Eevee Rendering Engine

Fundamentals

HIRAKO SAN

Table of Contents

Headless Rendering

Introduction

Eevee is the new real-time render engine for Blender. Using OpenGL, it focuses on speed and interactivity while achieving the goal of previewing PBR materials. It is designed to be a viewport engine but also offers high quality output for final renders.
Rendering is the process of creating a 2D image or video, from your 3D scene. The rendered images are based on four main factors:

- Camera
- Lighting
- Materials applied to objects
- Renderer settings.

Various complex calculations are made based on those factors. This process may take some time depending on the complexity of the scene and your hardware.

Render Engines

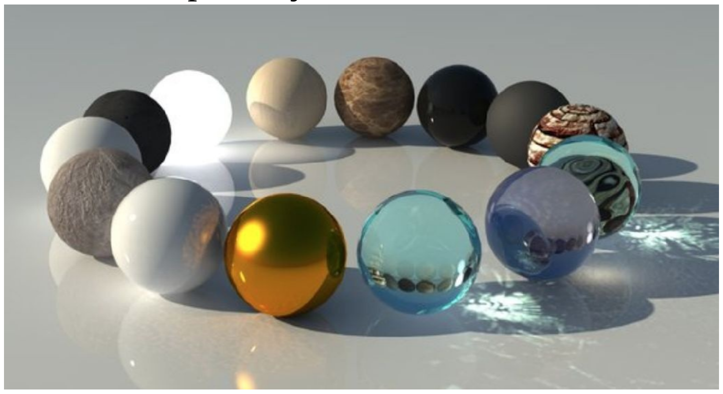
Renderers are programs that turn meshes, materials and lights into images. Some renderers may be better at certain things than others due to the math they use or core principles around which they were written. Blender ships with two renderers:

- Eevee
- Cycles

More renderers from third-party developers can also be added using Add-ons. This book focuses on the Eevee engine, we will start explaining how Materials work.

Chapter 1 - Materials

Materials define the substance qualities of an object. It is represented by its surface qualities such color, shininess, reflectance, but it can also exhibit more complicated effects like transparency, diffraction or subsurface scattering.

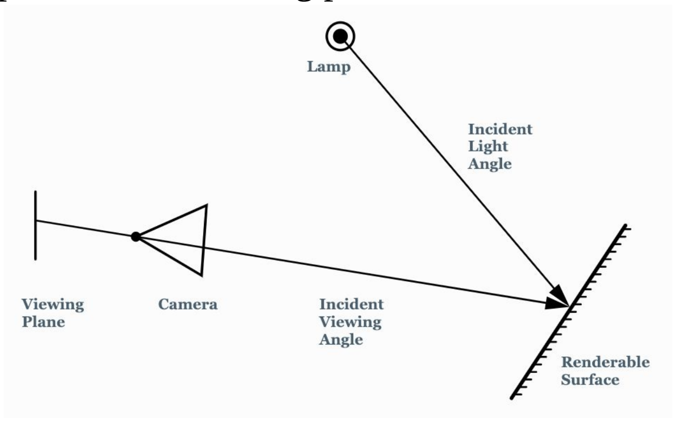


Materials may also contain displacement information to described how the material raises and lowers the surface of the material.

The basic material is uniform across each face of the object.. However, different faces of the object may use different materials. Materials can also have associated textures.

1.1 How Materials Work

To design effective materials, you must understand how simulated light and surfaces interact with the rendering engine and how material settings control those interactions. A deep understanding of the Eevee engine will help you get the most of it.
The rendered image is a projection of the scene onto an imaginary surface called the viewing plane. The viewing plane is analogous to the film in a traditional camera, except that it receives simulated light.
To render the image of a scene, the engine determines what light from the scene is arriving at each point on the viewing plane.



The shading (or coloring) of the object during render will take into account the base color and the light intensity.

1.2 Material Settings

1.2.1 Blend Modes

After calculating the color of a surface, Eevee needs to know how to add it to the color buffer. Depending on the blend mode the final color will be different.

1.2.2 Opaque

The previous color will be overwritten by the surface color. The alpha component is ignored. This is the fastest option.

1.2.3 Additive

The surface color will be added to the previous color. The alpha value will be used to mix surface color with the neutral color (black)

1.2.4.Multiplicative

The previous color will be multiplied by the surface color. The alpha value will be used to mix surface color with the neutral color white (1.0, 1.0, 1.0).

1.2.5 Alpha Clip

The previous color will be overwritten by the surface color, but only if the alpha value is above the clip threshold.

1.2.6 Alpha Hashed

The previous color will be overwritten by the surface color, but only if the alpha value is above a random clip threshold. This statistical approach is noisy but is able to approximate alpha blending without any sorting problem. Increasing the sample count in the render settings will reduce the resulting noise.

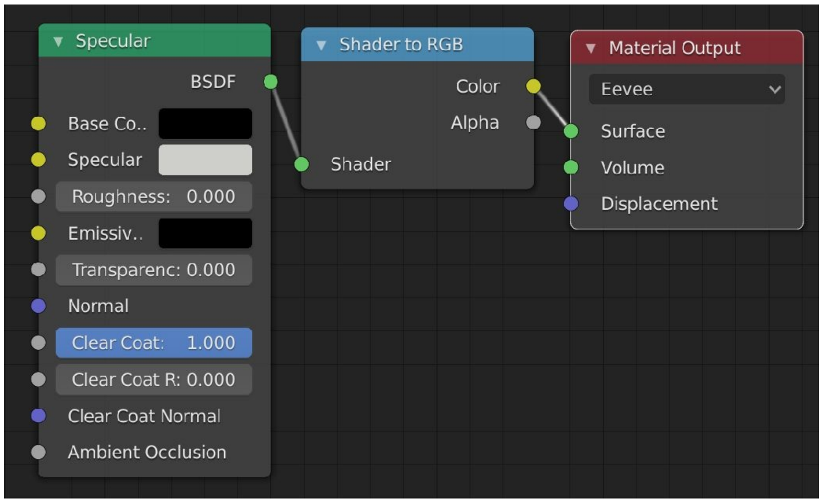
1.2.7 Alpha Blending

Use alpha blending to overlay the surface color on top of the previous color.

1.3 Nodes

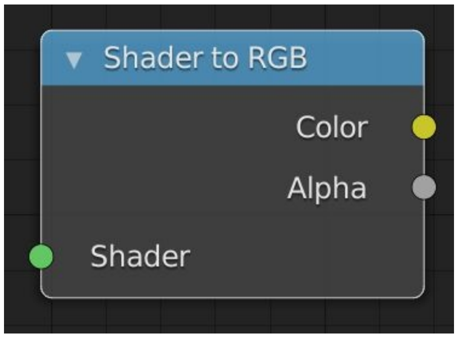
In Blender 2.8, the nodes system has been revamped, it splits the nodes panel into multiple editors: Shader, Compositing, and Texture editor. The rational is to make the other types of nodes easier to discover.
Many nodes supported by the Cycle renderer will work the same with Eevee. However, some features are missing or the result might differ. See the Node support section at the end of this chapter to understand the caveats.
Two new Nodes have been introduced and will work exclusively with Eevee.

New Eevee Nodes



1.3.1 Shader To RGB

Eevee supports the conversion of BSDF outputs into color inputs to make any kind of custom shading.



