

[Skip to content](#)

ShaderNodeBsdfPrincipled(ShaderNode)

base classes — [bpy_struct](#) , [Node](#) , [NodeInternal](#) , [ShaderNode](#)

class bpy.types.ShaderNodeBsdfPrincipled(ShaderNode)

Physically-based, easy-to-use shader for rendering surface materials, based on the OpenPBR model

distribution

Light scattering distribution on rough surface

- `GGX` `GGX`.
- `MULTI_GGX` Multiscatter GGX – GGX with additional correction to account for multiple scattering, preserve energy and prevent unexpected darkening at high roughness.

TYPE:

enum in ['GGX', 'MULTI_GGX'], default 'GGX'

subsurface_method

Method for rendering subsurface scattering

- `BURLEY` Christensen-Burley – Approximation to physically based volume scattering.
- `RANDOM_WALK` Random Walk – Volumetric approximation to physically based volume scattering, using the scattering radius as specific
- `RANDOM_WALK_SKIN` Random Walk (Skin) – Volumetric approximation to physically based volume scattering, with scattering radius automatically adjusted to match color textures. Designed for skin shading..

TYPE:

enum in ['BURLEY', 'RANDOM_WALK', 'RANDOM_WALK_SKIN'], default 'BURLEY'

classmethod `is_registered_node_type()`

True if a registered node type

RETURNS:

Result

RETURN TYPE:

boolean

classmethod `input_template(index)`

Input socket template

PARAMETERS:

index (*int in [0, inf]*) – Index

RETURNS:

result

RETURN TYPE:

[NodeInternalSocketTemplate](#)

classmethod `output_template(index)`

Output socket template

PARAMETERS:

index (*int in [0, inf]*) – Index

RETURNS:

result

RETURN TYPE:`NodeInternalSocketTemplate`**classmethod** `bl_rna_get_subclass(id, default=None)`**PARAMETERS:****id** (*str*) – The RNA type identifier.**RETURNS:**

The RNA type or default when not found.

RETURN TYPE:`bpy.types.Struct` subclass**classmethod** `bl_rna_get_subclass_py(id, default=None)`**PARAMETERS:****id** (*str*) – The RNA type identifier.**RETURNS:**

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- `bpy_struct.id_data`
- `Node.type`
- `Node.location`
- `Node.location_absolute`
- `Node.width`
- `Node.height`
- `Node.dimensions`
- `Node.name`
- `Node.label`
- `Node.inputs`
- `Node.outputs`
- `Node.internal_links`
- `Node.parent`
- `Node.warning_propagation`
- `Node.use_custom_color`
- `Node.color`
- `Node.color_tag`
- `Node.select`
- `Node.show_options`
- `Node.show_preview`
- `Node.hide`
- `Node.mute`
- `Node.show_texture`
- `Node.bl_idname`
- `Node.bl_label`
- `Node.bl_description`
- `Node.bl_icon`
- `Node.bl_static_type`
- `Node.bl_width_default`
- `Node.bl_width_min`
- `Node.bl_width_max`
- `Node.bl_height_default`
- `Node.bl_height_min`
- `Node.bl_height_max`

Inherited Functions

- `bpy_struct.as_pointer`
- `bpy_struct.driver_add`
- `bpy_struct.driver_remove`
- `bpy_struct.get`
- `bpy_struct.id_properties_clear`
- `bpy_struct.id_properties_ensure`
- `Node.poll_instance`
- `Node.update`
- `Node.insert_link`
- `Node.init`
- `Node.copy`
- `Node.free`

- `bpy_struct.id_properties_ui`
- `bpy_struct.is_property_hidden`
- `bpy_struct.is_property_overridable_library`
- `bpy_struct.is_property_readonly`
- `bpy_struct.is_property_set`
- `bpy_struct.items`
- `bpy_struct.keyframe_delete`
- `bpy_struct.keyframe_insert`
- `bpy_struct.keys`
- `bpy_struct.path_from_id`
- `bpy_struct.path_resolve`
- `bpy_struct.pop`
- `bpy_struct.property_overridable_library_set`
- `bpy_struct.property_unset`
- `bpy_struct.type_recast`
- `bpy_struct.values`
- `Node.socket_value_update`
- `Node.is_registered_node_type`
- `Node.poll`
- `Node.draw_buttons`
- `Node.draw_buttons_ext`
- `Node.draw_label`
- `Node.debug_zone_body_lazy_function_graph`
- `Node.debug_zone_lazy_function_graph`
- `Node.poll`
- `Node.bl_rna_get_subclass`
- `Node.bl_rna_get_subclass_py`
- `NodeInternal.poll`
- `NodeInternal.poll_instance`
- `NodeInternal.update`
- `NodeInternal.draw_buttons`
- `NodeInternal.draw_buttons_ext`
- `NodeInternal.bl_rna_get_subclass`
- `NodeInternal.bl_rna_get_subclass_py`
- `ShaderNode.poll`
- `ShaderNode.bl_rna_get_subclass`
- `ShaderNode.bl_rna_get_subclass_py`