UNITY FOR GAMES

E-BOOK

THE DEFINITIVE GUIDE TO LIGHTING IN THE

HIGH DEFINITION RENDER PIPELINE (HDRP)

2020 LTS EDITION

Unity®

Contents

Introduction

It all begins with light.

Unity built the High Definition Render Pipeline (HDRP) to help creators like you realize your vision, tapping the power of high-end PC or console hardware to reach new levels of graphical realism.

This guide was created to guide existing Unity artists and developers in exploring HDRP's physically based lighting techniques. HDRP represents a technological advance in Unity's real-time rendering, allowing you to work with light just as it behaves in the real world.

Do you want to render a lush rainforest landscape with natural sunlight? Or is the concrete jungle of the big city – brimming with neon emissive lighting – more your style? With HDRP, you can paint your scene levels with the brush of a cinematographer.

Build game environments that transport your players to places, anywhere from the familiar to fantastical.

Let's get started with HDRP and Unity.

Book of the Dead used HDRP to create its atmospheric lighting.

HDRP lighting overview

HDRP extends Unity's existing lighting system with a variety of features to make rendering your scene more closely resemble real-world lighting:

— **Physical light units and exposure**: HDRP uses real-world lighting intensities and units. Match the specs from known light sources, and set exposures using physical cameras.

— **Advanced lighting**: Take control over light placement with additional shape options for spot and area lights. Use Light Layers to limit the influence of lights onto specific meshes. Apply real-time effects like Screen Space Global Illumination and Screen Space Refraction.

— **Skyscapes**: Generate natural-looking skies with varied techniques. Use the Physically Based Sky system to simulate planetary atmosphere procedurally, or apply HDRIs as cubemap textures.

— **Fog**: Add depth and dimension to your scenes with fog. Enable volumetrics to integrate fog effects with your foreground objects and render cinematic shafts of light. Maintain per-light control of volumetric light and shadows and use the Local Volumetric Fog component for fine control of fog density with a 3D mask texture.

— **Volume system**: HDRP features an intuitive system that lets you block out different lighting effects and settings based on camera location or by priority. Layer and blend volumes to allow expert-level control over every square meter of your scene.

— **Post-processing**: HDRP post-processing is controlled by a series of Volume Overrides on top of the existing Volume system. Add anti-aliasing, tonemapping, color grading, bloom, depth of field, and a host of other effects.

— **Advanced shadows**: HDRP offers advanced artistic and performance control over shadows. Modify their tint, filtering, resolution, memory budget, and update modes. Accentuate small details and additional depth with contact shadows and micro shadows.

— **Advanced reflections**: Reflective surfaces can use several techniques to render. Reflection Probes offer a traditional reflection mapping approach,with Planar Reflection Probes giving you more advanced options for flat surfaces. Screen-space reflection (SSR) adds a real-time technique using the depth buffer.

Extensibility: HDRP is built on Unity's Scriptable Render Pipeline. Experienced technical artists and graphics programmers can extend the pipeline even beyond what's available out of the box.

Completely new to HDRP? Make sure you read Getting started with the High Definition Render Pipeline.

Installation

Unity 2020 LTS and above includes the HDRP package with the installation to ensure that you're always running on the latest verified graphics code. When you install the most recent Unity release, it also installs the corresponding version of HDRP.

HDRP package version	Compatible Unity version
12.x	2021.2
11.x	2021.1
10.x	2020.3 or Long Term Support (used in this guide)

Tying the HDRP graphics packages to a specific Unity release helps ensure compatibility. However, you can also switch to a custom version of HDRP by overriding the manifest file.

System requirements

HRDP is currently compatible with the following target platforms:

— Windows and Windows Store, with DirectX 11 or DirectX 12 and Shader Model 5.0

— Modern consoles (minimum Sony PlayStation 4 or Microsoft Xbox One)

— MacOS (minimum version 10.13) using Metal graphics

— Linux and Windows platforms with Vulkan

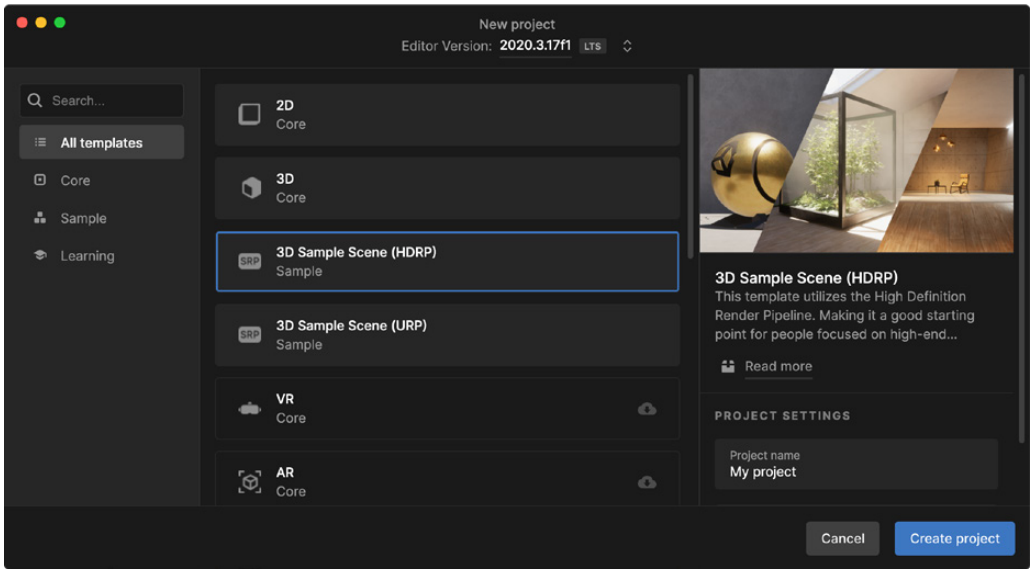
HDRP only works on console and desktop platforms that support compute shaders. HDRP does not support OpenGL or OpenGL ES devices. Refer to the documentation for more complete requirements and compatibility information.

See Virtual Reality in the High Definition Render Pipeline to learn about supported VR Platforms and devices.

Unity Hub

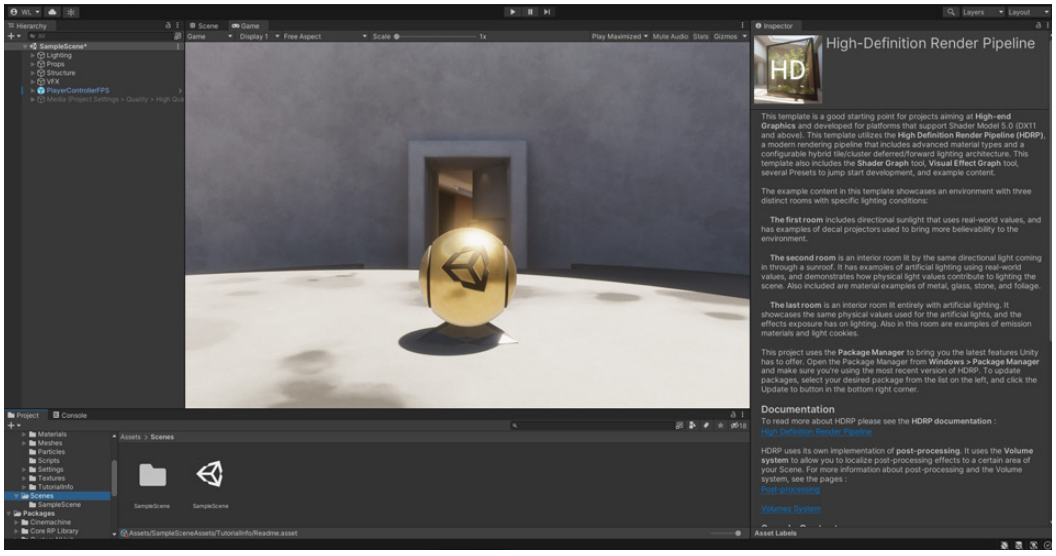
The Unity Hub is the simplest way to set up an HDRP project.

To get started, create a new Project. Select **3D Sample Scene (HDRP)** from the available templates (called High Definition RP in older versions of the Hub). This imports the HDRP package with some example presets.



Select the 3D Sample Scene (HDRP) template.

Load the **SampleScene**. You should see something like this:

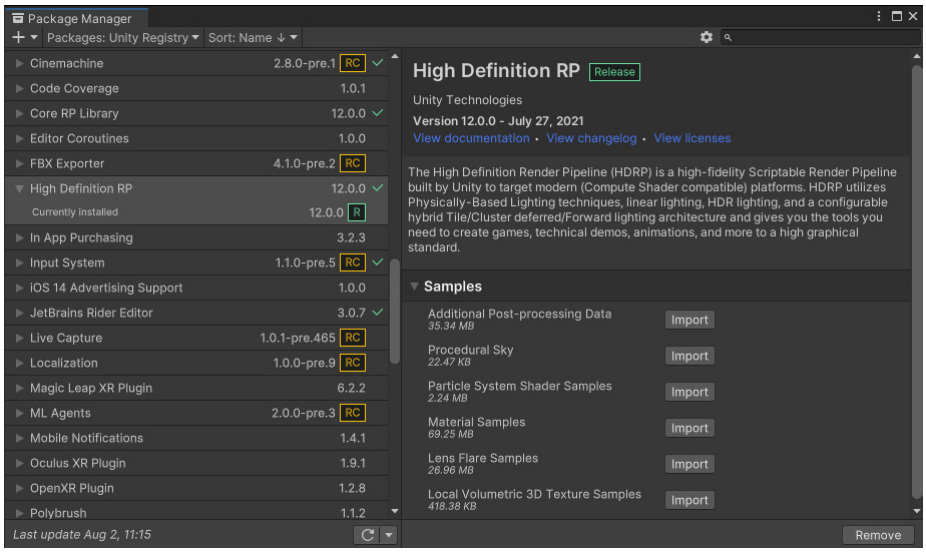


3D Sample Scene project setup

Package Manager installation

If you create your project with the **3D Core template**, Unity uses the older Built-In Render Pipeline. You can migrate the project to HDRP manually from the **Package Manager** (**Window > Package Manager**).

Find the High Definition RP package in the Unity Registry (or use the Search field to locate it), and install.

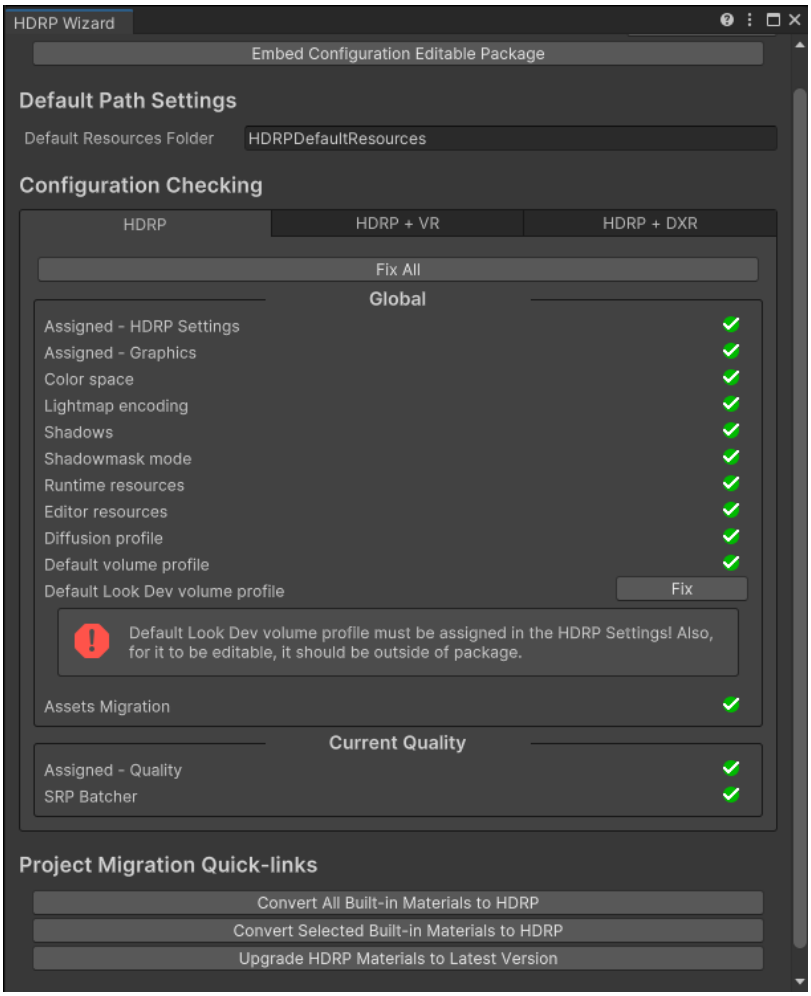


Installing the Package Manager

If there is a conflict with the current Project Settings, the **HDRP Render Pipeline Wizard** will appear to help you troubleshoot (also found under **Window > Render Pipeline > HDRP Render Pipeline Wizard**).[1]

Click **Fix All** under **Configuration Checking**, or click **Fix** for each issue to repair individually. This checklist can help you migrate from a non-SRP project.