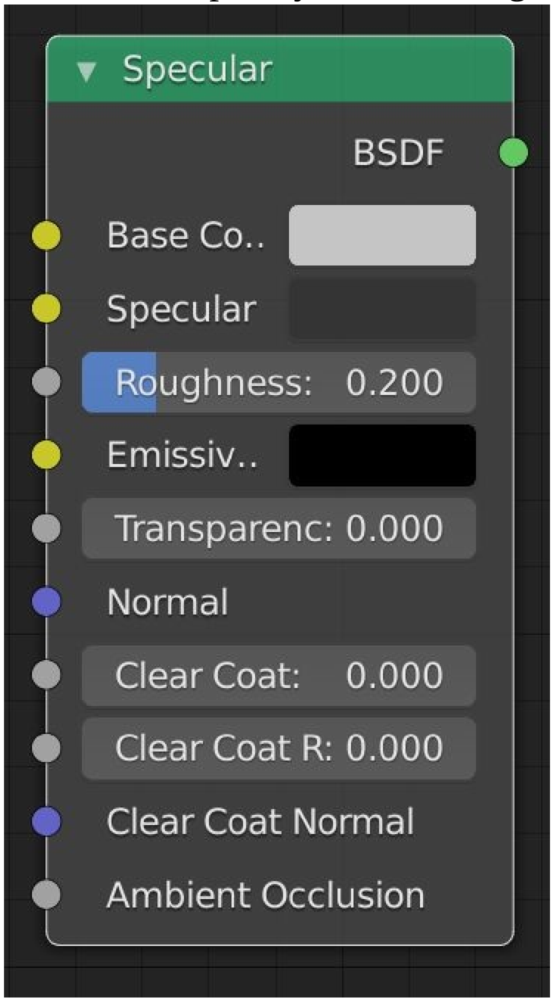
Note that the Shader to RGB node is breaking the PBR pipeline and thus makes the result unpredictable when other effects are used. This might be the case if you use effects that need temporal accumulation to converge. Namely ambient

occlusion, contact shadows, soft shadows, screen space refraction.
For instance if you quantize the result of the ambient occlusion you will not get a fully quantized output but an accumulation of a noisy quantized output which may not converge to a smooth result.
Also note that any upstream BSDF will invisible to the following effects:

- Screen Space Reflection
- Subsurface Scattering

1.3.2 Specular BSDF

The Specular BSDF combines multiple layers into a single easy to use node.



It is similar to the Principled BSDF node but uses the specular workflow instead of the metallic. It has less parameters and supports less features. The specular workflow works by specifying the facing and normal) reflection color. The result may not be physically accurate as there is no energy conservation.

Inputs Properties

Base Color

Diffuse surface color. For conductor materials (metals) it should be black.

Specular Color

Amount of specular reflection. Specifies facing (along normal) reflectivity. Conductor materials (metals) can have colored specular reflection.

Roughness

Specifies microfacet roughness of the surface for diffuse and specular reflection.

Emissive Color

Color of the emitted light. This light is added to the BSDF result.

Transparency

Transparency factor. It is the inverse of the alpha channel (1 - alpha) you can find in an image. Use an Invert node to convert alpha to transparency. This will only have an effect if the material uses a blend mode other than opaque.

Normal

Controls the normals of the base layers.

Clear Coat

Extra white specular layer on top of others. This is useful for materials like car paint and the like.

Clear Coat Roughness

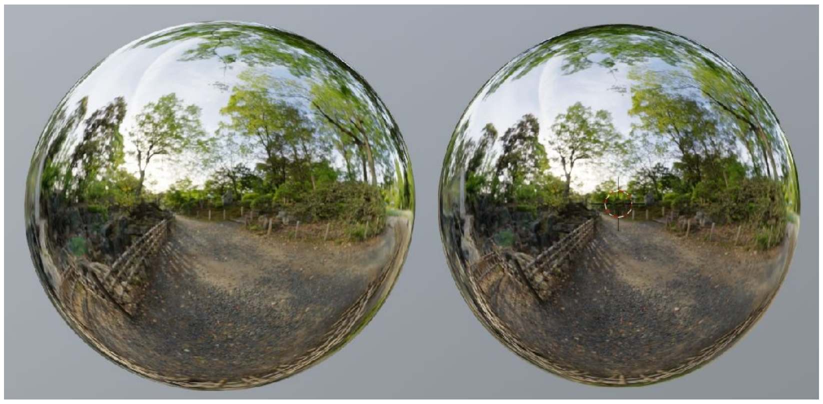
Roughness of clearcoat specular.

Clear Coat Normal

Controls the normals of the Clear coat layer.

Ambient Occlusion

Amount of occlusion to apply to indirect lighting. Usually a bake ambient occlusion map. The final occlusion factor is the minimum of this input and the runtime ambient occlusion effect.
As an example, two spheres, on the left is a Specular BSDF shader along with Shader to RGB node. On the right is applied a Principled BSDF shader.


With Eevee, the rendering is nearly immediate with quasi photorealist results.

1.4 Nodes Support

The following nodes are supported in Cycle, and also work with Eevee, just the following limitation.

1.4.1 Shaders Nodes

In most cases, shader nodes behave like in Cycles. Although most BSDFs are supported, many of them are approximations, not feature complete.

Diffuse BSDF

Roughness is not supported, and only allows Lambertian diffusion.

Emission

It is treated as indirect lighting and will only show up in *SSRs* and *Probes*.

Glass / Refraction BSDF

Does not refract lamps, and no support for Beckmann and GGX Multiscatter distribution.

Glossy BSDF

Does not support Beckmann and GGX Multiscatter distribution either, also does not support Ashikhmin-Shirley.

Subsurface Scattering

Random Walk sampling is not supported.
Per color channel Radius is specified by the default socket value. Any link plugged into this socket gets ignored.
If Separate Albedo is off, then Texture Blur will be treated as 1.0. Texture Blur is not accurate for any value other than 0.0 and 1.0.

Transparent BSDF

Only supports monochromatic transparency.. The color input is converted to float alpha value. Make sure the *Material blend mode* is not *Opaque*, otherwise transparency won’t take effect.

Translucent BSDF

Does not diffuse the light inside the object. It only lights the object with reversed normals.

Principled BSDF

Cumulate limitations from Diffuse BSDF, Glossy BSDF, Refraction BSDF and Subsurface Scattering. Anisotropy is not supported. The Sheen layer is just an approximation.

Volume Absorption

Approximate light absorption of the surrounding volume objects. This makes the volumes more opaque to light. This is a very expensive option.

Volume Scatter and Principled Volume

The anisotropy parameter will be mixed and average for all overlapping volumetric objects, which is not physically correct hence differs from Cycles.

Not supported

Holdout, Anisotropic BSDF, Toon BSDF, Hair BSDF, Velvet BSDF and Principled Hair BSDF

1.4.2 Input Nodes

Ambient Occlusion

All parameters will have no effects except for Normal and Color. This is because Ambient Occlusion is computed before evaluating this node and it uses the scene settings for that.

Camera Data

Everything is supported.

Geometry

Everything is supported except *Pointiness*.

Attribute

Defaults to active UV layer. Only *density*, *color*, *flame* and *temperature* attributes are supported. *UVs* and *Vertex Color* layers are supported.

Fresnel

Everything is supported.

Hair Info

The Random output uses a different RNG algorithm. Range and statistical distribution of the values should be the same but the values will be different.

Layer Weight

Everything is supported.

Light Path

Eevee is not a raytracer renderer, hence has no proper concept of Rays. However, some of the outputs are supported. This node makes it possible to tweak indirect lighting in the shader.
Only a subset of the outputs is supported and the ray depth doesn't have the exact same meaning:

- Is Camera: Supported
- Is Shadow: Supported
- Is Diffuse: Supported
- Is Glossy: Supported
- Is Singular: Not supported. Same as Is Glossy
- Is Reflection: Not supported. Same as Is Glossy
- Is Transmission: Not supported. Same as Is Glossy
- Ray Length: Not supported. Defaults to 1.0
- Ray Depth: Indicate the current bounce when baking the light cache
- Diffuse Depth: Same as Ray Depth but only when baking diffuse light
- Glossy Depth: Same as Ray Depth but only when baking specular light
- Transparent Depth: Not supported. Defaults to 0
- Transmission Depth: Not supported. Same as Glossy Depth

Object Info

Everything is supported.

Not supported

Bevel, Particle Info

Tangent

Everything is supported.

Texture Coordinate

From Dupli and coordinate from custom Object is not supported.

UV Map

From Dupli is not supported.

Wireframe

Pixel size option does not give the exact same output as with Cycles. The width can may differ.

1.4.3 Other Nodes

Bump

Imprecision due to less precise derivatives.

Image Texture

Smart Interpolation always uses Cubic interpolation. Artifact present using Tube or Sphere projection with linear interpolation. This is due to hardware mip-mapping and Anisotropic filtering. This kind of artifacts will also be visible if the texture coordinates provided are not continuous. Using Box projection with Extend type set to Clip or Extend is not supported. It will always use Repeat instead.

Material Output

Displacement output behavior won't work properly.

