PxrLightSaturation

A simple display filter that changes the color saturation in the beauty pass (or any other arbitrary output variable) according to whether a region is lit or shadowed by particular light group.

Parameters

aov

Name of a color AOV to adjust saturation in.

light

Name of the color AOV with the contribution from the light group. The AOV should be populated using a direct-lighting LPE such as:

```
Display "+satlighttiny" "null" "color lpe:C[DGS]<L.'tiny'>"
```

in which case the value of this parameter would be `satlighttiny`. This LPE shows direct illumination from any lights with parameter: `"string __group" "tiny"`.

threshold

Luminance in the light AOV less than this is considered shadowed.

invert

If unset then areas lit by the light will be modified and areas fully in shadow will be left alone. Set this to invert that and change the saturation of the shadows of this light.

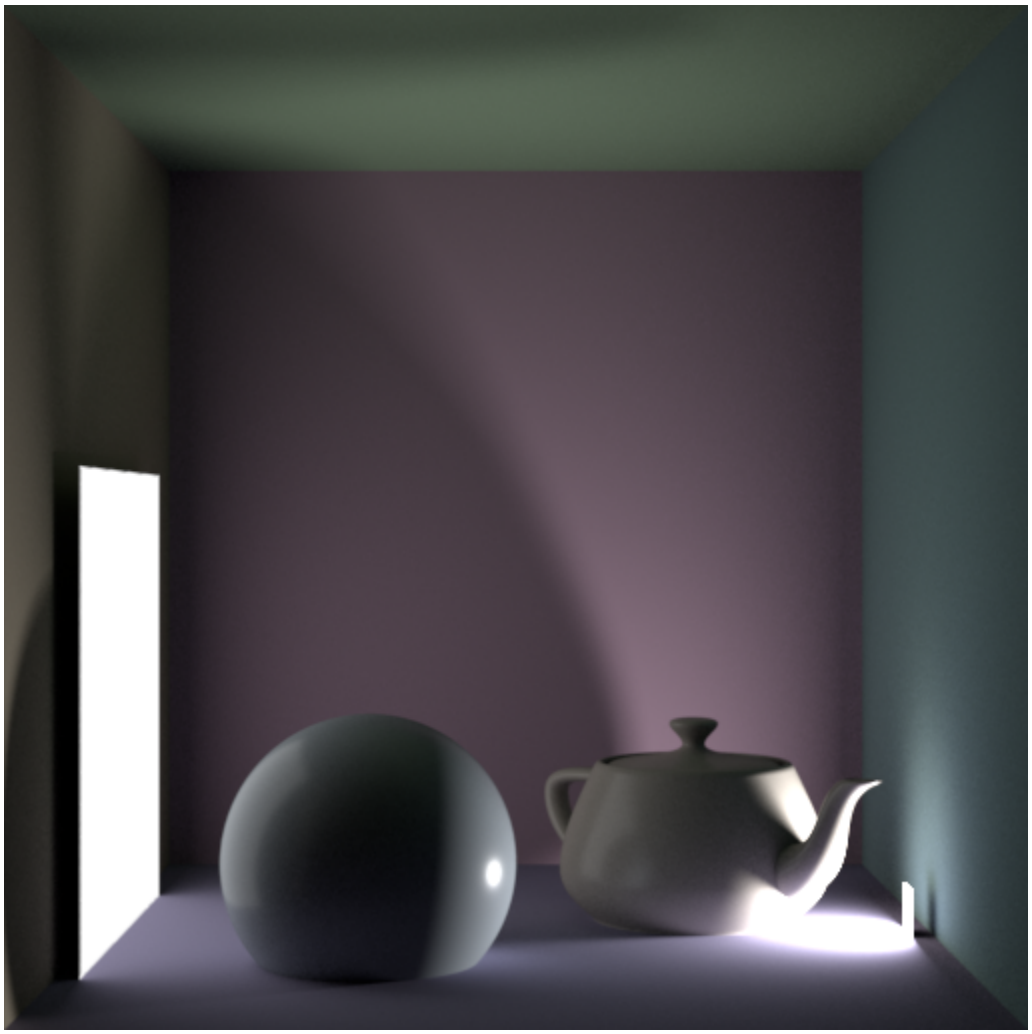
shift

Amount and direction to shift saturation. Positive values increase saturation, negative values desaturate. Recommended range is -2.0 to 2.0.

