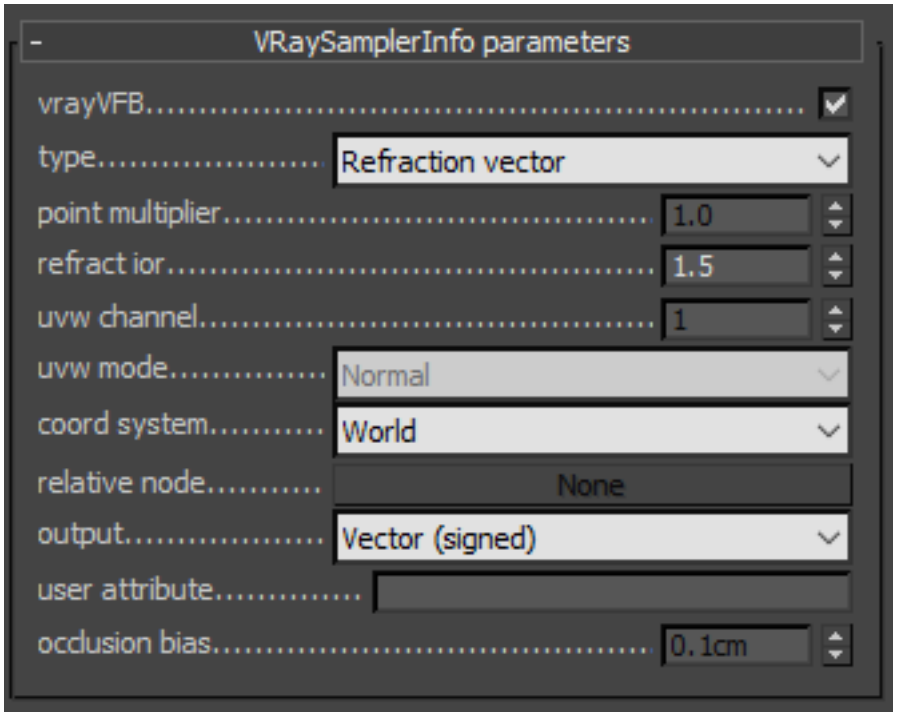
Calculation spectral light effects can be very expensive if calculated in a 3D renderer.

This gimzo simulates a spectral "Refraction Diffusion" effect and requires three separate channels inputs:

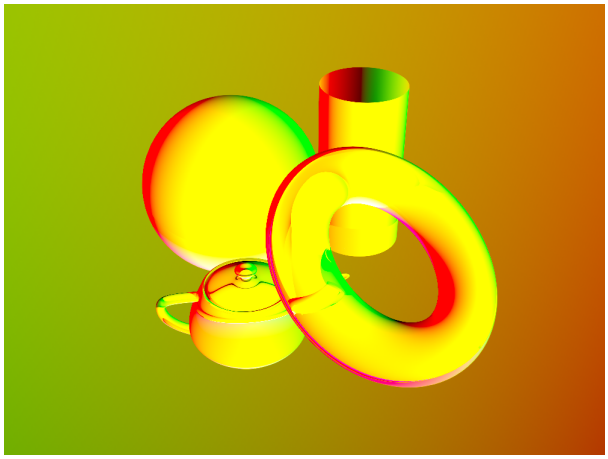
- Refraction RGB
- Refraction/IOR vectors with a low IOR, default: `1.0`