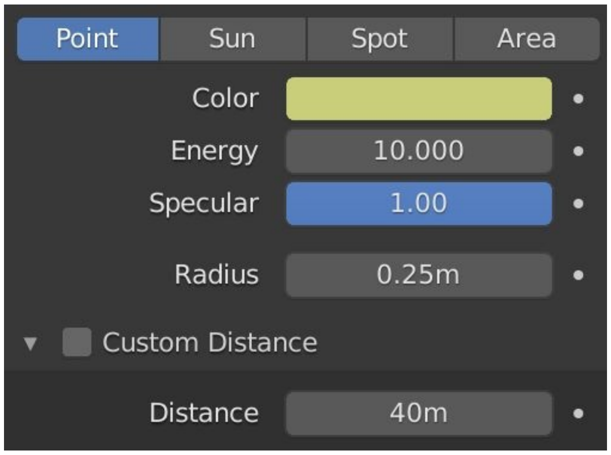
Not supported

Light Falloff, Displacement/Vector Displacement, IES Texture, Sky Texture,

Chapter 2 - World

The world environment can emit light, ranging from a single solid color, to arbitrary textures. It works the same as the world material in Cycles.
In Eevee, the world lighting contribution is first rendered and stored into smaller resolution textures before being applied to the objects. This makes the lighting less precise than Cycles.

2.1 Lights



Color

Energy 10.000

Specular 1.00

Radius 0.25m

Custom Distance

Distance 40m

Eevee is supposed to have a concept of Lamps, which is probably going to come in an upcoming release, but as of 2.8, Lights cannot be assigned nodes.
Lights will likely support emission shaders in the future. Lamps are another way to add light into the scene.
The difference is that they are not directly visible in the rendered image, and can be more easily managed as objects of their own type.

Common Settings

Type

Defines the physical light shape: Area, Point, Spot, Sun, Spot.

Energy

Light intensity. Affect both specular and diffuse light. Note: Can be negative but breaks PBR.

Specular

Specular Light intensity multiplier. Use it for more artistic control. Setting this to anything but 1.0 will yield non-photorealistic result.

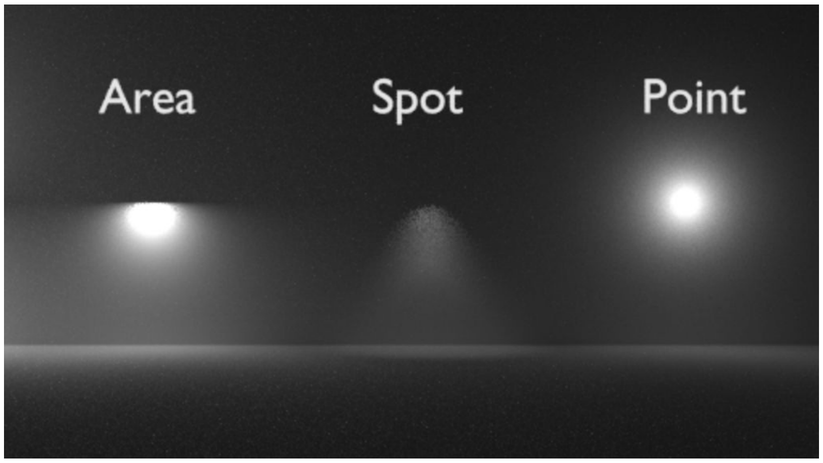
Radius

Also called Size for lights of type Area, increasing this will result in softer shadows and shading.

Custom Distance

If enabled, uses Distance as the custom attenuation distance instead of global light threshold. The Distance specify where light influence will be set to 0.

2.2 Light Types



2.2.1 Point Light

Point lights emit light equally in all directions. By setting the Size larger than zero, they become spherical lamps, which give softer shadows and shading. The strength of point lights is specified in Watts.

2.2.2 Spot Light

Spot lights emit light in a particular direction, inside a cone. By setting the Size larger than zero, they can cast softer shadows and shading.

2.2.3 Area Light

Area lights emit light from a rectangular or elliptic area. They can be of different shape.

Light Shape

Shape of the area lamp. The size parameter defines the size of the cone, while the blend parameter can soften the edges of the cone.

Square

The shape of the lamp can be represented as a square and changed with the Size property.

Rectangle

The shape of the lamp can be represented as a rectangle and changed with the X and Y values.

Disk

The shape of the lamp can be represented as a disk and changed with the Size property.

Ellipse

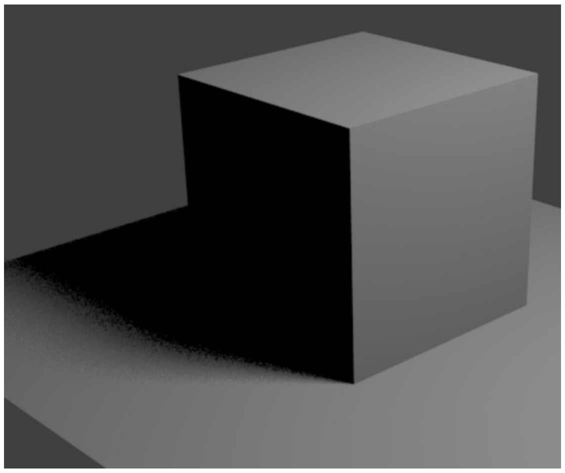
The shape of the lamp can be represented as a ellipse and changed with the X and Y values.

2.2.4 Sun Light

Sun lights emit light in a given direction. Their position is not taken into account; they are always located outside of the scene, infinitely far away, and will not result in any distance falloff.
Because they are not located inside the scene, their strength uses different units, and should typically be set to lower values than other lights.

2.3 Shadows

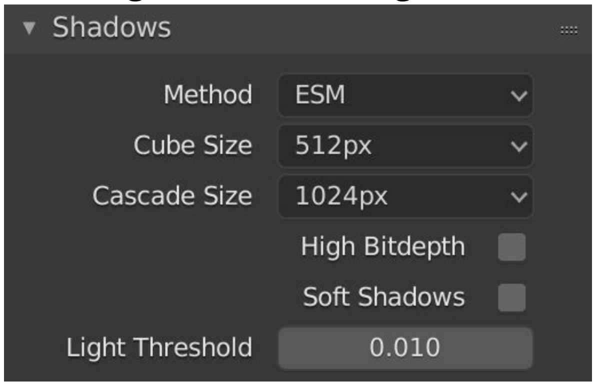
Eevee uses Shadow Mapping techniques to properly shadow the light coming directly from light objects.



A shadow map is a texture that stores the nearest occluder from the light position. Eevee also filters the shadow maps in order to smooth out the pixelated appearance.

2.3.1 Global Settings

Find the global shadow settings in the rendering editor.



▾ Shadows

Method	ESM
Cube Size	512px
Cascade Size	1024px
High Bitdepth	☐
Soft Shadows	☐
Light Threshold	0.010

Method

Select the shadow map type. It changes how shadows are stored and filtered.

ESM

They are fast to filter but suffer from light leaking near the occluder. This can be minimized by increasing the Exponent parameter. Another issue is the artifacts present at depth discontinuity. Unfortunately, there is no workaround for this issue and the only way to minimize it is to reduce the Soft parameter.

VSM

Filters nicely and gives smooth shadow map appearance across the whole shadow range. However it suffers from visible grainy artifacts when using low bit depth. It is also prone to light leaking when two occluders overlap. In this case the shadows can be over darkened to reduce the leak, by using the Bleed Bias parameter. VSM uses twice as much memory as ESM, and are also slower.

Cube Size

Size of the shadow cubemaps used to shadow Point, Area and Spot lights. Higher shadow map size will give higher precision and sharper shadows.

Cascade Size

Size of one cascade used by Cascaded Shadow Maps. This is only for Sun lamps.

High Bit depth

This option can help reduce some artifacts due to float imprecision inside the shadow maps. This option effectively double the memory usage of shadow maps and will slow down their update.

Soft Shadows

Randomize the shadow maps origin to create soft shadows. It needs a lot of samples to get rid of the banding.