Example

While RenderMan is primarily focused on physically plausible rendering, the complete toolkit of features provided by PRMan can go beyond this to produce non-physical effects for artistic purposes.

In this example (using RIB), we'll show how to produce a light source that appears to alter the color saturation of any objects that are in shadow from it. Here's the baseline image with ordinary physically lighting that we'll start with:



To derive the saturating light effect, we'll use a combination of light groups, light path expressions (LPEs), and the PxrLightSaturation display filter.

Start by tagging the tiny light on the right with a light group so that we isolate the contributions from it:

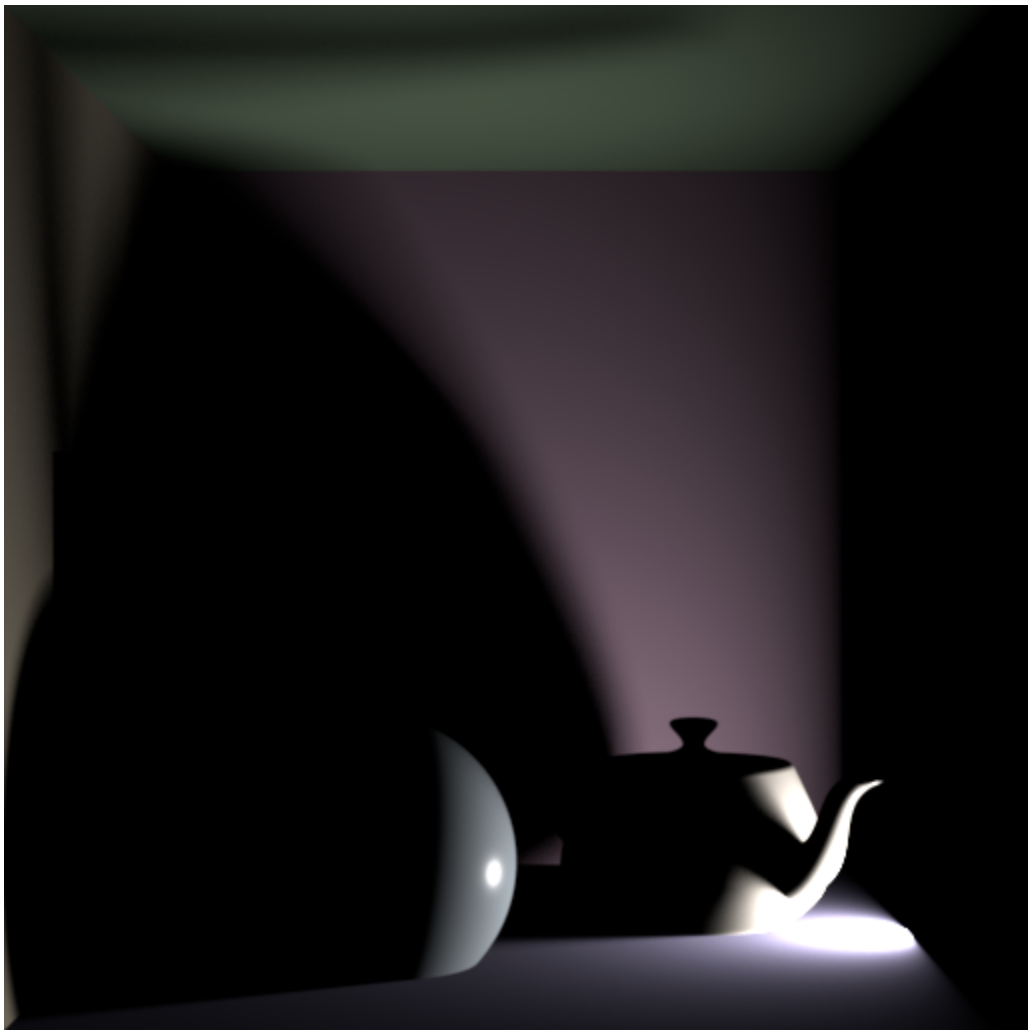
```
Light "PxrMeshLight" "tiny" "float intensity" 400 "string lightGroup" "tiny"
```

Any number of lights can be added to a given light group by tagging them all with the same name. However, for this example we'll just tag the one.