[1] In Unity 2021.2/HDRP 12, this is located under **Window > Rendering > HDRP Wizard**.



The HDRP wizard

When the wizard finishes, a prompt will ask you to create a new HDRP Pipeline Asset. This is a file on disk that will hold your specific pipeline settings. Select **Create One** to add a new Render Pipeline Asset and assign the file here.

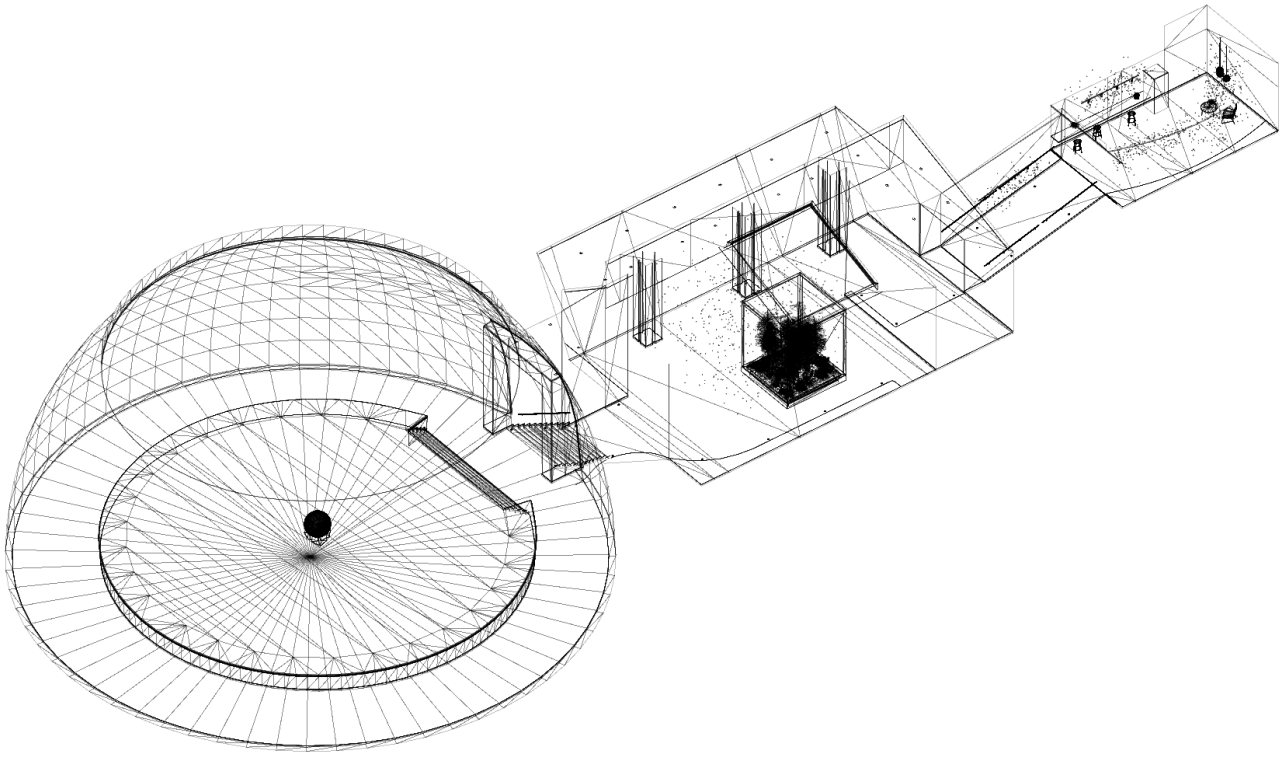
Once HDRP is functioning properly, all of the checkboxes in the Configuration Checking should be green, and the background environment may noticeably change color.

Note that manual installation from a blank project does *not* import the **3D Sample Scene (HDRP)**. Use the 3D Sample Scene template if you want access to the example assets shown in this guide.

HDRP Sample Scene

The 3D Sample Scene available from the Unity Hub is a template project that helps you get started with HDRP and physically based lighting. This lightweight project is less than 100 megabytes, and it offers a good working example of HDRP that you can always load quickly for reference.

We'll use this project to demonstrate many of the HDRP's features throughout this guide.



A wireframe representing the 3D Sample Scene environment

The small, multi-room environment demonstrates three distinct areas with different lighting setups. The directional light representing the sun has a real-world intensity of 100,000 lux, and each location corrects the camera's exposure to match the lighting environment.

Use the WASD keys and mouse to drive the FPS Controller around the level.

The 3D Sample Scene consists of three rooms.

— **Room 1** is a round platform lit by the overhead sunlight. Decals add grime and puddles of water to the concrete floor.

— **Room 2** adds volumetric shafts of light from the skylight, as well as advanced materials for the tree inside the glass case.

— **Room 3** showcases interior artificial lighting and emissive materials.

The Sample Scene is a lightweight project demonstrating HDRP features.

For a deeper look at the HDRP 3D Sample Scene, please check out this blog post from Unity technical artist Pierre Yves Donzallaz, which describes the template scene in more detail.

More HDRP sample content

You might find some other projects helpful once you're done exploring the HDRP 3D Sample Scene.

Though not originally intended for gaming, the **Auto Showroom** project demonstrates a highly detailed vehicle with realistic lighting. Change the stage lights, car paint materials, and backdrop in an interactive demo. This project is available via the Unity Hub.

The Auto Showroom template

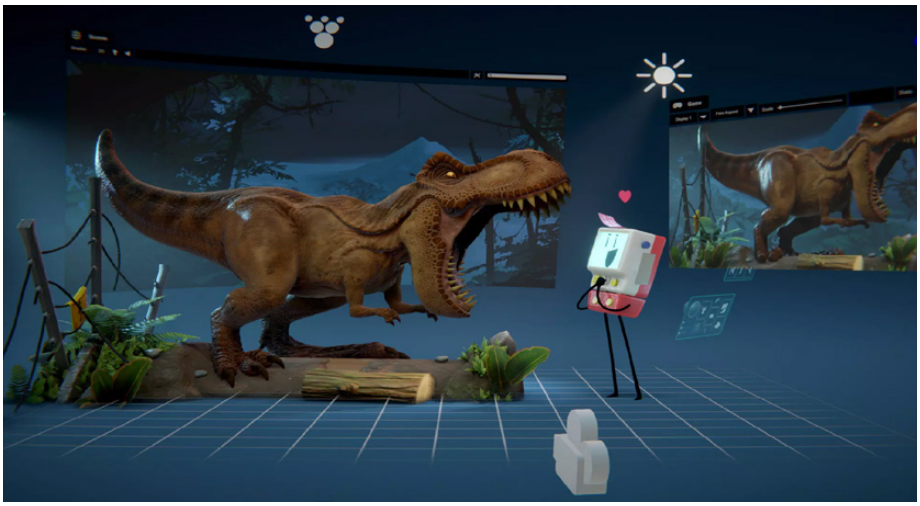
The **Spaceship Demo** primarily showcases the Visual Effect Graph, but it also features many HDRP features in a sci-fi environment. You can download it from Unity's GitHub repository.

The Spaceship Demo

If you're using HDRP with VR, you'll appreciate the **VR Alchemist Lab**. This project showcases interactive effects in a small medieval laboratory.

VR Alchemist Lab demo

To learn how to make cinematics or animated films, use our **Cinematic Studio Sample**, which showcases how to set up and light the shots of a funny short movie called ***Mich-L***, mixing stylized and photoreal rendering.



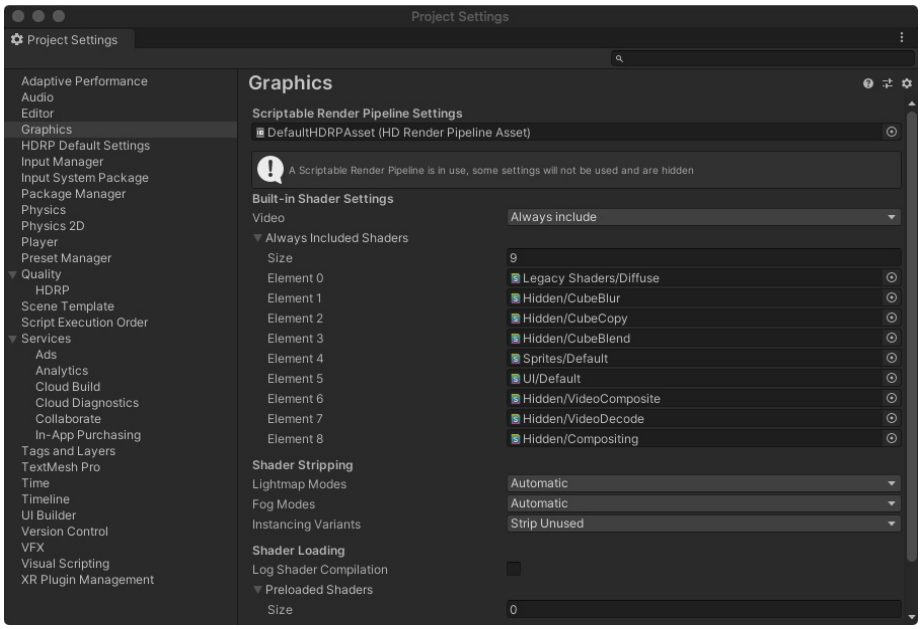
Cinematic Studio Sample

We invite you to explore these additional projects as you discover the High Definition Render Pipeline.

Project Settings

You'll find a few essential settings in the **Project Settings (Edit > Project Settings)** under **Graphics**, **HDRP Default Settings**, and **Quality**.

Note: **HDRP Default Settings** is called **HDRP Global Settings** in Unity 2021.2/ HDRP 12 and above.



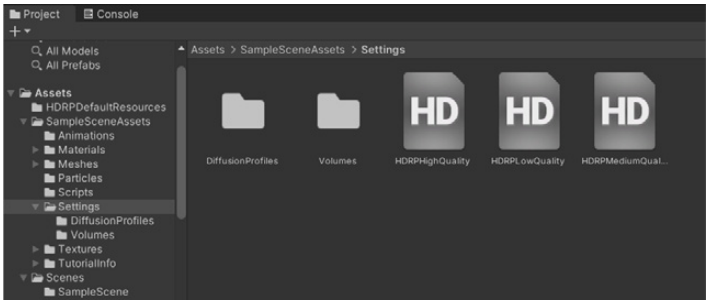
Project Settings

Graphics Settings

The top field, the **Scriptable Render Pipeline Settings**, represents a file on disk that stores all of your HDRP settings.

You can have multiple such **Pipeline Assets** per project. Think of each one as a separate configuration file. For example, you might use them to store specialized settings for different target platforms (Xbox, PlayStation, and so on), or they could also represent different visual quality levels that the player could swap at runtime.