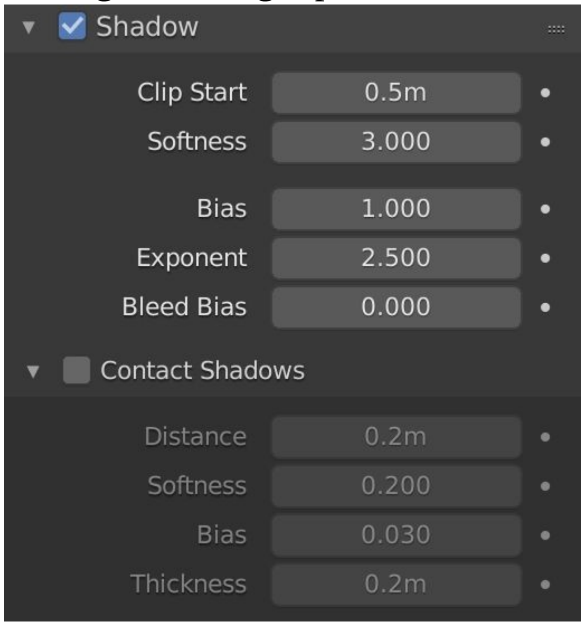
Light Threshold

In order to avoid costly setup time, this distance is first computed automatically based on a light threshold.
The distance is computed at the light origin and using the inverse square falloff. The setting can be found inside the Render Settings panel, Shadow tab.
This light threshold does not take the light shape into account and may not suit every case. The influence distance is also used as shadow far clip distance, which might affect how shadows looks. This influence distance does not concerns sun lights that still have a far clip distance.

Note that the Soft Shadows method is not physically based and will not match Cycles for very large lamps. Also note that
A 512px cubemap is 6 x 512 x 512 pixels.
Tweaking the Size parameters can have a big impact on memory consumption and performance.

2.3.3 Light Shadow Settings

Find the light shadow settings in the light panel, below the light settings.



Shadow ✓

Clip Start 0.5m
Softness 3.000
Bias 1.000
Exponent 2.500
Bleed Bias 0.000

Contact Shadows

Distance 0.2m
Softness 0.200
Bias 0.030
Thickness 0.2m

Common Parameters

Clip Start

Distance from the light object at which the shadow map starts and ends. Any object before this distance will not appear to cast shadows. Clip End will only appear for sun lamps.

Softness

Size of the filter applied to the shadow map. This filter size is independent from the shadow map resolution. Higher filter size can have a big impact on performance. There is a maximum cap to filter size (in pixels) that depends on shadow resolution.

Bias

Bias applied to the depth test to reduce self shadowing artifacts.

Exponent

Exponent applied to ESM to reduce light leaking.

Bleed Bias

Bias applied to VSM to reduce light leaking.

Contact Shadows

This type of shadows exists to fix light leaking caused by Bias or shadow map undersampling. They uses the depth buffer to find occluders, just like Screen Space Reflections. However, exactly like Screen Space Reflections, they suffer from the same limitations: unknown object thickness, effect disappearing at screen edges.
The distance of action of Contact Shadows should ideally remain quite small. They are not accurate enough to shadow the entire scene.

Distance

World space distance in which to search for screen space occluder.

Softness

Control how soft the contact shadows will be. Contact shadow blurring does not match light’s physical size.

Bias

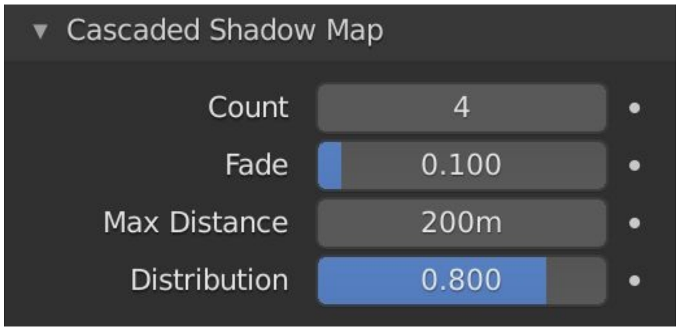
Bias applied to the ray tracing to reduce self shadowing artifacts.

Thickness

Pixel thickness used to detect occlusion. Treat any potential occluder to be this thick.

Cascaded Shadow Map

These special kind of shadow maps only apply to Sun lights because they can shadow large scenes by distributing multiple shadow maps over the frustum range.


▾ Cascaded Shadow Map

Count 4

Fade 0.100

Max Distance 200m

Distribution 0.800

Each cascade covers a different portion of the view frustum. Do note that cascade shadow map are always updated because they are view dependent. This means they have a high performance impact.

Count

Number of cascade to use. More cascade means better precision but slower update.

Fade

Fade transition area between two cascades. Higher values means less overall resolution because cascades need to overlap.

Max Distance

Distance away from the view origin (or camera origin if in camera view) to cover with the cascade.
If the view far clip distance is lower than Max Distance, the lowest of the two will be used. Only works in perspective view.

Distribution

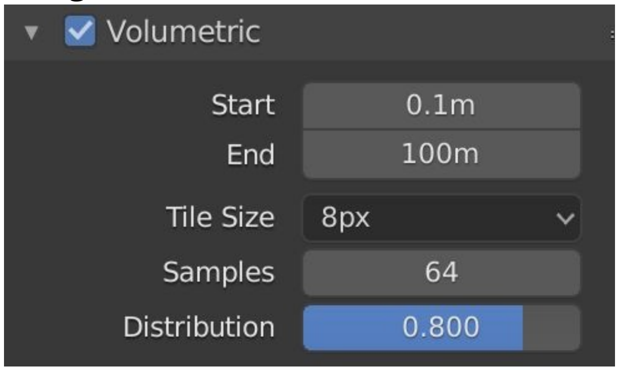
Puts more resolution towards the near clip plane. Only works in perspective view.

Lights shadow Limitations

- Shadows are not supported on light instances (dupli objects, group instancing).
- Only 128 active lights can be supported by Eevee in a scene.
- Only 8 Shadowed sun lights can be supported at the same time.

2.4 Volumetrics

Eevee simulate volumetric scattering by evaluating all volume objects inside a the view frustum.
It uses several 3D texture leveraging video memory. The textures dimensions can be tweaked using the Tile Size and Samples parameters.Object volumes have limitations, listed further.
Volumetrics can be configured under the Render editor.



Volumetric

Start	0.1m
End	100m
Tile Size	8px
Samples	64
Distribution	0.800

Start

Start distance of the volumetric effect.

End

End distance of the volumetric effect.

Tile Size

Control the quality of the volumetric effects. Lower size increase video memory usage and quality. This is the size in pixels, of a volumetric cell.

Samples

Number of samples to compute volumetric effects. Higher count increase video memory usage and quality. These samples are distributed along the view depth (view Z axis).

Distribution

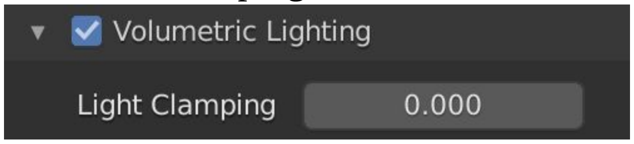
Blend between linear and exponential sample distribution. Higher values puts more samples near the camera.

2.5 Volumetric Lighting

Let the volume scattering scatter light in the scene. Unnecessary if no Volume Scatter is present in the scene.

Light Clamping

Clamp light contribution to the volume scattering effect. Reduce flickering and noise. Set to 0.0 to disable the clamping.

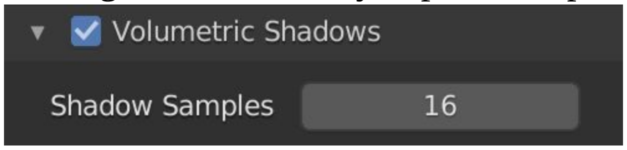


☑ Volumetric Lighting

Light Clamping 0.000

2.6 Volumetric Shadows

Approximate light absorption of the surrounding volume objects. This makes the volumes more opaque to light. This is a very expensive option.



✓ Volumetric Shadows

Shadow Samples 16

Shadow Samples

Number of samples to compute volumetric shadowing.

2.7 Volumetrics Limitations

- Only single scattering is supported.
- Volumetrics are rendered only for the camera “rays”. They don’t appear in reflections/refractions and probes.
- Volumetrics don’t receive Light from light grids but does receive from the world probe.
- Volumetric shadowing does only work on other volumetrics.
- Volumetric shadowing does only work for volumes inside the view frustum.
- Volumetric lighting does not respect the lights shapes. They are treated as point lights.

Chapter 3 - Screen Space Effects

Eevee is not a ray tracing engine. It cannot do ray-triangle intersection; instead it uses the depth buffer as an approximated scene representation. This reduces the complexity of scene scale effects and allows high performances.

However, only what is in inside the view can be considered when computing these effects.