•Refraction/IOR vectors with a higher (glass-like) IOR, default: `1.5`

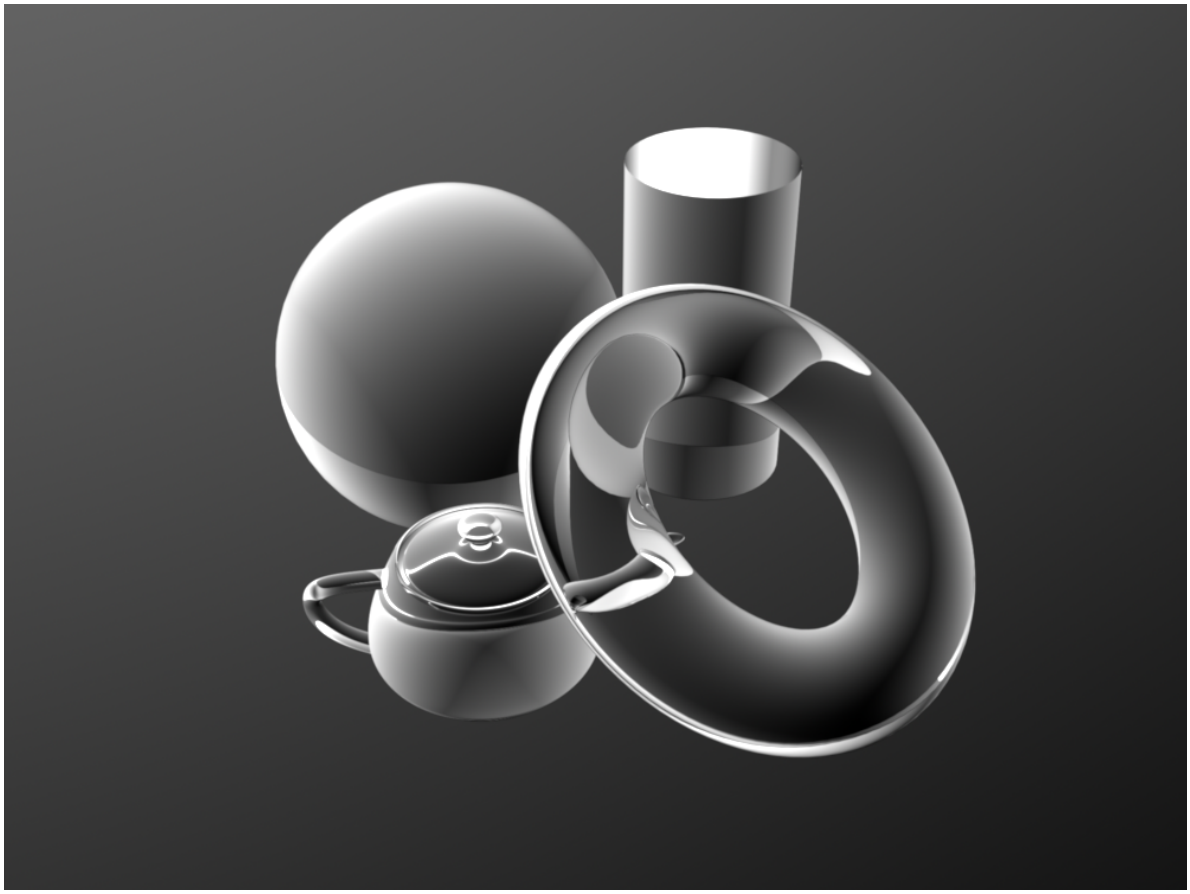
The Refraction/IOR vector passes can be rendered as a V-Ray SamplerInfo Render Element as described below. Adjust the `Refract IOR` value for each of the Refraction/IOR vector passes.