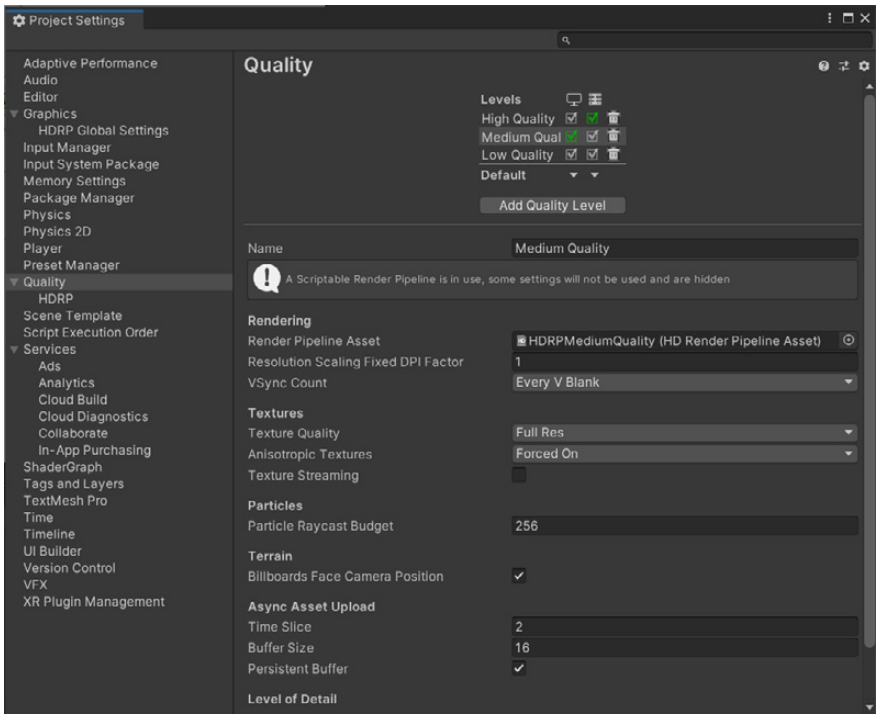
The **3D Sample Scene** begins with several Pipeline Assets in the Settings folder: **HDRPHighQuality**, **HDRPLowQuality**, and **HDRPMediumQuality**. There is also a **HDRPDefaultResources** folder containing a **DefaultHDRPAsset**.



The 3D Sample Scene includes low-, medium-, and high-quality Pipeline Assets.

Quality Settings

The Quality Settings allows you to correspond one of your Pipeline Assets with a predefined quality level. Select a **Level** at the top to activate a specific **Render Pipeline Asset**, shown in the **Rendering** options.

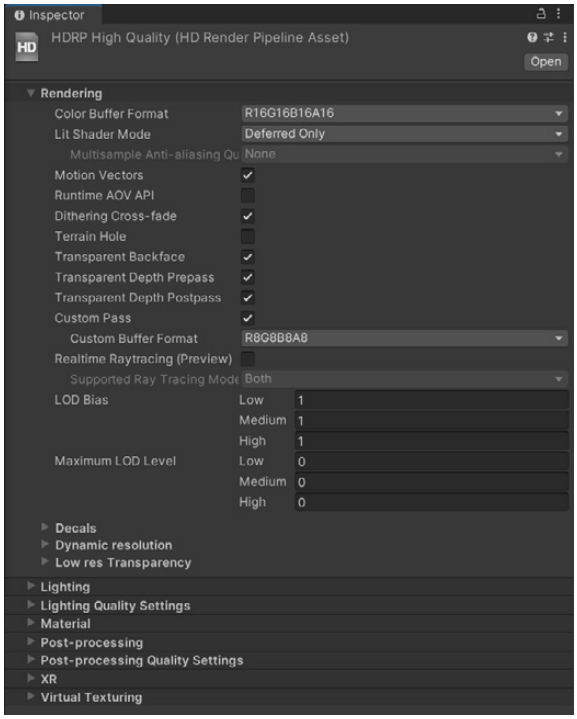


Select a quality level at the top to activate a Pipeline Asset.

You can customize the defaults or create additional Quality Levels, each paired with additional Pipeline Assets.

A Quality Level represents a specific set of visual features active in the pipeline. For example, you could create several graphics tiers within your application. At runtime, your players could then choose the active Quality Level, depending on hardware.

Edit the actual pipeline settings in the **Quality/HDRP** subsection. You can also select the Pipeline Asset in the Project view and edit the settings in the Inspector.



Editing the Pipeline Asset

Optimizing HDRP

Be aware that enabling more features in the Pipeline Asset will consume more resources. In general, optimize your project to use only what you need to achieve your intended effect. If you don't need a feature, you can turn it off to improve performance and save resources.

Here are some typical features that you can disable if you don't use them:

— In the **HDRP Asset**: Decals, low-res transparency, transparent backface/ depth prepass / depth postpass, SSAO, SSR, contact shadows, volumetrics, subsurface scattering, and distortions

— In the camera's **Frame Settings** (Main Camera, cameras used for integrated effects like reflections, or additional cameras used for custom effects): Refraction, Post-Process, After Post-Process, Transmission, Reflection Probe, Planar Reflection Probe, and Big Tile Prepass

HDRP Default Settings

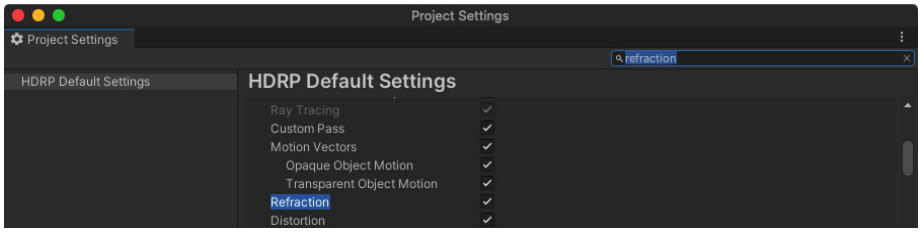
The HDRP Default Settings section (called **HDRP Global Settings** in Unity 2021.2/ HDRP 12) determines the baseline configuration of which features you have enabled when you start the project. You will override these with local settings based on camera position and/or priority (see Volumes below).

Global Settings save in their own separate Pipeline Asset defined at the top field. Set up the default rendering and post-processing options here.

Enabling HDRP features

As you develop your project, you may need to return to the HDRP Default Settings to toggle a specific feature on or off. **Some features will not render unless the corresponding checkbox in the Default Settings is explicitly enabled**. Be aware that certain settings appear in the **Volume Profiles** and some features appear in the **Frame Settings**, depending on usage.

While familiarizing yourself with HDRP's feature set, make use of the top right Search field in the Project Settings. This will only show you the relevant panels with the search terms highlighted.

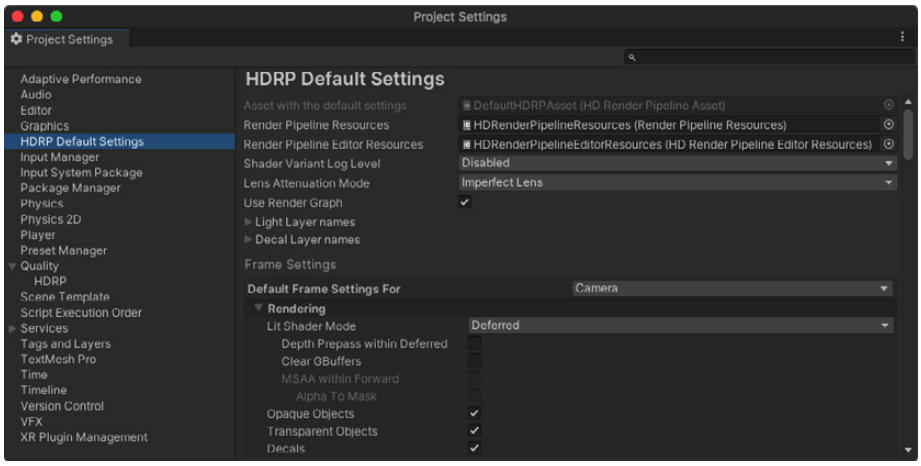


Search for HDRP features

Enabling a feature in the HDRP Default Settings does not guarantee that it is active. To verify, you must also locate the Pipeline Asset currently selected as the Quality level. Check the corresponding option there as well. **HDRP features must be enabled in both the global and local settings in order to function.**

Forward vs Deferred rendering

When configuring your HDRP settings in the Pipeline Asset, you will usually start with the **Lit Shader Mode** under **Rendering**. Here you can choose between **Deferred**, **Forward**, or **Both**. These represent the *rendering path*, a specific series of operations related to how the pipeline will render and light the geometry.



Modifying the default HDRP settings

Customizing the render path

Choose **Forward** or **Deferred** in the **Lit Shader Mode**to set your default rendering path.

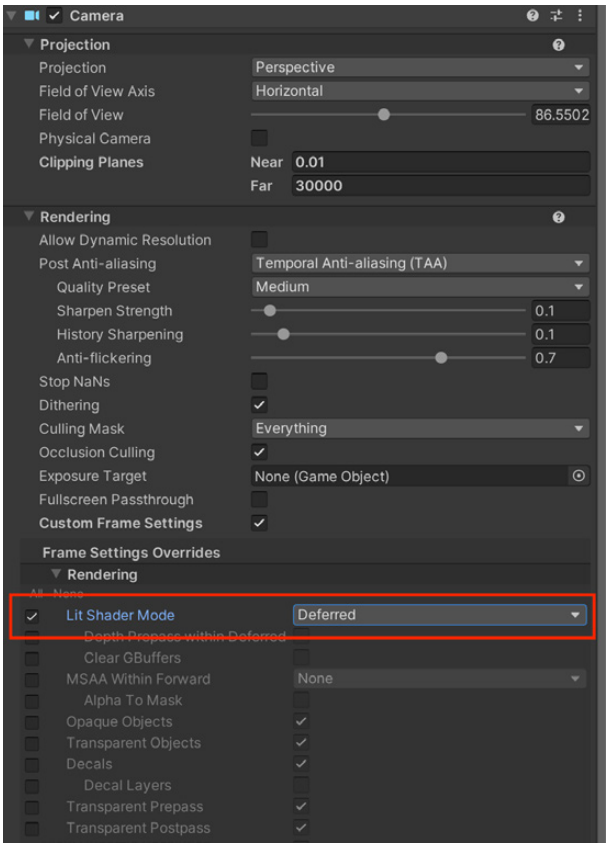
HDRP is flexible and also allows you to choose **Both**. This option lets you use one render path for most rendering and then override it per camera. However, this approach uses more GPU memory. In most cases, it is better to choose either Forward or Deferred.

— To affect all cameras by default, go to **HDRP Default Settings** and locate **Default Frame Settings**. This can apply for a **Camera**, **Baked** or **Custom Reflection**, or **Realtime Reflections**.

  In the **Rendering** group, set the render path in the **Lit Shader Mode**.

— For a specific camera, check its **Custom Frame Settings** to override it.

  Then, in the **Rendering** group, override and change the rendering path of the **Lit Shader Mode**.



Modifying the custom frame settings for a camera

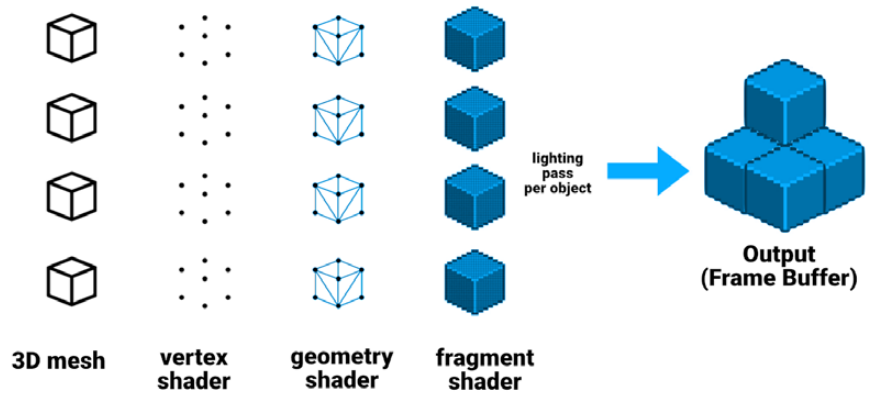
More about rendering paths

You may want to understand how these rendering paths work to see how Lit Shader Mode will impact the other settings in our pipeline.