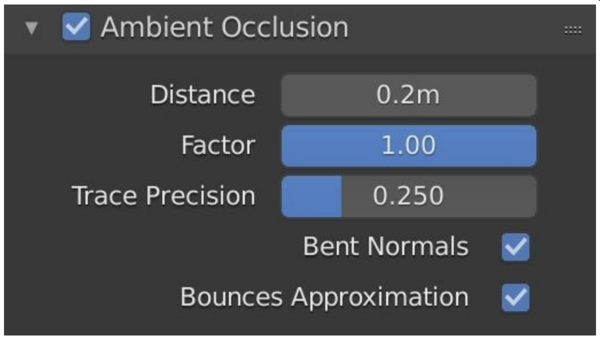
Also, since it only uses one layer of depth, only the front-most pixel distance is known.

These limitations creates a few problems:

- The screen space effects disappear when reaching the screen border. This can be partially fixed by using the overscan feature.
- The screen space effects don't know how deep (or thick) the objects are. This is why most effects have a thickness parameter to control how to consider potential intersected pixels.
- Blended surfaces are not considered by these effects. They are not part of the depth prepass and does not appear in the depth buffer.

3.1 Ambient Occlusion

Ambient occlusion is computed using GTAO and applied to indirect lighting. The bent normal option will make the diffuse lighting come from only the least occluded direction.
Ambient occlusion can be rendered as a separate pass in the Render Layers panel. To enable it, find Ambient Occlusion in the Rendering Editor.



Distance 0.2m
Factor 1.00
Trace Precision 0.250
Bent Normals ✓
Bounces Approximation ✓

Distance

Distance of object that contribute to the ambient occlusion effect.

Factor

Blend factor for the ambient occlusion effect.

Trace Precision

Increase precision of the effect but introduce more noise and lower the maximum trace distance.
Increased precision also increases performance cost. Lower precision will also miss occluders and lead to under shadowing.

Bent Normals

Compute the least occluded direction. This direction can be used to sample the diffuse irradiance in a more realistic way.

Bounce Approximation

An approximation to simulate light bounces giving less occlusion on brighter objects.

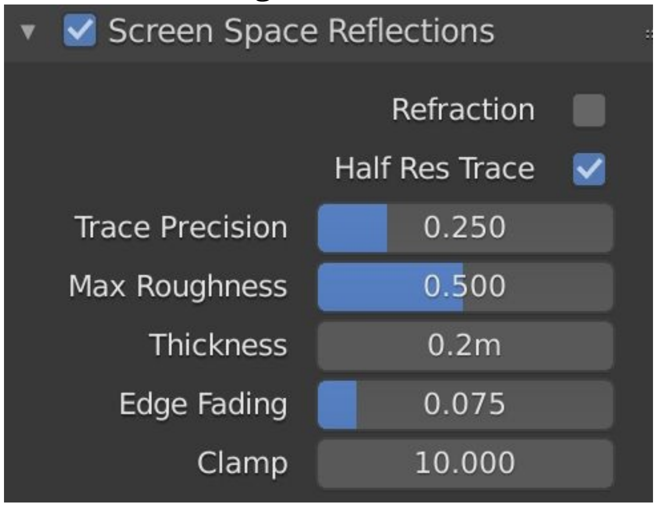
It only takes into account the surface color and not its surroundings. This is not applied to the ambient occlusion pass.

Ambient Occlusion Limitations

- Objects are treated as infinitely thick, producing overshadowing if the Distance is really large.

3.2 Reflections

When this effect is enabled, all Materials will use the depth buffer and the previous frame color to create more accurate reflection than reflection probes.
If a Reflection Plane is near a reflective surface, it will be used as the source for tracing rays more efficiently and fix the partial visibility problem.
However, the reflected color will not contain the following effects: Subsurface scattering, volumetrics, screen space reflections, screen space refractions. To enable reflections, use the Rendering editor.



✓ Screen Space Reflections

Refraction ☐

Half Res Trace ✓

Trace Precision	0.250
Max Roughness	0.500
Thickness	0.2m
Edge Fading	0.075
Clamp	10.000

Refractions

Screen space refractions work the same way as screen space reflections and use same parameters. But they are not enabled by default on all surfaces.
Enabling it will have a small performance cost. You need to enable them in Material Properties ‣ Options.
Materials using screen space refractions will not be able to cast screen space reflections.

Half Resolution Trace

Use half resolution ray tracing. Only cast a ray for every fourth pixels. Enabling this option reduces drastically video memory usage and increase performances at the cost of quality.

Trace Precision

Increase precision of the ray trace but introduce more noise and lower the maximum trace distance. Increased precision also increases performance cost.

Thickness

How thick to consider the pixels of the depth buffer during the tracing. Higher values will stretch the reflections and add flickering. Lower values may make the ray miss surfaces.

Edge Fading

Smoothly fade out the reflected pixels if they are close to a screen edge. The unit is in screen percentage.

Clamp

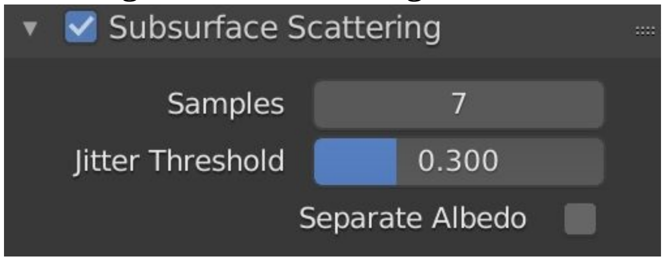
Clamp the reflected color intensity to remove noise and fireflies.

Reflections Limitations

- Only one glossy BSDF can emit screen space reflections.
- The chosen BSDF is currently arbitrary chosen.
- Only one refraction event is correctly modeled.
- Screen Space Reflections will reflect transparent object but without accurate positioning due to the one layer depth buffer.

3.3 Subsurface Scattering

This effect mimic real subsurface scattering by blurring the diffuse lighting in screen space. To enable it, go to the Rendering editor.



Subsurface Scattering ✓

Samples 7

Jitter Threshold 0.300

Separate Albedo

Samples

Number of samples to compute the scattering effect.

Jitter Threshold

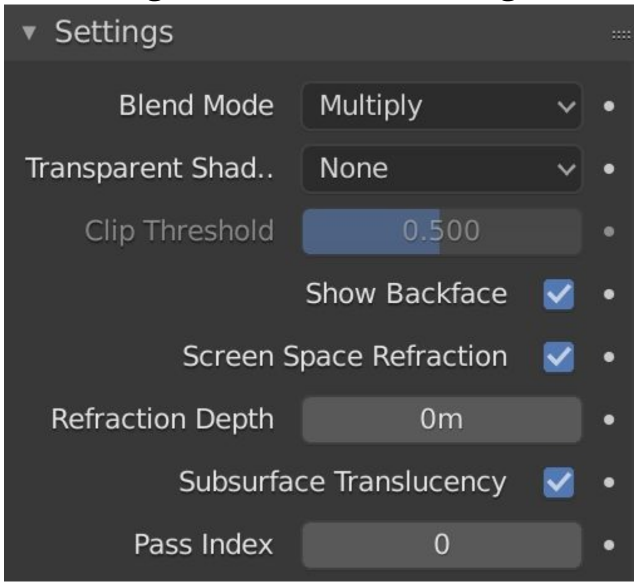
For the effect to be efficient, samples needs to be coherent and not random.
This can lead to cross shaped pattern when the scattering radius is high.
Increasing the Jitter Threshold will rotate the samples below this radius percentage in a random pattern in order to hide the visible pattern. This affects performance if the scattering radius is large.

Separate Albedo

Output the albedo of a BSSRDF in a separate buffer in order to not blur it.
The Texture Blur parameter require this option to be enable to work correctly.
This option increases the video memory usage but does not have a big impact on performance.

3.4 Subsurface Translucency

The Subsurface Translucency option needs to be enabled in order to make the light go through an object (like simulating a human ear lit from behind).
To enable it, go to the Shading Editor, find the setting in the Material panel.



▾ Settings

Blend Mode	Multiply
Transparent Shad..	None
Clip Threshold	0.500
Show Backface	✓
Screen Space Refraction	✓
Refraction Depth	0m
Subsurface Translucency	✓
Pass Index	0

Note that:

- This option does only work if Subsurface Scattering is enabled.
- This option only works with shadowed light and does not work with indirect lighting.