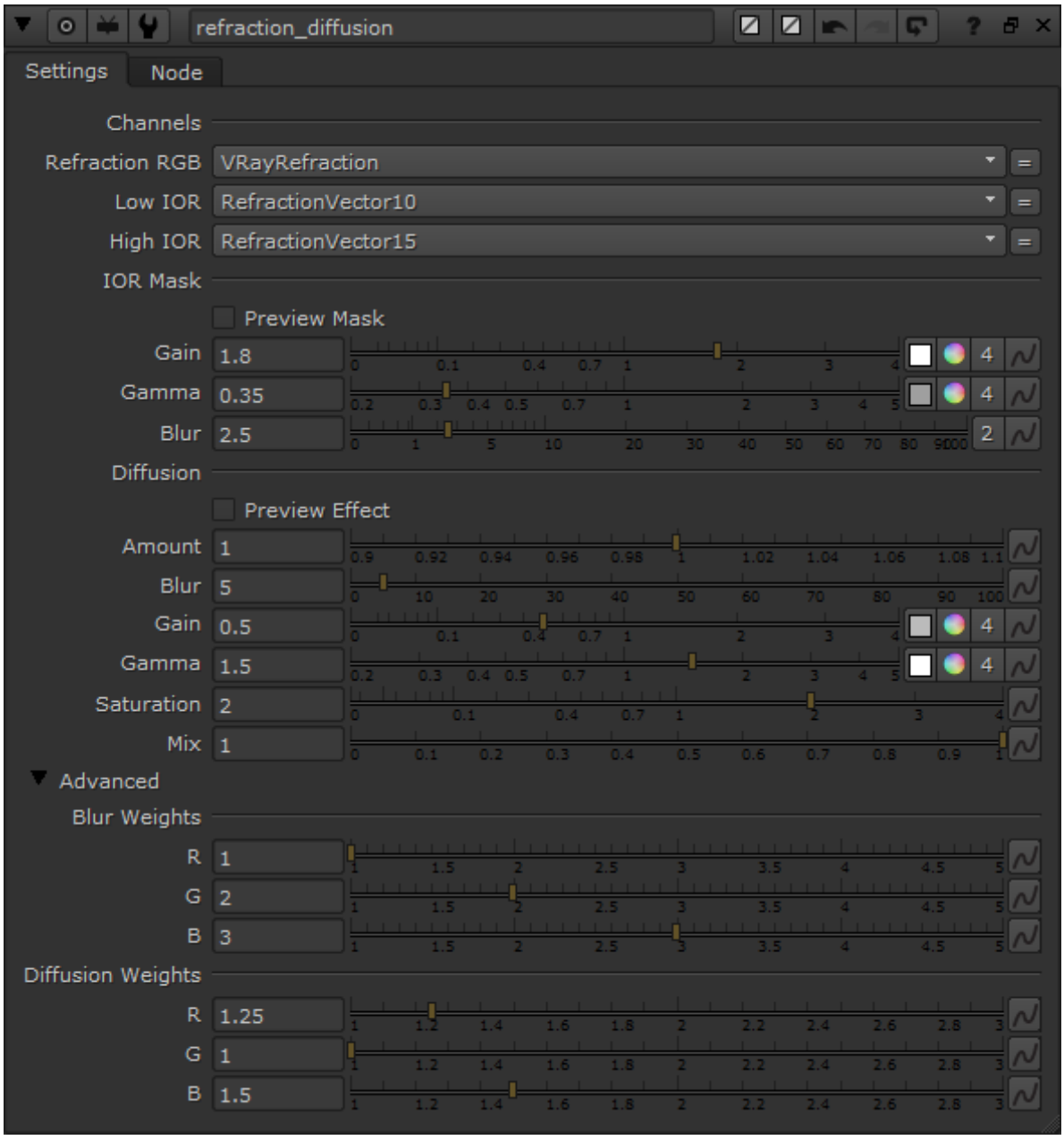
The rendered result should should look roughly like this (IOR `1.0` and `1.5`):



From this the gizmo will calculate a Refraction/IOR mask that is the basis of the effect. The mask can be previewed by toggling the `Preview Mask` switch in the UI:



Adjusting the parameters described below will yield results similar to the effect example image at the the end of this section. The effect can be previewed by toggling the `Preview Effect` switch in the UI.



Channels:

Parameters

- **Refraction_RGB** (*str*) – Refraction RGB pass, default: `VRayRefraction`
- **Low_IOR** (*str*) – Refraction vector pass with lower IOR value, default: `RefractionVector10`
- **High_IOR** (*str*) – Refraction vector pass with higher IOR value, default: `RefractionVector15`

IOR Mask:

Parameters

- **Preview_Mask** (*bool*) – Display the IOR/refraction mask only, default: `False`

- **Gain** (*float*) – IOR/refraction mask multiplier, default: `1.8`
- **Gamma** (*float*) – IOR/refraction mask gamma, default: `0.35`
- **Blur** (*float*) – IOR/refraction mask overall blur in pixels, default: `2.5`

Diffusion:

Parameters

- **Preview_Effect** (*bool*) – Display the effect only, default: `False`
- **Amount** (*float*) – Overall strength of the effect, i.e. separation of RGB channels, default: `1.0`
- **Blur** (*float*) – Effect RGB blur in pixels, default: `5.0`
- **Gain** (*float*) – Effect RGB multiplier, default: `0.5`
- **Gamma** (*float*) – Effect RGB gamma, default: `1.5`
- **Saturation** (*float*) – Effect RGB saturation, default: `2.0`
- **Mix** (*float*) – Effect mix or opacity value, default: `1.0`

Advanced > Blur Weights:

Parameters

- **Blur_R** (*float*) – Relative strength of the effect blur on the R channel, default: `1.0`
- **Blur_G** (*float*) – Relative strength of the effect blur on the G channel, default: `2.0`
- **Blur_B** (*float*) – Relative strength of the effect blur on the B channel, default: `3.0`