Forward rendering

In Forward rendering, the graphics card splits the on-screen geometry into vertices. Those vertices are further broken down into fragments, or pixels, which render to screen to create the final image.

Each object passes, one at a time, to the graphics API. Forward rendering comes with a cost for each light. The more lights in your Scene, the longer rendering will take.



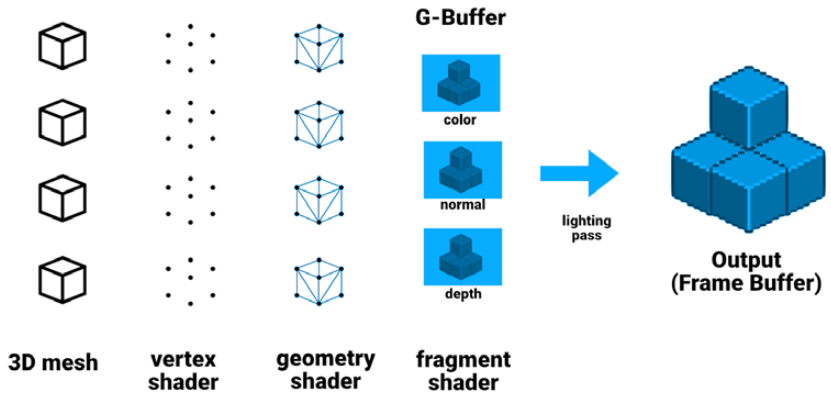
Forward rendering path

Forward rendering draws lights in separate passes. If you have multiple lights hitting the same GameObject, this can create significant *overdraw*, slowing down when a lot of lights and objects are present.

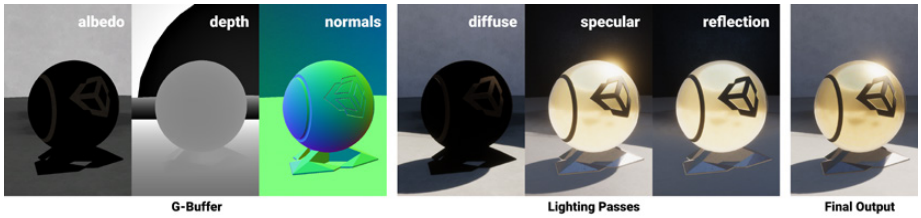
Unlike traditional forward rendering, HDRP does add some efficiencies to the forward renderer. For example, it culls and renders several lights together in a single pass per object material. However, it's still a relatively expensive process. If performance is an issue, you may want to use Deferred Shading instead.

Deferred shading

HDRP can also use deferred shading, where lighting is not calculated per object. Instead deferred shading postpones heavy rendering to a later stage and uses two passes.



Deferred shading path



Deferred shading applies lighting to a buffer instead of each object. Each of these passes contributes to the final rendered image.

In the first pass, or the G-buffer geometry pass, Unity renders the GameObjects. This pass retrieves several types of geometric properties and stores them in a set of textures (e.g., diffuse and specular colors, surface smoothness, occlusion, normals, and so on).

In the second pass, or lighting pass, Unity renders the Scene's lighting after the G-buffer is complete. Hence, it *defers* the shading. The deferred shading path iterates over each pixel and calculates the lighting information based on the buffer instead of the individual objects.

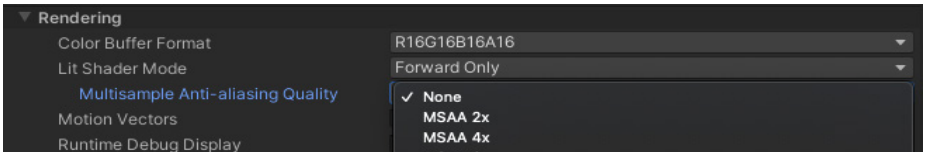
For more information about the technical differences between the rendering paths, see Forward and Deferred rendering in the HDRP documentation.

Anti-aliasing

The rendering path in the Lit Shader Mode influences how you can use anti-aliasing to remove the jagged edges from your renders. HDRP offers several anti-aliasing techniques, depending on your production needs.

Multisample anti-aliasing (MSAA)

Multisample anti-aliasing (MSAA) is a popular anti-aliasing method among PC gamers. This is a high-quality hardware method that smooths the edges of individual polygons, and it only works with forward rendering in Unity. Most modern GPUs support 2x, 4x, and 8x MSAA samples.

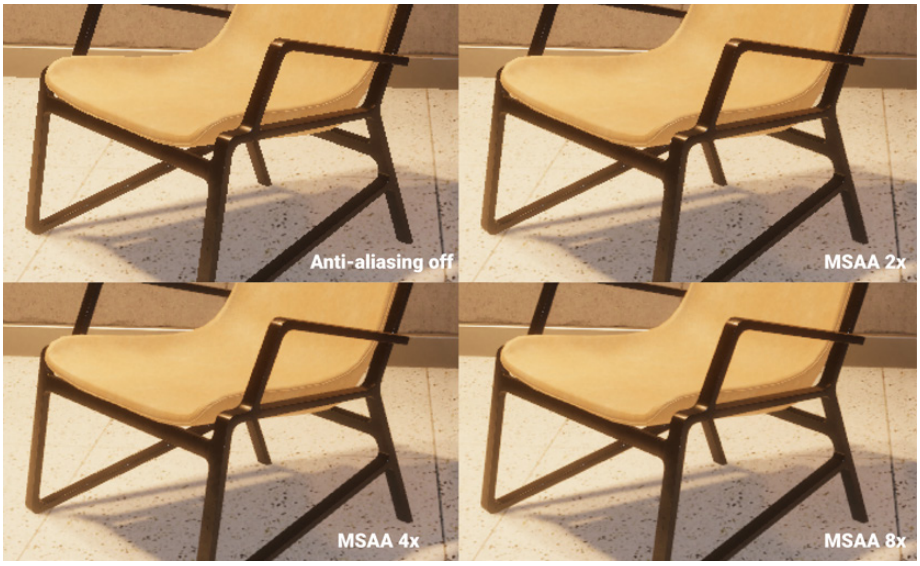


MSAA quality settings

In your active Pipeline Asset, set the Lit Shader Mode to **Forward Only**. Next select **MSAA 2x**, **MSAA 4x**, or **MSAA 8x** for the **Multisample Anti-aliasing Quality**. Higher values result in better anti-aliasing, but they are slower.

We can see this more clearly when we zoom into the camera view.

Original scene

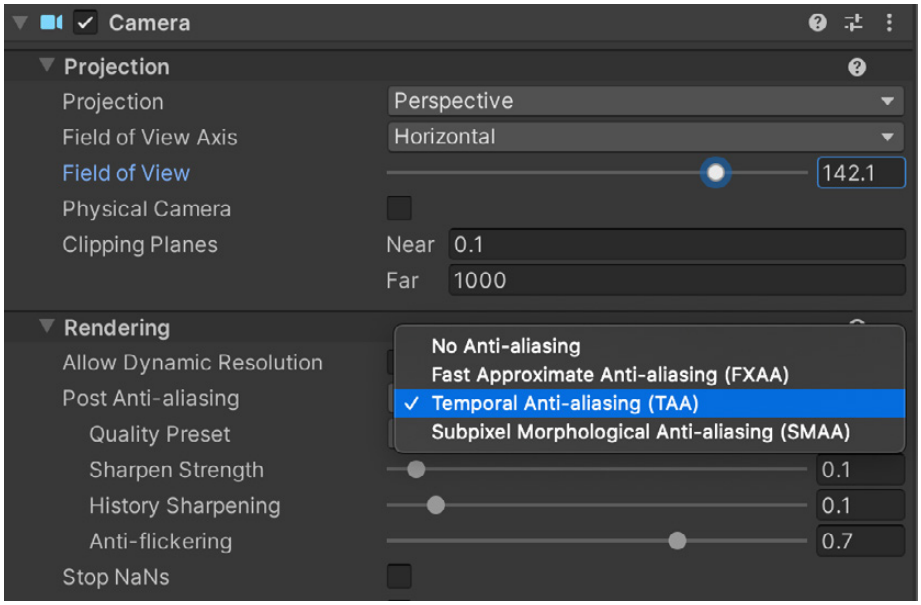


MSAA settings applied to an image

Note these limitations:

— MSAA is incompatible with deferred shading's G-buffers, which store the scene geometry in a texture. Thus, deferred shading requires one of the Post-processing Anti-aliasing techniques (below).

— Because MSAA only deals with polygon edge aliasing, it cannot prevent aliasing found on certain textures and materials hit by sharp specular lighting. You may need to combine MSAA with another Post-processing Anti-aliasing technique (right) if that is an issue.



Adjust Post Anti-aliasing on your camera when using deferred shading.

Post-processing Anti-aliasing

Your camera also allows you to apply anti-aliasing as a post-processing technique with the Post Anti-aliasing setting:

— **Temporal Anti-aliasing (TAA)** combines information from past frames and the current frame to remove jaggies in the current frame. You must enable motion vectors in order for it to work. TAA generally produces great results, but it may create ghosting artifacts in some situations (e.g., a GameObject moving quickly in front of a contrasting surface). HDRP10 introduced improvements to cut down on typical TAA artifacts. Unity's implementation reduces ghosting, improves sharpness, and prevents flickering found in other solutions.

— **Fast Approximate Anti-aliasing (FXAA)** is a screen-space anti-aliasing algorithm that blends pixels between regions of high contrast. It is a relatively fast technique that does not require extensive computing power, but it can reduce the overall sharpness of the image.

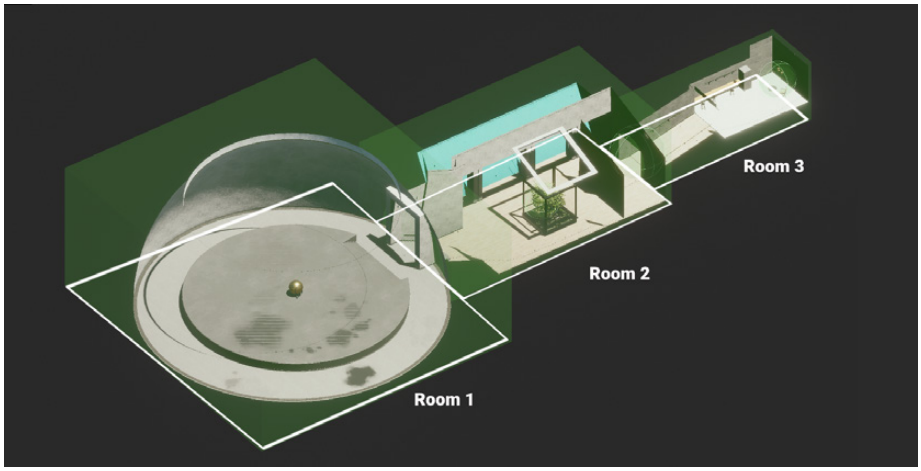
— **Subpixel Morphological Anti-aliasing (SMAA)** detects borders in the image, then looks for specific patterns to blend. This produces sharper results than FXAA, and it works well with flat, cartoon-like, or clean art styles.

Post-processing anti-aliasing – compare the results of FXAA, SMAA, and TAA settings.

Note: When combining Post-processing Anti-aliasing with Multisample Anti-aliasing, be aware of the rendering cost. As always, optimize your project to balance visual quality with performance.