Chapter 4 - Indirect Lighting Cache

While not physically correct, all lighting that is not coming straight out from a light object is considered indirect lighting in Eevee.
That means distant HDRI lighting is considered as indirect lighting. Mesh objects using an Emission node is also considered as indirect lighting.
In Eevee, indirect lighting is separated into two component:

- Diffuse
- Specular

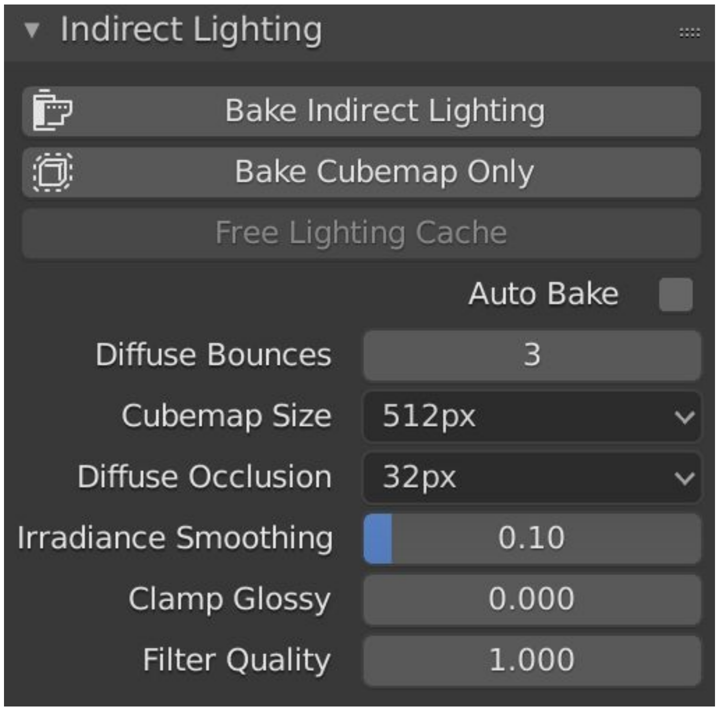
Both have different needs and representation. For efficiency, the indirect lighting data is precomputed on demand into a static lighting cache.
The light cache is static and needs to be computed before rendering. It cannot be updated per frame, unless via scripting. This limitation is being worked on and will be removed in future versions.
Only view independent lighting can be baked. This is why Reflection Planes are not stored inside the light cache.
The visibility and collections used during the baking process are the ones in the current Active View Layer.

Enable Light Bounce

To enable light bounces through large environment, the light baking process can be run multiple times while injecting previous bake result into the a bake.
Total baking time is more or less multiplied by the number of bounce. Light bounces only concerns diffuse lighting.

Indirect Lighting

To enable indirect lighting, find the setting in the Rendering editor, in the Render panel.



Auto Bake ☐
Diffuse Bounces 3
Cubemap Size 512px
Diffuse Occlusion 32px
Irradiance Smoothing 0.10
Clamp Glossy 0.000
Filter Quality 1.000

Auto Bake

Enabling this option will trigger baking when a probe is changed. Useful when positioning probes objects.

Diffuse Bounce

Number of bounce to compute when baking the diffuse irradiance.

Cubemap Size

Size of the reflection cubemaps.

Diffuse Occlusion Size

Each irradiance sample also store a shadow map that is used to minimize indirect light leaking. This parameter define the size of this shadow map.

Irradiance Smoothing

Smoother irradiance interpolation but introduce light bleeding. The irradiance visibility term can make the lighting not interpolate smoothly on some surfaces. This relax the weights so that the interpolation.

Clamp Glossy

Clamp pixel intensity to reduce noise inside glossy reflections from reflection cubemaps (0 to disabled).

Filter Quality

Take more samples during cubemap filtering to remove artifacts. For now, only have effect on cubemaps

Cubemap Display

Display the Reflection Cubemaps present in the cache directly in the 3D View.

Irradiance Display

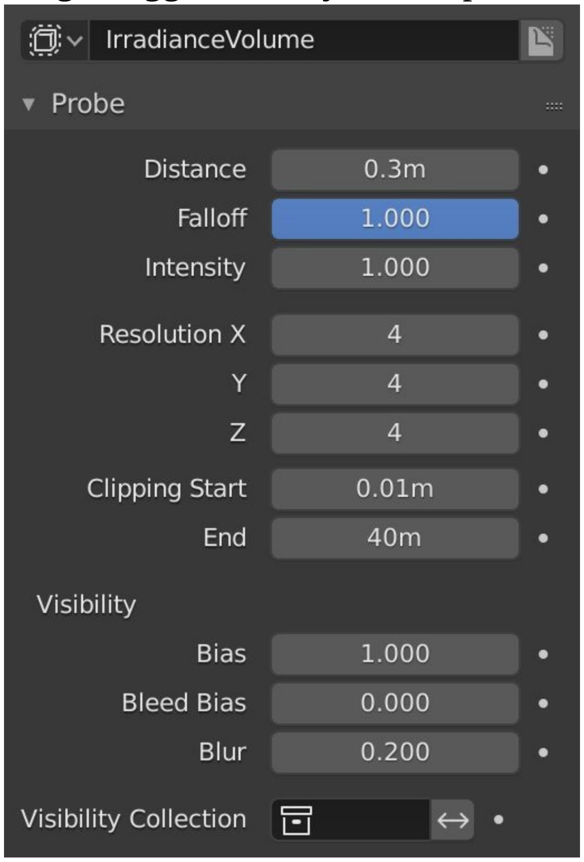
Display the Irradiance Samples present in the cache in the 3D View.

Chapter 5 - Light Probes

Light Probes provide a way to capture information about light passing through empty space in your scene
.
Like like light maps, they store *baked* information about lighting. While light maps store information about light hitting the surface of objects, probes store information about light passing through empty space in the scene.
An extremely simple scene showing light probes placed around two cubes
Light Probes are useful in to provide high quality lighting, such as indirect bounced light, on moving objects.
Probe objects are used by Eevee as support objects. They record lighting information locally in order to lit the scene using indirect lighting.
There are three different probe types. One for diffuse light lighting two for specular lighting.
They are meant to guide the engine to compute better lighting faster.

5.1 Irradiance Volumes

Diffuse indirect lighting is stored into volumetric arrays. Those arrays get defined when using Irradiance Volumes objects.
They give control on how arrays are placed in the world, we can also control their resolution.
Lighting computation happens on the little dots, visible when an Irradiance Volume object is selected.
If Ambient Occlusion is enabled, it gets applied to diffuse indirect lighting. When both Ambient Occlusion and *Bent Normals* are enabled, the indirect lighting will be sampled from the least occluded direction and appear more correct.
To add an Irradiance Volumes probe object, using the Modeling editor, make sure to be in Object mode, click *Add* -> *Light Probe* -> *Irradiance volume.*
To adjust the probe settings, trigger the Object data panel.


IrradianceVolume

Probe

Distance 0.3m
Falloff 1.000
Intensity 1.000

Resolution X 4
Y 4
Z 4
Clipping Start 0.01m
End 40m

Visibility
Bias 1.000
Bleed Bias 0.000
Blur 0.200

Visibility Collection

Probe Main Properties

Distance

A probe object only influences lighting of nearby surfaces. The influence area is defined by the Distance parameter and object scaling. It can be different, depending on the probe type.
For Irradiance Volumes, the influence inside the volume is always 100%.
The influence decays only outside the volume until the distance to the volume reaches the Distance parameter value.

Falloff

Defines the Percentage of the influence distance, during which the probe influence fades linearly.

Intensity

Defines the Intensity factor of the recorded lighting. Setting this parameter to anything other than 1.0 is not physically correct. Use it for artistic purpose, not for photorealism.

Resolution

Defines the Spatial resolution for Irradiance Volumes determined per probe. The local volume is divided into a regular grid of the specified dimensions. One irradiance sample will be computed for each cell in this grid.

Clipping

Define the near and far clip distances when capturing the scene. Note that the distance is measured from each sample, not from the grid origin.

Visibility Collection

In some cases, it is useful to limit which objects appear in the light probe’s captured lighting. For example, an object that is too close to a capture point might be better excluded.
This is what the visibility collection does. Only objects that are in this collection will be visible when this probes will capture the scene.
It is also possible to invert this behavior and effectively hide the objects inside this collection.
Note that this is only a filtering option. This means that if an object is not visible at render time it won’t be visible in during the probe render.

Probe Visibility

For every grid point a small Variance Shadow Map is rendered. This visibility cubemap is used to reduce light leaking behind occluders.
You can tweak the size of this map inside the render settings and tweak the bias and blur factors per Grid inside the Probe Properties tab.

