Skip to content

Shader Nodes

Cycles applies a number of shader node optimizations both at compile time and run-time. By exploiting them it is possible to design complicated "Uber Shader" style node groups that incur minimal render time overhead for unused features.

Node Optimizations

As the first step in preparing a node shader for execution, Cycles expands all node groups, as if using the Ungroup tool, and discards UI only features lik frames and reroute nodes.

After that, it applies some obvious transformations, for example, it can (the list is not exhaustive):

- Replace the following nodes with the constant result of their evaluation, if all their inputs are determined to be constant:

  RGB, Value, Mix RGB, Math, Vector Math, RGB to BW, Gamma, Bright Contrast, Invert, Separate/Combine RGB/XYZ/HSV, Blackbody, RGB Curves, Vector Curves, Color Ramps.
- Detect Mix RGB, Math and Vector Math nodes that become no-op (without Clamp) or evaluate to 0 as a result of addition, subtraction, multiplicatic division or dot/cross product with a known constant 0 or 1 input, and replace with the appropriate input link or constant result.
- Eliminate Mix RGB Mix (without Clamp) and Mix Shader nodes when Factor is known to be 0 or 1 by replacing with the appropriate input value or link.
- Eliminate no-op Mix RGB (except Color Burn, Color Dodge, Lighten, or enabled Clamp), Invert, RGB Curves and Vector Curves nodes with know zero Factor.
- Eliminate Emission and Background shader nodes that do not emit any light, and Add Shader nodes with one or both input arguments missing.
- Eliminate Bump with constant Height input, using its Normal input or Geometry Normal instead. This is useful for implementing node group inputs that default to normal via routing through a no-op Bump before doing math.
- Replace Attribute nodes of the View Layer type with the evaluated attribute value (it is constant within the whole Render Layer).
- Combine multiple copies of the same node with the same inputs into only one instance.

Finally, any nodes that end up not connected either directly or indirectly to the Output node are removed.

Run-Time Optimizations

When executing shaders, a special optimization is applied to Mix Shader nodes. If Factor evaluates to 0 or 1, any nodes that are only reachable via the unused branch of the mix are not evaluated.

This can substantially reduce the performance cost of combining multiple materials in one shader with a Color Attribute, texture, or other input used as a switch.

Open Shading Language

If Open Shading Language is chosen as the rendering backend, node shaders are translated to OSL code and then compiled and executed by the OSL runtime. In the process it applies its own extensive set of optimizations, both at compile time and run-time.

Open Shading Language can optimize out Script nodes if their outputs are unused or constant, even if their OSL shaders have side effects like debug tracing and message passing, which may be confusing. For that reason message passing with `setmessage` and `getmessage` should generally nc be used for passing information forward in the graph; explicitly passing information through sockets should be preferred.

Previous
Reducing Noise

Ne
Workben

View Source
View Translation
Report issue on this page