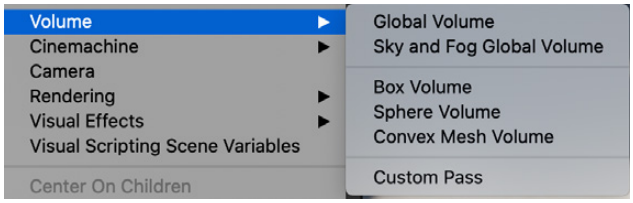
Volumes

HDRP uses a **Volume framework**. This system allows you to split up your Scene and enable certain settings or features based on camera position. For example, the HDRP template level contains three distinct parts, each with its own lighting setup. Thus, we have different Volumes encompassing each room.



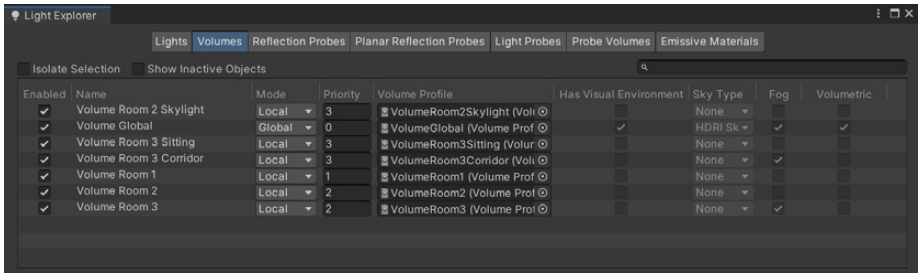
Volumes cover spaces with distinct lighting conditions.

A Volume is just a placeholder object with a Volume component. You can create one through the **GameObject > Volume** menu by selecting a preset. Otherwise, simply make a GameObject with the correct components manually.



Creating a Volume object using the presets

Because Volume components can be added to any GameObject, it can be difficult to find them via the Hierarchy. The Light Explorer **(Window > Rendering > Light Explorer > Volumes)** can help you locate the volumes in the loaded Scenes. Use this interface to make quick adjustments.

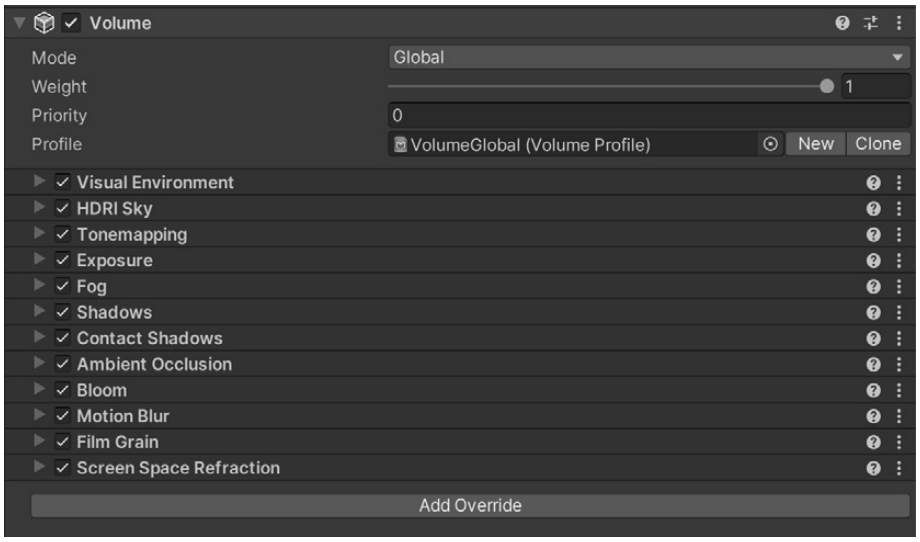


The Light Explorer can list all the Volumes in the open Scene(s).

Local and Global

Set the Volume component's **Mode** setting to either **Global** or **Local**, depending on context.

A global Volume works as a "catch-all" without any boundaries, and it affects all cameras in the Scene. In the HDRP template scene, the VolumeGlobal defines an overall baseline of HDRP settings for the entire level.

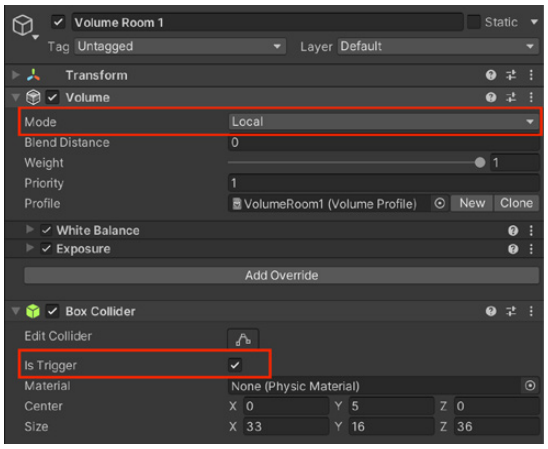


The Global Volume overrides

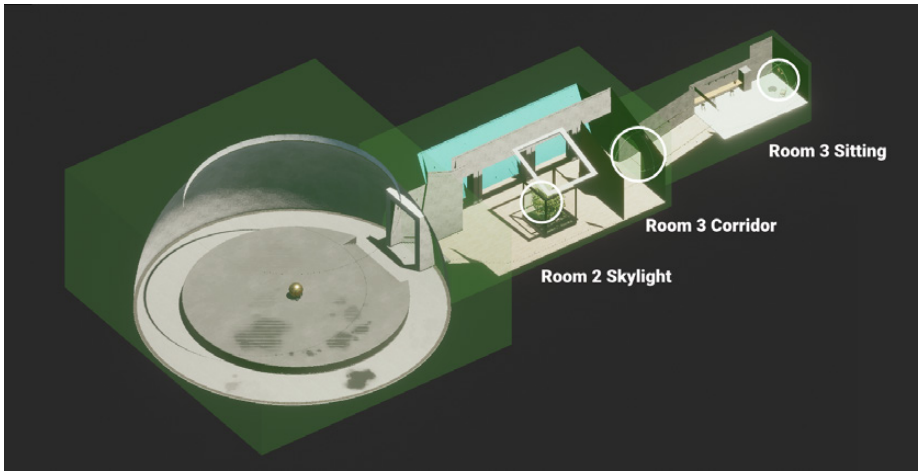
A local Volume defines a limited space where its settings take effect. It uses a Collider component to determine its boundaries. Enable **IsTrigger** if you don't want the Collider to impede the movement of any physics bodies like your FPS player controller.

In the template scene, each room has a local Volume with a BoxCollider that overrides the global settings.

Room 2 has a small, spherical Volume for the bright center next to the glass case. Likewise, Room 3 has smaller Volumes at its entrance corridor and at the seated area below the pendant lights.



Each room has a local Volume with a Collider set to IsTrigger.



Use smaller Volumes for special lighting conditions.

In the SampleScene, the local Volumes override White Balance, Exposure, and/or Fog. Anything not explicitly overridden falls back to the global defaults.

As your camera moves around the scene, the global settings take effect until your player controller bumps into a local Volume where those settings take over.

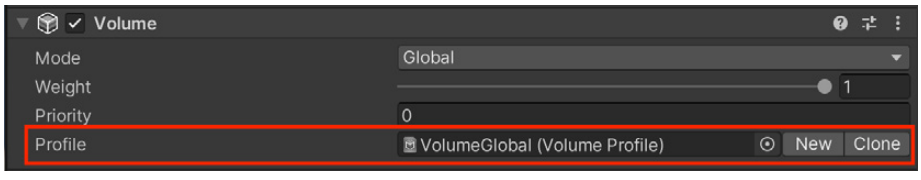
Performance tip

Don't use a large number of Volumes. Evaluating each Volume (blending, spatialization, override computation, and so on) comes with some CPU cost.

Volume Profiles

A Volume component itself contains no actual data. Instead, it references a **Volume Profile**, a ScriptableObject Asset on disk that contains HDRP settings to render the scene. Use the Profile field to create a new Volume Profile with the **New** or **Clone** buttons.

You can also switch to another Profile you already have saved. Having the Volume Profile as a file makes it easier to reuse previous settings and share Profiles between your Volumes.



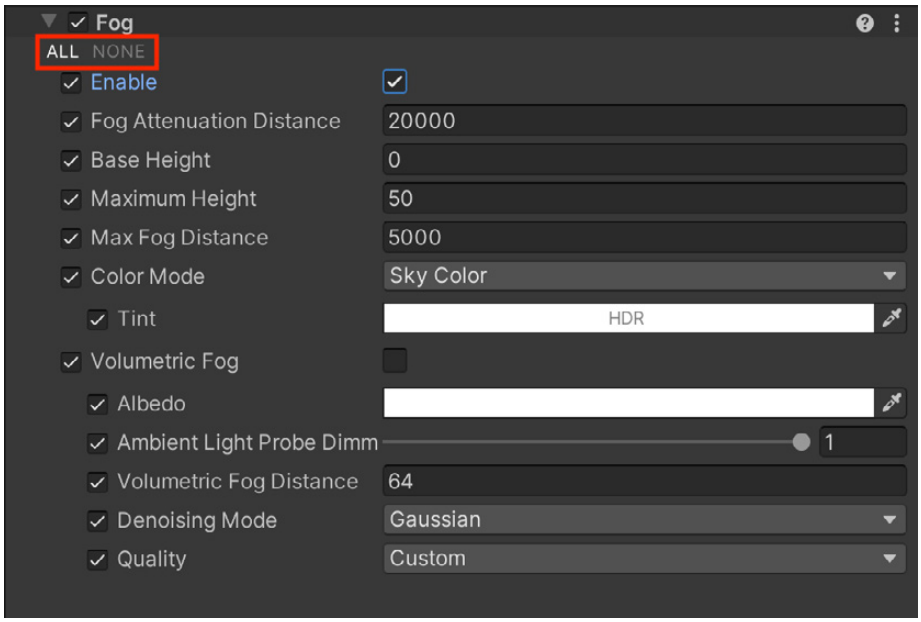
Use the Profile field to switch Volume Profiles or create a new one.

Note that changes done to Volume Profiles in Play mode will not be lost when leaving said mode.

Volume Overrides

Each Volume Profile begins with a set of default properties. To edit their values, use Volume Overrides and customize the individual settings. For example, Volumes Overrides could modify the Volume's Fog, Post-processing, or Exposure.

Once you have your **Volume Profile** set, click **Add Override** to customize the Profile settings. A Fog override could look like this:



An example of Fog as a Volume Override

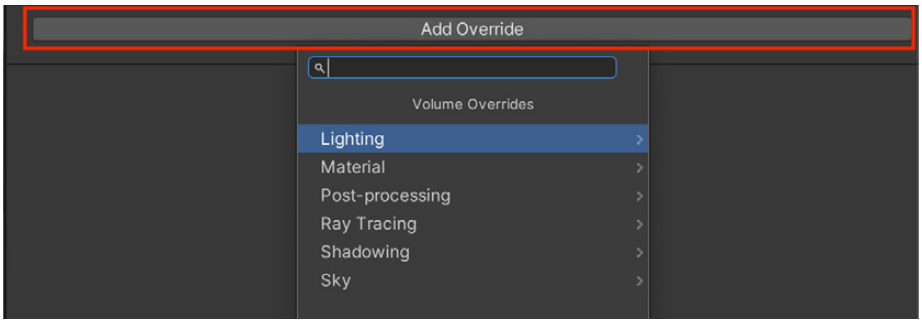
Each of the Volume Override's properties has a checkbox at the left, which you can enable to edit that property. Leaving the box disabled means HDRP uses the Volume's default value.

Each Volume object can have several overrides. Within each one, edit as many properties as needed. You can quickly check or uncheck all of them with the **All** or **None** shortcut at the top left.