Bias

Reduce self shadowing.

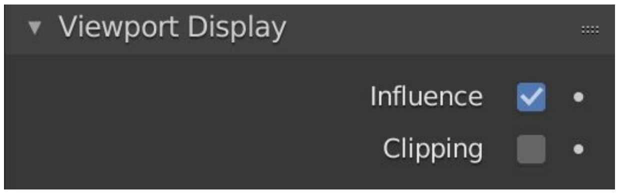
Bleed Bias

Increase the "contrast" of the depth test result.

Blur

Amount of blur to apply when filtering the visibility shadow map. Does not increase runtime cost, has a small effect on baking time.

Viewport Display



Influence

Enable it for the influence bounds to display in the 3D View. The inner sphere is where the falloff starts.

Clipping

Enable it for the clipping distance to display in the 3D View.

Baking

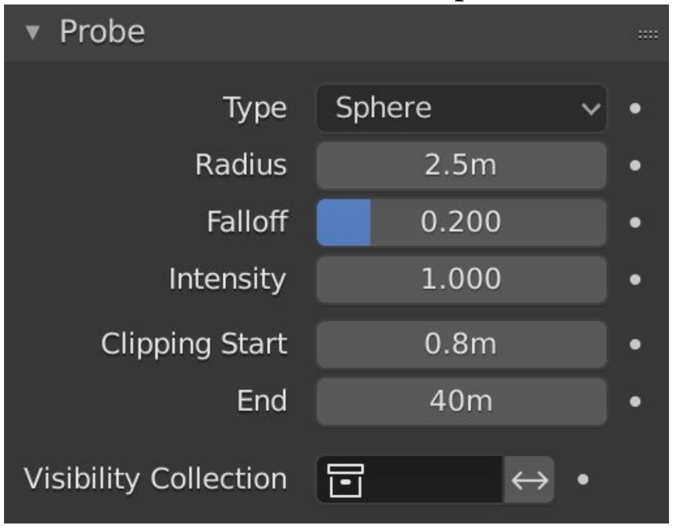
Bake indirect lighting in the Render panel, Indirect Lighting settings, bake cube map only.

5.2 Reflection Cubemaps

Specular Indirect Lighting is stored into an array of cubemaps. These are defined by the Reflection Cubemap objects. They specify where to sample the scene's lighting and where to apply it.
Screen Space Reflection are much more precise than reflection cubemaps. If enabled, they are applied in priority and we only use cubemaps as fall back if a ray misses.
If Ambient Occlusion is enabled, it will be applied in a physically plausible manner to specular indirect lighting.

Blending

The lighting values from a Reflection Cubemap will fade outwards until the volume bounds are reached.
They will fade into the world's lighting or another Reflection Cube map lighting.
If multiple Reflection Cubemap overlaps, smaller (in volume) ones will always have more priority.
If an object is not inside any Reflection Cubemap influence, or if the indirect lighting has not been baked, the world's cubemap will be used to shade it.


Probe

Type: Sphere
Radius: 2.5m
Falloff: 0.200
Intensity: 1.000
Clipping Start: 0.8m
End: 40m
Visibility Collection

Radius/Size

A probe object only influences the lighting of nearby surfaces.
This influence zone is defined by the radius or size parameter (radius for sphere, size for box type) and object scaling.

The influence volume is centered on the probe’s origin.

Falloff

Percentage of the influence distance during which the influence of a probe fades linearly.

Intensity

Intensity factor of the recorded lighting. Making this parameter anything other than 1.0 is not physically correct. Use it for tweaking or artistic purpose.

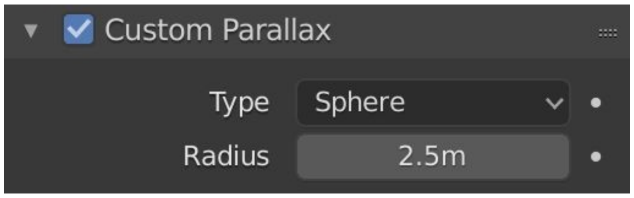
Clipping

Define the near and far clip distances when capturing the scene.

Visibility Collection

Sometimes, it is useful to limit which objects appear in the light probe’s captured lighting.
For instance, an object that is too close to a capture point might be better excluded. This is what the visibility collection does.
Only objects that are in this collection will be visible when this probes will capture the scene.
There is also an option to invert this behavior and effectively hide the objects inside this collection.

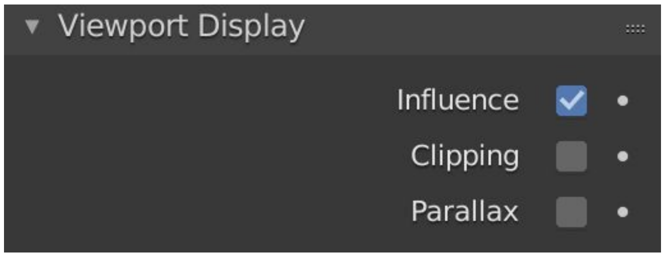
Custom Parallax


Custom Parallax
Type: Sphere
Radius: 2.5m

By default, the parallax volume is the same as the influence volume is. The parallax volume is a volume on which is projected the recorded lighting. It should roughly fit it surrounding.
In some cases it may be better to adjust the parallax volume without touching the influence parameters. In this case, just enable the Custom Parallax and change

the shape and distance of the parallax volume independently.

Viewport Display



Influence ✓ ; Clipping ☐ ; Parallax ☐

Influence

Show the influence bounds in the 3D View. The inner sphere is where the falloff starts.

Clipping

Show the clipping distance in the 3D View.

Parallax

Show the Custom Parallax shape in the 3D View.

5.3 Reflection Planes

These special types of Probe object is suited to smooth planar surfaces. They basically capture the whole scene with a flipped camera.
Using reflection planes is really heavy on the render time because the scene needs to be rendered as many times as there is Reflection Planes in the view.
Unless Screen Space Reflection is enabled, Reflection Planes only work on specular surfaces that have there roughness around 0.
If Screen Space Reflection is enabled, Reflection Planes will serves as support buffers.
This accelerate the tracing process and complete the missing data from the view space. and make reflection more correct for the affected surface that have medium roughness and disturbed normals (i.e. normal maps).

Placement

If Backface Culling is not enabled, snapping the Reflection Plane to the planar surface will effectively capture the underneath of the surface.
Manually moving the Reflection Plane above the surface enough for it to not appear in the capture. Alternatively you can put the floor object inside a collection and use this collection as Visibility Collection (inverted) inside the Reflection Plane's probe settings.

Distance

A probe object only influences the lighting of nearby surfaces. This influence zone is defined by the Distance parameter and object scaling. It is a bit different, depending on the probe type.
For Reflection Planes the influence distance is the distance from the plane. Only surfaces whose normals are aligned with the Reflection Plane will receive the captured reflection.

Falloff

Percentage of the influence distance during which the influence of a probe fades linearly.
Also defines how much shading normals needs to be aligned with the plane to receive reflections.

Clipping Offset

Define how much below the plane the near clip is when capturing the scene. Increasing this can fix reflection contact problems.

Visibility Collection

Sometimes, it is useful to limit which objects appear in the light probe’s captured lighting.
For example, an object that is too close to a capture point might be better excluded.
This is what the visibility collection does. Only objects that are in this collection will be visible when this probes will capture the scene.
There is also an option to invert this behavior and effectively hide the objects inside this collection.

Viewport Display

Arrow Size

Size of the arrow showing the reflection plane normal.

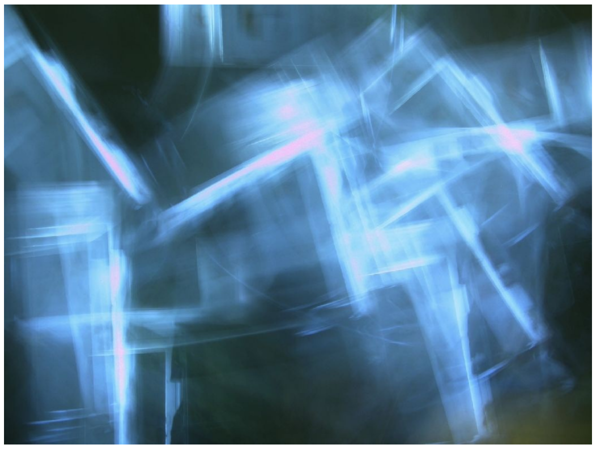
Show Data

Show the captured reflected image onto a fully reflective plane in the 3D View.