The next step is to add an AOV output with a light path expression that shows all of the surfaces directly lit by this light:

```
DisplayChannel "color tinydirect" "string source" "color lpe:C[DS]<L.'tiny'>"
Display "+tinydirect" "it" "tinydirect"
```

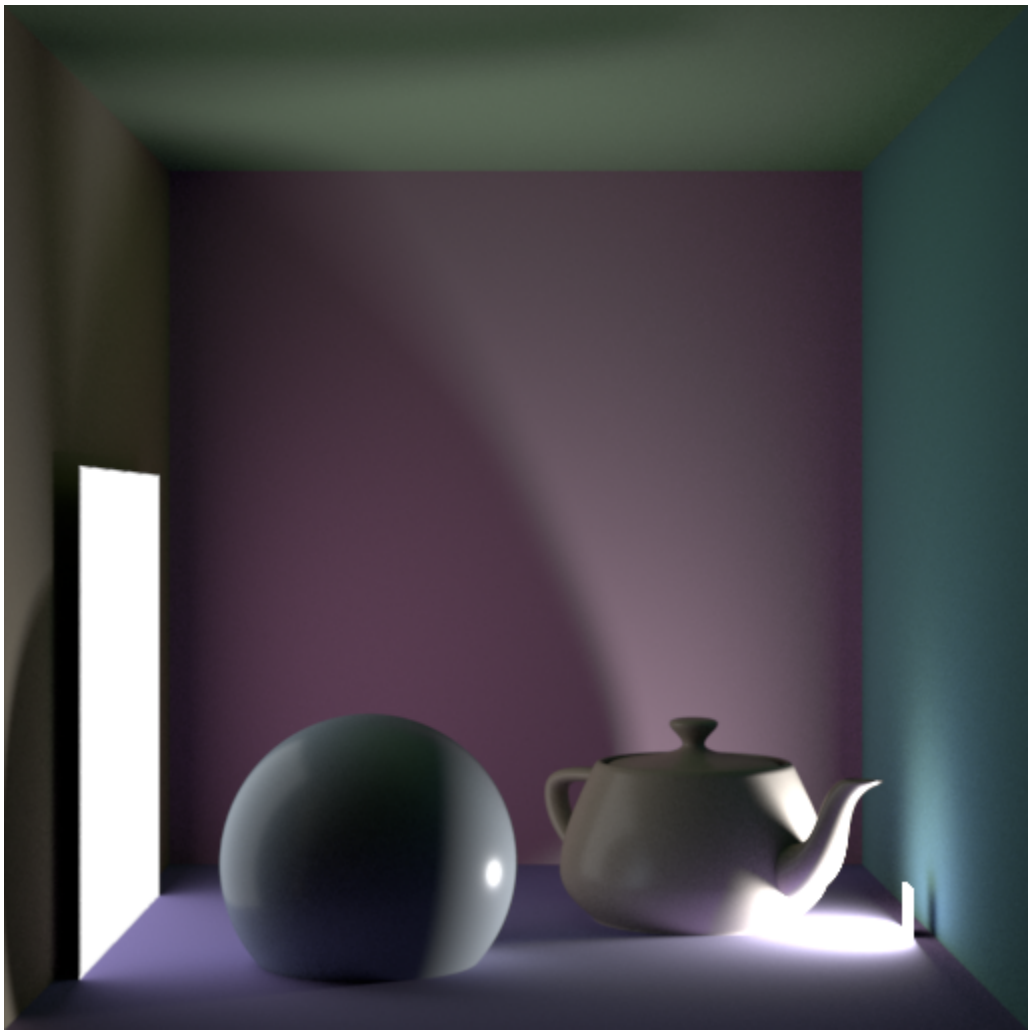
Here, the `'tiny'` in single quotes must match the name given to the light group. The output of that AOV looks like this:



Finally, we add a display filter to tie this all together. Essentially, they can perform post-process operations on the pixels in the beauty image and any AOVs just before they are displayed or sent to file. The particular one used here, PxrLightSaturation, reads from a light group AOV and modifies the beauty image or other AOV accordingly. It takes five parameters: the name of the AOV to modify, the name of the AOV with direct light from a light group, a luminance threshold above which a pixel in the AOV is considered lit, whether to invert its behavior and change shadows instead of lit areas, and an amount to shift the color saturation for lit or shadowed pixels. Here's what the line for it in the scene file looks like:

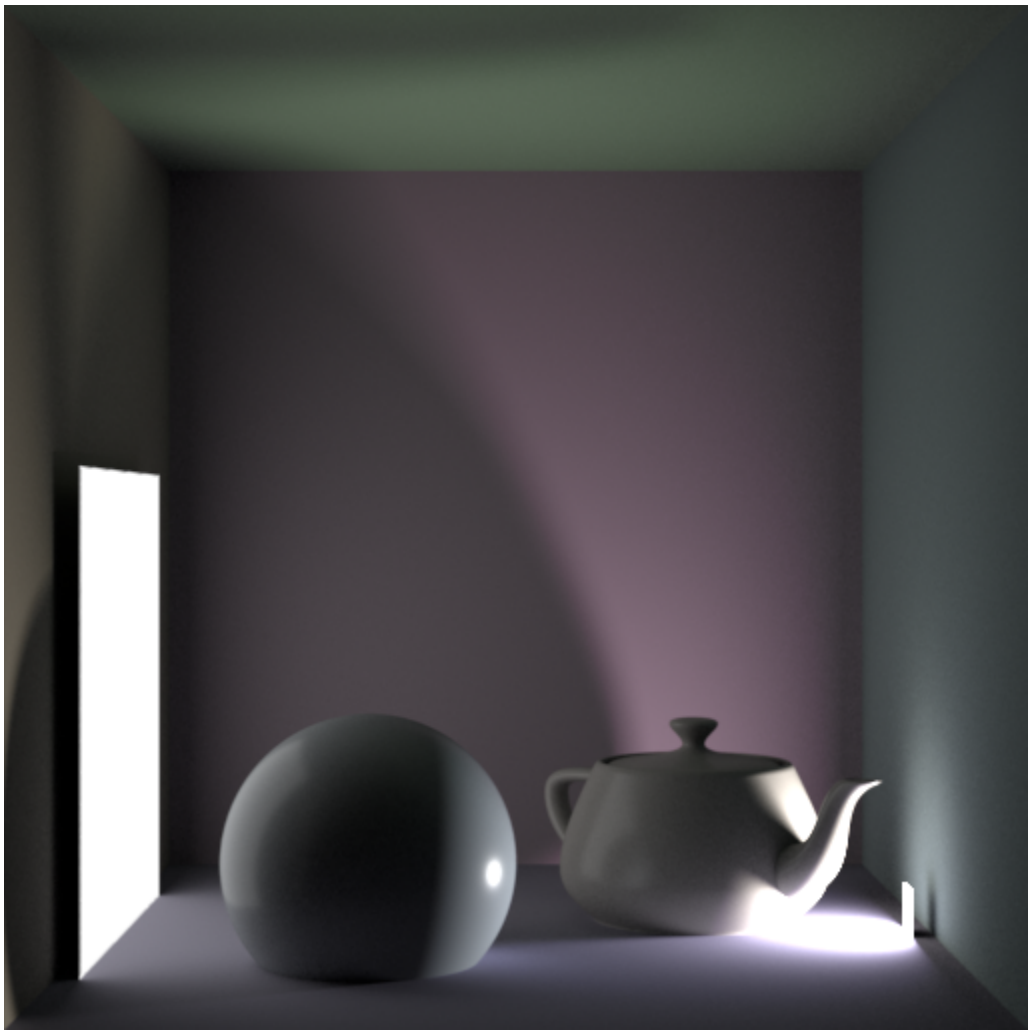
```
DisplayFilter "PxrLightSaturation" "sat" "string light" "tinydirect" "float threshold" 0.03 "int invert" 1
"float shift" 1
```

And here is the complete result:



Compared to the image at the top, the color of right wall as well as the back wall and floor behind the sphere and teapot are now stronger. Likewise, there's stronger color now in The reflection of the floor and ceiling on the sphere and the umbra of the spout's shadow on the ceiling is distinctly greener.

We could use larger shift amounts to exaggerate the color in the shadow even more. Or we can go the other way. With a negative shift amount such as -1.0, we can partly desaturate the areas in shadow from the small light:



Note that all of this works correctly with incremental rendering to a live frame buffer or with interactive rendering. There is no need to wait for the final image to see the results of the imager shader as it will be applied on the fly.

Code for the display filter can be found in the examples distribution as `PxrLightSaturation.cpp` together with the full scene, `satlight.rib`.