Overrides workflow

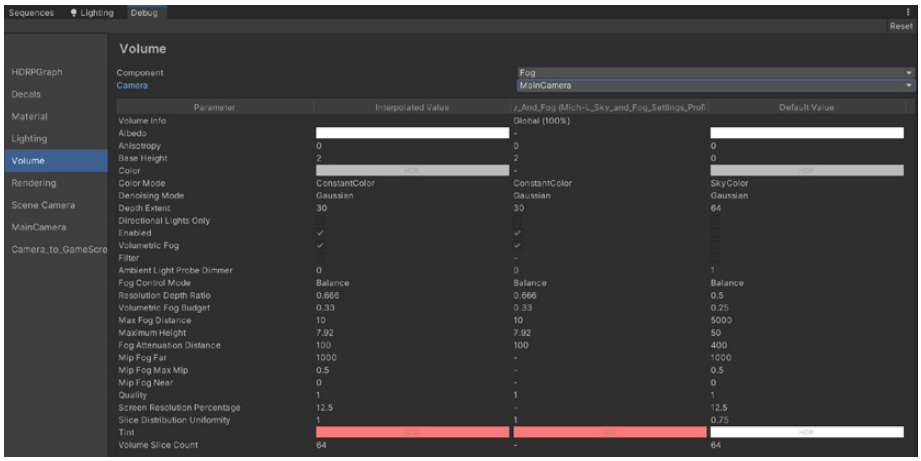
Adding overrides is a key workflow in HDRP. If you understand the concept of inheritance from programming, Volume Overrides will seem familiar to you.

The higher-level Volume settings serve as the defaults for lower-level Volumes. Here, the HDRP Default Settings pass down to the global Volume. This, in turn, serves as the “base” for the local Volumes.



Adding HDRP features using Volume Overrides

The Global Volume overrides the HDRP Default Settings. The Local Volumes, in turn, override the Global Volume. Use the **Priority**, **Weight**, and **Blend Distance** (outlined below) to resolve any conflicts from overlapping Volumes.



Debugging a Volume

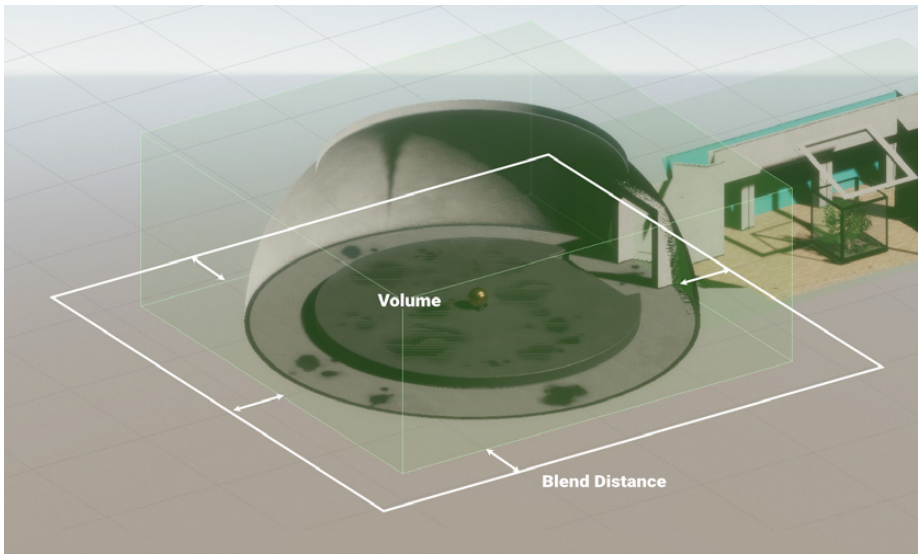
To debug the current values of a given Volume component, you can use the Volume tab in the Rendering Debugger.

You can find a complete Volume Overrides List in the HDRP documentation.

Blending and priority

Because you often need more than one Volume per level, HDRP allows you to blend between Volumes. This makes transitions between them less abrupt.

At runtime, HDRP uses the camera position to determine which Volumes affect the final HDRP settings.

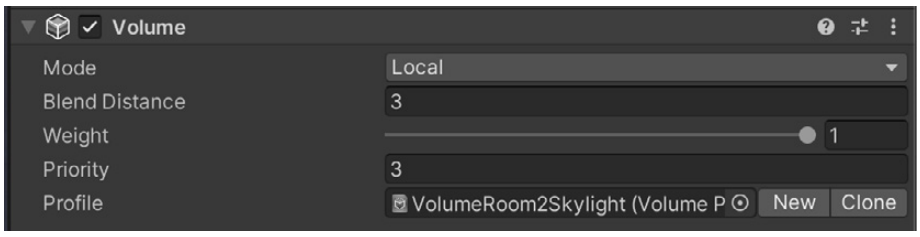


Blend Distance defines a transition zone around the Volume.

Blend Distance determines how far outside the Volume's Collider to begin fading on or off. A value of 0 for Blend Distance means an instant transition, while a positive value means the Volume Overrides begin blending once the camera enters the specified range.

The Volume framework is flexible and allows you to mix and match Volumes and overrides as you see fit. If more than one Volume overlaps the same space, HDRP relies on **Priority** to decide which Volume takes precedence. Higher values mean higher priority.

In general, set your Priority values explicitly to eliminate any guesswork. Otherwise, the system will use creation order as the Priority "tiebreaker," which may lead to unexpected results.



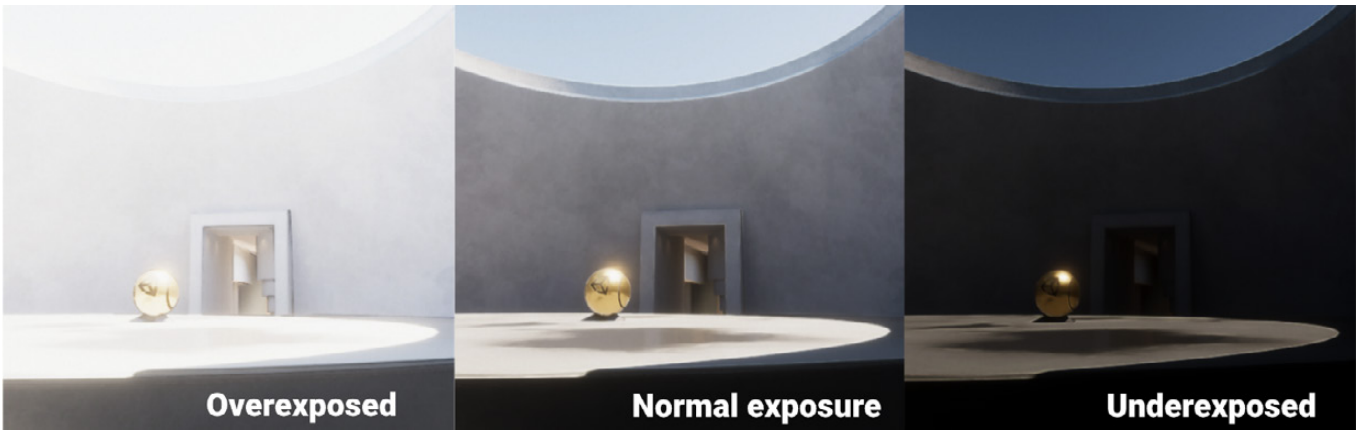
Use Blend Distance, Weight, and Priority when overlapping local Volumes.

Exposure

HDRP uses real-world lighting models to render each scene. As such, many properties are analogous to their counterparts in traditional photography.

Understanding exposure value

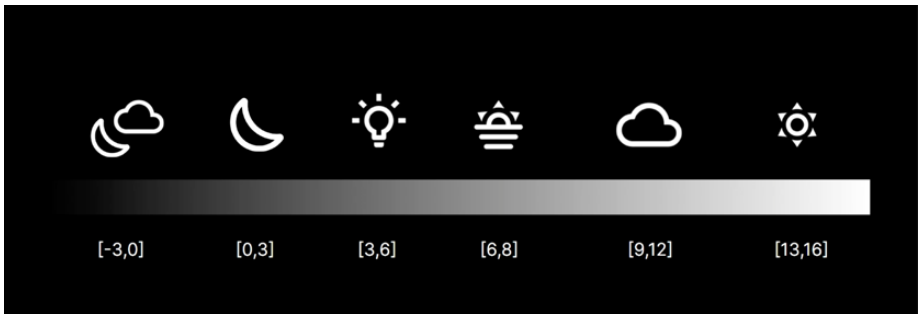
Exposure value (EV) is a numeric value that represents a combination of a camera's shutter speed and f-number (which determines the size of the lens opening, or aperture). You need to properly set exposure to reach ideal brightness, capturing high levels of detail in both the shadows and highlights. Otherwise, overexposing or underexposing the image leads to less-than-desirable results.



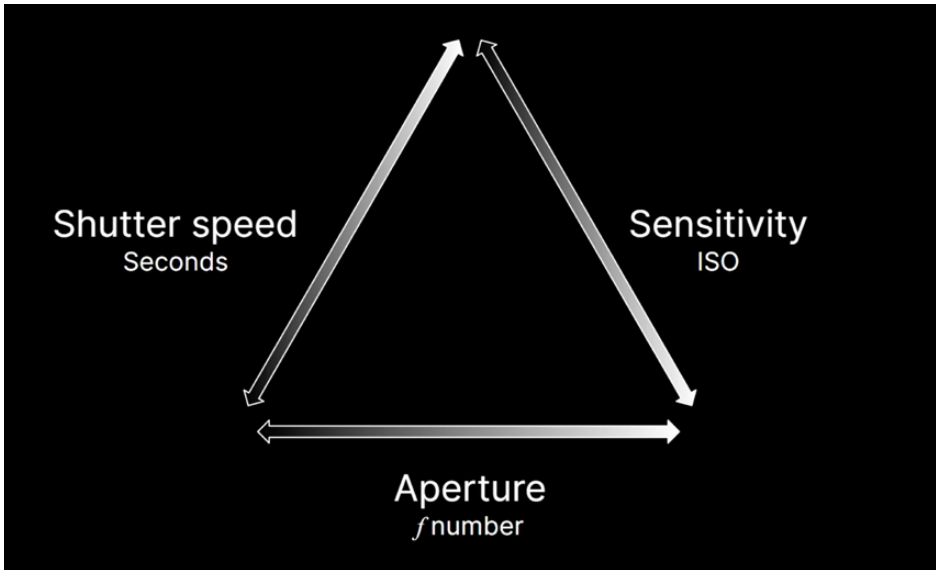
Compare overexposed, underexposed, and balanced images.

Your exposure range in HDRP will typically fall somewhere along this spectrum:

Greater exposure values allow less light into the camera and are appropriate for



Exposure range, from a moonless night to a bright, sunny day



Exposure triangle

more brightly lit situations. Here, an EV value between 13 and 16 is suitable for a sunny daytime exterior. In contrast, a dark, moonless, night sky might use an EV between -3 and 0.

You can vary a number of factors in an actual camera's settings to modify your Exposure value:

— The shutter speed, the length of time the image sensor is exposed to light

— The f-number, or the size of the aperture/lens opening

— The ISO, or sensitivity of the film/sensor

Photographers call this the *exposure triangle*. In Unity, as with a real camera, you can arrive at the same exposure value using different combinations of these numbers.

HDRP expresses all exposure values in **EV_{100}**, which fixes the sensitivity to that of 100 International Standards Organisation (ISO) film.

Exposure value formula

This formula actually calculates exposure value.

$$EV = \log_2 \left(\frac{f\,\text{number}^2 / \text{shutter speed}}{ISO / 100} \right)$$

Exposure formula

It's a logarithmic base-2 scale. As the exposure value increases 1 unit, the amount of light entering the lens decreases by half.

HDRP allows you to match the exposure of a real image. Simply shoot a digital photo with a camera or smartphone. Grab the metadata from the image to identify the f-number, shutter speed, and ISO.

Use the digital photo's Exif data to match exposure.

Then, calculate the exposure value using the formula above. If you use the same value in the Exposure override (see below), the rendered image should fall in line with the real-world image exposure.

In this way, you can use digital photos as references when lighting your level. While your goal isn't necessarily to recreate the image perfectly, matching an actual photograph can take the guesswork out of your lighting setups.