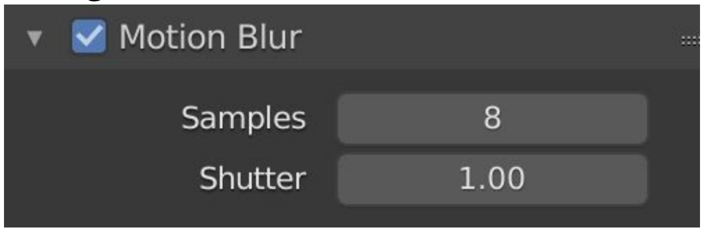
Chapter 6 - Camera Effects

6.1 Motion Blur

Motion blur in Eevee is done by post processing the image after rendering.
It is very basic, only blurs the render results based on pixel velocity, and does not support object/deformation motion blur. Only camera within the Camera View.



To enable motion blur, go to the Render Panel -> Motion Blur.



☑ Motion Blur

Samples 8

Shutter 1.00

Samples

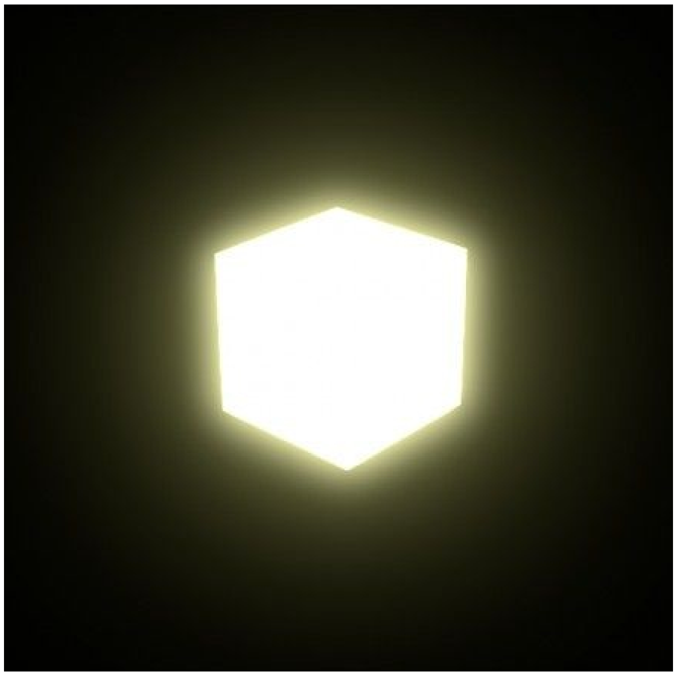
Number of samples to take for the motion blur effect.

Shutter

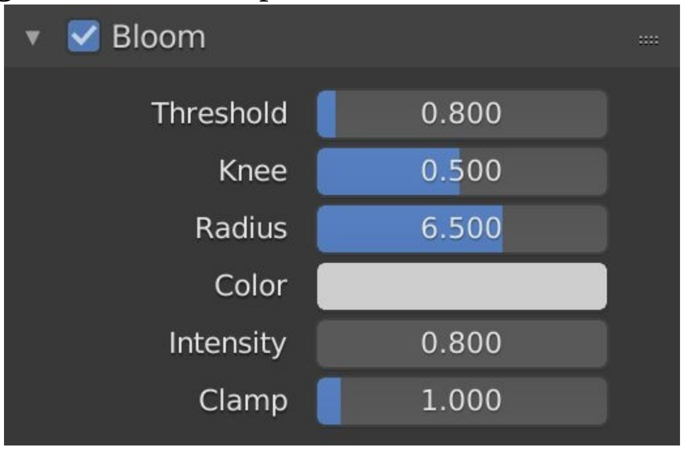
Time taken in frames between shutter open and close.

6.2 Bloom

A post-process effect that diffuses bright pixels. This simulate what happens inside a real camera.



It helps get a better sense of the pixel intensities
To enable bloom, go to the Render panel -> Bloom



Bloom ✓

Threshold	0.800
Knee	0.500
Radius	6.500
Color	
Intensity	0.800
Clamp	1.000

Threshold

Filters out pixels under this level of brightness.

Color

Color applied to the effect.

Knee

Makes transition between under/over-threshold gradual.

Radius

Spread distance.

Intensity

Blend factor.

Clamp

Maximum intensity a pixel can have.

6.3 Depth of Field

In Eevee, depth of field is done as a post-process This effect is using the same camera settings as Cycles, and only works in the Camera View.
To configure the camera Depth of Field, go to the Render panel -> Depth of Field.

Max Size

Max size of the bokeh shape for the depth of field (lower means faster).
For performance reason, the viewport will likely exhibit color artifacts when using large bokeh sizes, but these artifacts are not present in the final render.

Chapter 7 - Eevee Limitations

Eevee’s purpose is to be a real time render engine.
Many rendering features are not yet included and may be impossible to implement into Eevee’s architecture without compromising performance.
This chapter covers a near exhaustive list of the limitations imposed on the renderer when working with Eevee.

Cameras

- Only perspective and orthographic projections are currently supported.

Lights

- Shadows are not supported on light instances (dupli objects, group instancing).
- Only 128 active lights are supported by Eevee in a scene.
- Only 8 Shadowed sun lights can be supported at the same time.
- Lights can only have one color and do not support light node trees.

Light Probes

- Only supports up to 128 active Reflection Cubemaps.
- Only supports up to 64 active Irradiance Volumes.
- Only supports up to 16 active Reflection Planes inside the view frustum.

Indirect Lighting

- Eevee does not support *specular to diffuse* light bounces nor specular to specular light bounces. All specular lighting is turned off during baking.

Volumetrics

- Only single scattering is supported.
- Volumetrics are rendered only for the Camera *Rays*. They don't appear in reflections/refractions and probes.
- Volumetrics don't receive light from Irradiance Volumes but does receive world's diffuse lighting.
- Volumetric shadowing does only work on other volumetrics. They won't cast shadows on solid objects in the scene.
- Volumetric shadowing does only work for volumes inside the view frustum.
- Volumetric lighting does not respect the Lights shapes. They are treated as point lights.

Screen Space Reflections

- Only one glossy BSDF can emit screen space reflections.
- The chosen BSDF is currently arbitrary chosen.
- Screen Space Reflections will reflect transparent objects and objects using Screen Space Refraction but without accurate positioning due to the one layer depth buffer.

Screen Space Refraction

- Only one refraction event is correctly modeled.

Ambient Occlusion

- Objects are treated as infinitely thick, producing overshadowing if the Distance is really large.

Materials

Refraction

Refraction is faked by sampling the same reflection probe used by the Glossy BSDFs, but using the refracted view direction instead of the reflected view direction.
Only the first refraction event is modeled correctly. An approximation of the second refraction event can be used for relatively thin objects using Refraction Depth.

Bump

Bump mapping is supported using OpenGL derivatives which are the same for each block of 2x2 pixels.
This means the bump output value will appear pixelated. It is recommended to use normal mapping instead.
Tip: If you absolutely need to render using Bump nodes, render at twice the target resolution and downscale the final output.

Volumes Objects

Object volume shaders will affect the whole bounding box of the object.
The shape of the volume must be adjusted using procedural texturing inside the shader.

Shader Nodes

- All BSDF are using approximations to achieve real time performance so there will always be small differences between Cycles and Eevee.
- Some utility nodes, such as the Sky Texture Node, are not yet compatible with Eevee.

Memory Management

Eevee uses OpenGL, GPU Memory management is done by the OpenGL driver.
In theory, only the needed textures and meshes (now referred as “the resources”) for a single draw call (i.e. one object) needs to fit into the GPU memory.
So if the scene is really heavy, the driver will swap things in and out to make sure all objects are rendered correctly.
In practice, using too much GPU memory can make the GPU driver crash, freeze, or kill the application. So be careful of your scene load.
.
There is no standard way of estimating if the resources will fit into the GPU memory and or if the GPU will render them successfully.

CPU Rendering

Being an OpenGL engine, Eevee only uses the power of the GPU to render.
There is no plan to support CPU rendering as it would be very inefficient.
CPU power is still needed to handle high complexity scene as the geometry is still being precomputed by the CPU before rendering each frame.

Multiple GPU Support

There is currently no support for multiple GPU system.

Headless Rendering

There is currently no support for using Eevee on headless systems (i.e. without a Display).