Advanced > Diffusion Weights:

Parameters

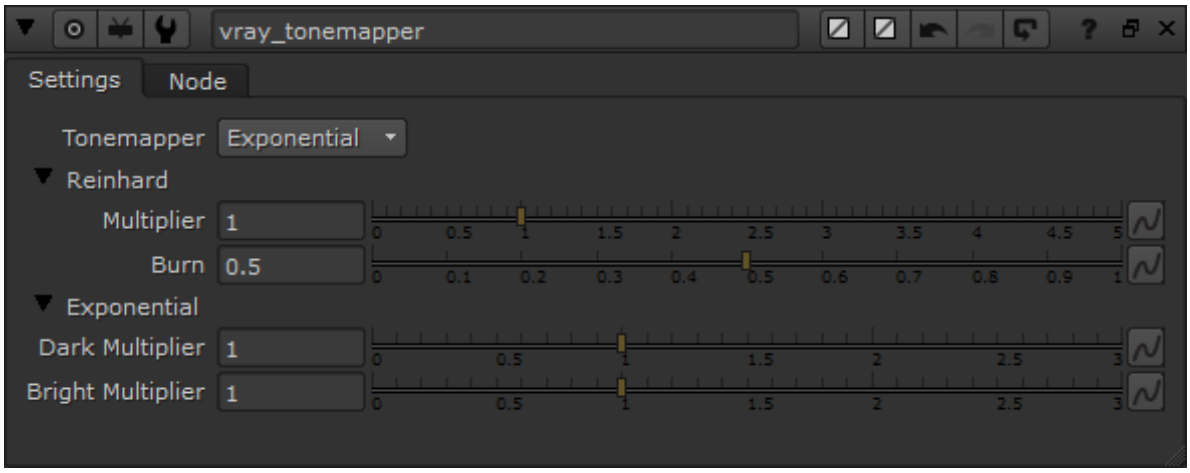
- **Diffusion_R** (*float*) – Relative strength of the effect on the R channel, default: `1.25`
- **Diffusion_G** (*float*) – Relative strength of the effect on the G channel, default: `1.0`
- **Diffusion_B** (*float*) – Relative strength of the effect on the B channel, default: `1.5`

Effect example:

V-Ray Tonemapper

vray_tonemapper()

This gizmo implements the V-Ray `Reinhard` and `Exponential` tone mappers. Documentation.



Global:

Parameters **Tonemapper** (*str*) – Tone mapping algorithm to be used

Reinhard:

Parameters

- **Multiplier** (*float*) – The overall multiplier when the `Tonemapper` is set to `Reinhard`.
- **Burn** (*float*) – Adjusts the blend of mapping between linear and exponential style for the `Reinhard` tone mapper. If Burn Value is `1.0`, the result is linear color mapping; if Burn Value is `0.0`, the result is exponential-style mapping.

Exponential:

Parameters

- **Dark_Multiplier** (*float*) – The multiplier for dark colors.
- **Bright_Multiplier** (*float*) – The multiplier for bright colors.

CHAPTER 4

Scripts

the_silo.scripts.multichannel.**create**()
For a selected group of read nodes from the same rendering process (i.e. same naming pattern, using "." as a separator for Render Elements), create a multichannel EXR.

the_silo.scripts.multichannel.**shuffle**(*width=4096*, *height=4096*)
For either selected read nodes in the compositing script, shuffle out all channels contained and create a write node for each channel.

Parameters

- **width** (*int*) – Contact sheet width
- **height** (*int*) – Contact sheet height

the_silo.scripts.nodes.**move_down**()
Move selected nodes down by 1x the GridHeight

Shortcut: Ctrl+Shift+Down

the_silo.scripts.nodes.**move_left**()
Move selected nodes to the left by 1x the GridWidth

Shortcut: Ctrl+Shift+Left

the_silo.scripts.nodes.**move_right**()
Move selected nodes to the right by 1x the GridWidth

Shortcut: Ctrl+Shift+Right

the_silo.scripts.nodes.**move_up**()
Move selected nodes up by 1x the GridHeight

Shortcut: Ctrl+Shift+Up

the_silo.scripts.reload.**all_reads**()
Reload all read nodes in the compositing script

Shortcut: Ctrl+Shift+R

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index