Exposure override

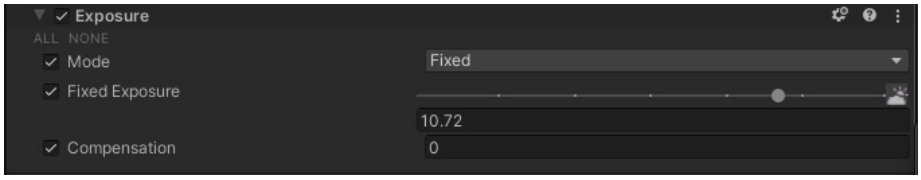
In HDRP, Exposure is a Volume Override. Add it to a local or global Volume to see the available properties.

In the **Mode** dropdown, you can select one of the following: **Fixed**, **Automatic**, **Automatic Histogram**, **Curve Mapping**, and **Physical Camera**.

Compensation allows you to shift or adjust the exposure. You would typically use this to apply minor adjustments and "stop" the rendered image up and down slightly.

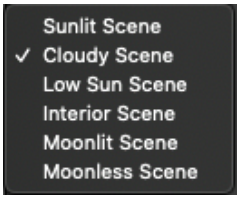
Fixed mode

Fixed mode lets you set the exposure value manually.



Fixed mode exposure

Follow the graduation marks on the **Fixed Exposure** slider for hints. The icon to the right also has a dropdown of Presets (e.g., 13 for a Sunlit Scene down to -2.5 for a Moonless Scene). You can also set the field directly to any value.



Fixed exposure presets

Fixed mode is simple but not very flexible. It usually only works if you have a Volume or Scene with relatively uniform lighting, where one exposure value can work throughout.

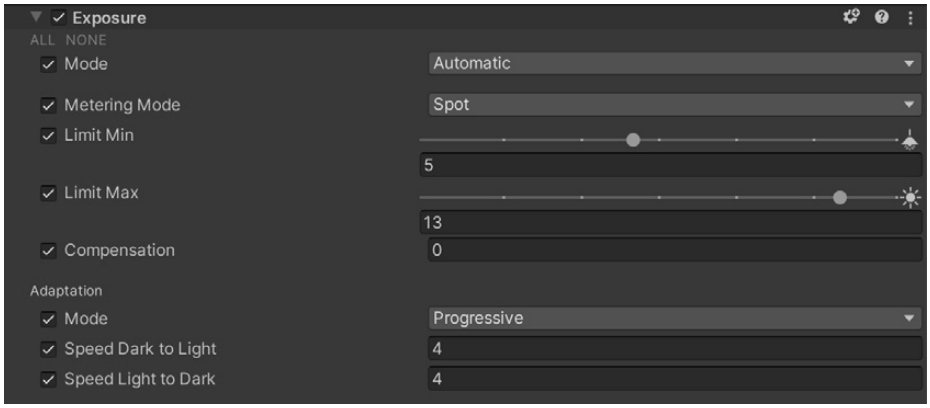
Automatic mode

Automatic mode dynamically sets the exposure depending on the range of brightness levels onscreen. This functions much like the human eye adapts to varying levels of darkness, redefining what is perceived as black.

While Automatic mode will work under many lighting situations, it can also unintentionally overexpose or underexpose the image when pointing the camera at a very dark or very bright part of the scene.

Use the **Limit Min** and **Limit Max** to keep the exposure level within a desirable range. Playtest to verify that your limits stay within your expected exposure throughout the level.

Metering Mode, combined with mask options, determines what part of the frame to use for autoexposure.



Automatic mode exposure

The **Adaptation mode** controls how the autoexposure changes as the camera transitions between darkness and light, with options to adjust the speed. Just like with the eye, moving the camera from a very dark to a very light area, or vice versa, can be briefly disorienting.

Metering mode options

Automatic, Automatic Histogram, and Curve Mapping modes use Metering mode to control what part of the frame to use when calculating exposure. You can set the Metering mode to:

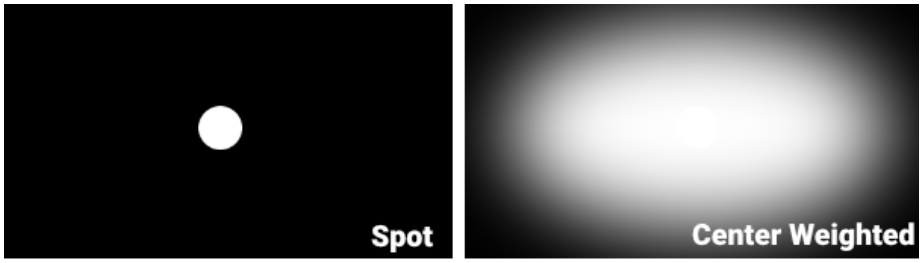
— **Average**: The camera uses the entire frame to measure exposure.

— **Spot**: The camera only uses the center of the screen to measure exposure.

— **Center Weighted**: The camera favors pixels in the center of the image and feathers out toward the edges of frame.

— **Mask Weighted**: A supplied image (Weight Texture Mask) determines which pixels are most important when determining exposure.

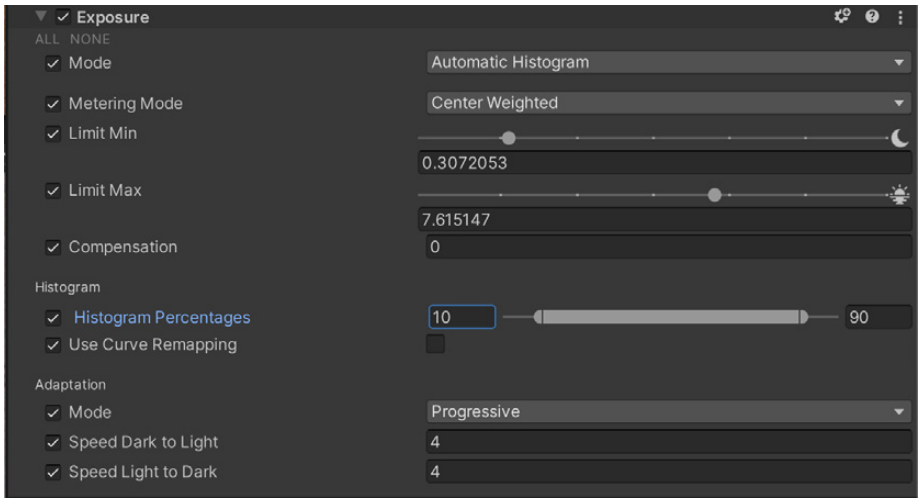
— **Procedural Mask**: The camera evaluates exposure based on a procedurally generated texture. You can change options for the center, radius, and softness.



Spot and Center Weighted metering modes

Automatic Histogram

Automatic Histogram mode takes Automatic mode a step further. This computes a histogram for the image and ignores the darkest and lightest pixels when setting exposure.

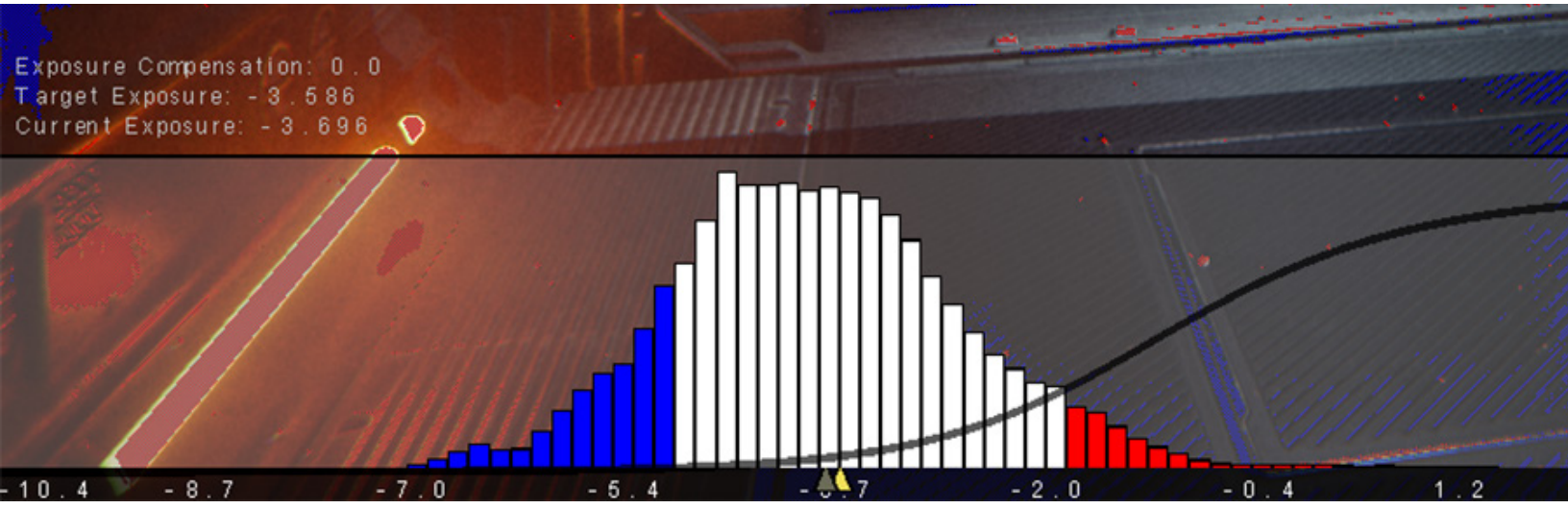


Automatic Histogram mode

By rejecting very dark or very bright pixels from the exposure calculation, you may experience a more stable exposure whenever extremely bright or dark pixels appear on the frame. This way, intense emissive surfaces or black materials won't underexpose or overexpose your rendered output as severely.

The **Histogram Percentages** setting allows you to discard anything in the histogram outside the given range of percentages (imagine clipping the brightest and darkest pixels from the histogram's leftmost and rightmost parts).

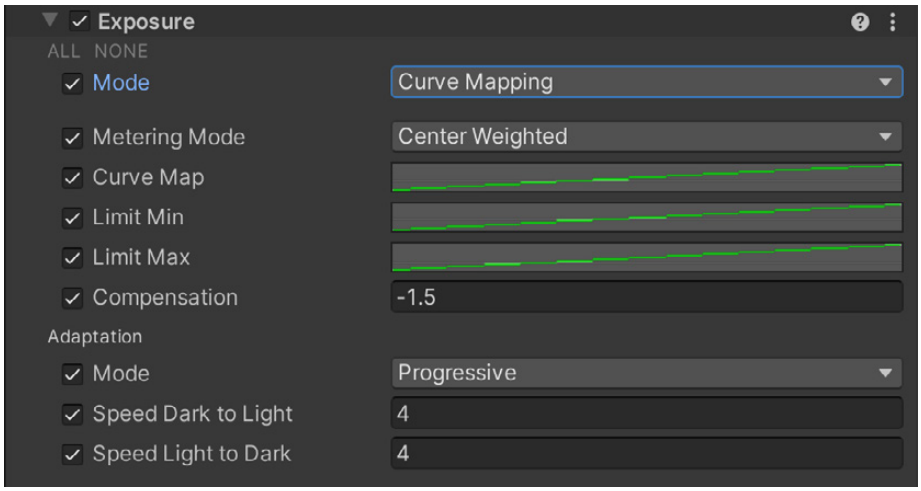
Curve Remapping lets you remap the exposure curve as well (see Curve Mapping below).



Automatic Histogram mode uses pixels from the middle of the histogram to calculate exposure.

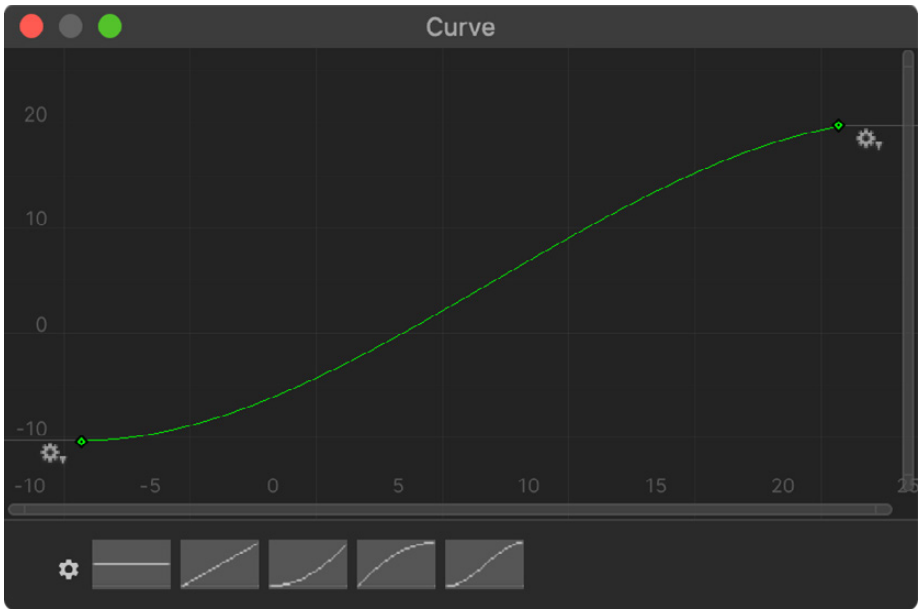
Curve Mapping

The Curve Mapping mode is another variant of Automatic mode.



Curve Mapping mode

Here, the x-axis of the curve represents the current exposure, and the y-axis represents the target exposure. Remapping the exposure curve can generate very precise results.

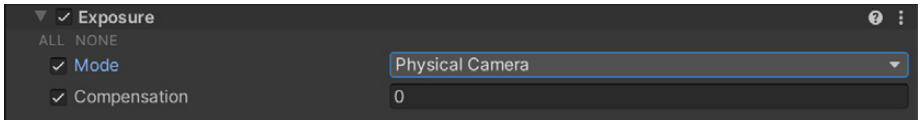


Adjust the exposure using a curve.

Physical Camera

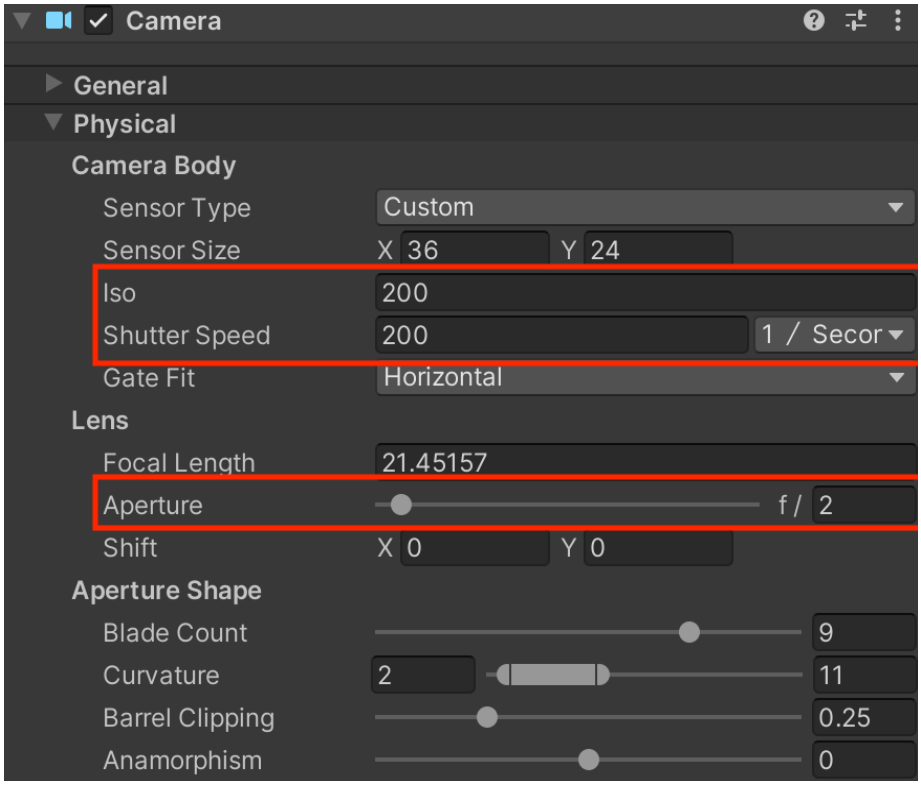
Those familiar with photography may find Physical Camera mode helpful for setting camera parameters.

Switch the Exposure override's **Mode** to **Physical Camera**, then locate the Main Camera.



Physical Camera mode

Enable **Physical Camera**. The Inspector shows the following properties.



Physical Camera properties on a camera

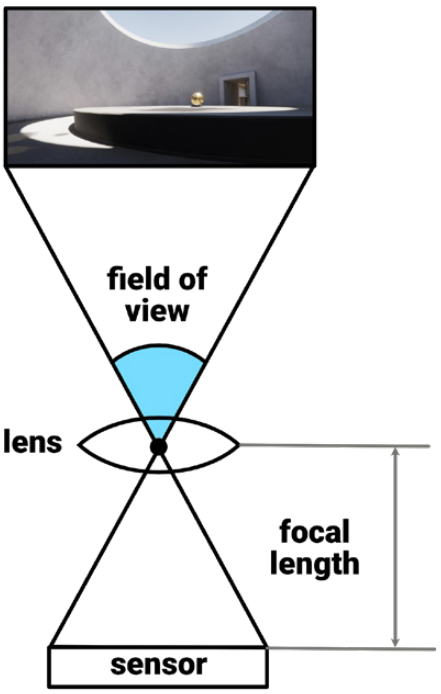
Important to exposure are the **ISO** (sensitivity), **Aperture** (or f-number), and **Shutter Speed**. If you are matching reference photos, copy the correct settings from the image's Exif data. Otherwise, this table can help you guesstimate Exposure Value based on f-number and shutter speed.

Additional Physical Camera parameters

Though not related to exposure, other Physical Camera properties can help you match the attributes of real-world cameras.

For example, we normally use **Field of View** in Unity (and many other 3D applications) to determine how much of the world a camera can see at once.

In real cameras, however, the field of view depends on the size of the sensor and focal length of the lens. Rather than setting the field of view directly, the Physical Camera settings allow you to fill in the **Sensor Type**, **Sensor Size**, and **Focal Length** from the actual camera data. Unity will then automatically calculate the corresponding Field of View value.

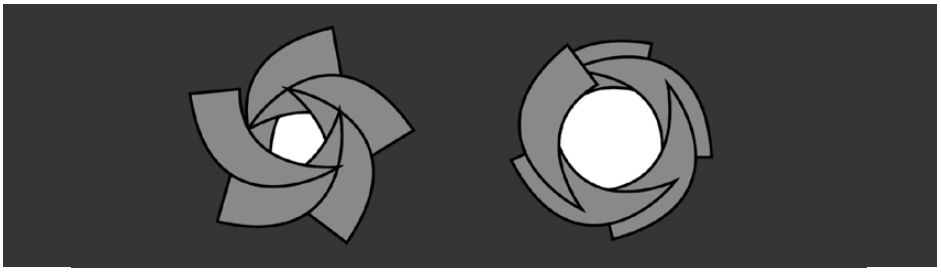


The relationship between focal length, sensor size, and field of view

Rely on the camera metadata included with the image files when trying to match a real photo reference. Both Windows and macOS can read the Exif data from digital images. You can then copy the corresponding fields to your virtual camera.

Note: You may need to search for the exact sensor dimensions on the manufacturer's website once you have the camera make and model from the metadata. This article includes an estimate of common image sensor formats.

Several of the bottom parameters influence the Depth of Field Volume. **Blade Count**, **Curvature**, and **Barrel Clipping** change the camera aperture's shape. This influences the look of the bokeh that results from the Depth of Field Volume component.



Use the physical camera parameters to reshape the aperture. The five-bladed iris can show a pentagonal (left) or circular (right) bokeh.

Lights

HDRP includes a number of different Light types and shapes to help you control illumination in your scene.

Light types

These Light types are available, similar to the other render pipelines in Unity:

— **Directional**: This behaves like light from an infinitely distant source, with perfectly parallel light rays that don't diminish in intensity. Directional lights often stand in for sunlight. In an exterior scene, this will often be your key light.

— **Spot**: This is similar to a real-world spotlight, which can take the shape of a cone, pyramid, or box. A spot falls off along the forward z-axis, as well as toward the edges of the cone/pyramid shape.

— **Point**: This is an omnidirectional light that illuminates all directions from a single point in space. This is useful for radiant sources of light, like a lamp or candle.

— **Area**: This projects light from the surface of a specific shape (a rectangle, tube, or disc). An area light functions like a broad light source with a uniform intensity in the center, like a window or fluorescent tube.

Modify how the spot, point, and area lights fall off with the **Range**. Many HDRP Lights diminish using the inverse square law, like light sources in the real world.

Shapes

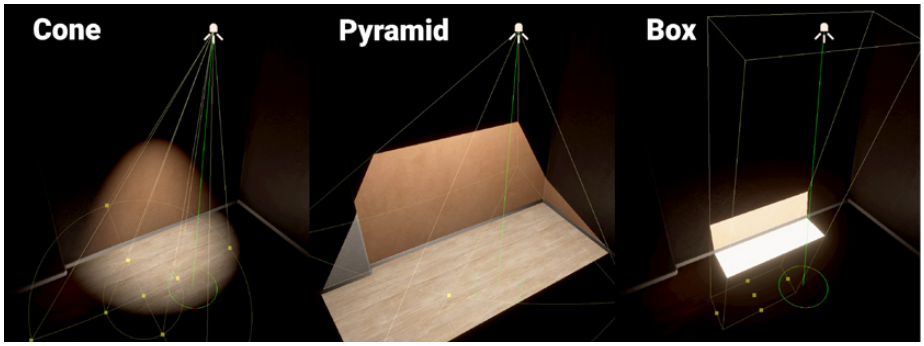
Spot and area lights have additional shapes for controlling how each light falls off.

HDRP spot lights can use three shapes:

— **Cone**: Projects light from a single point to a circular base. Adjust the **Outer Angle** (degrees) and **Inner Angle** (percentage) to shape the cone and modify its angular attenuation.

— **Pyramid**: Projects light from a single point onto a square base. Adjust the pyramid shape with the **Spot Angle** and **Aspect Ratio**.

— **Box**: Projects light uniformly across a rectangular volume. An X and Y size etermine the base rectangle, and the Range controls the Y dimension. This light has no attenuation unless **Range Attenuation** is checked and can be used to simulate sunlight within the boundary of the box.



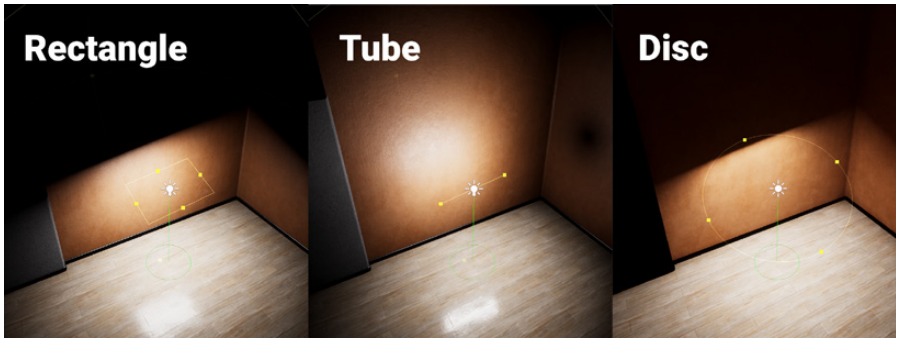
HDRP spot light shapes

HDRP area lights can use three shapes:

— **Rectangle**: Projects light from a rectangle shape in the local, positive Z direction out to a defined Range.

— **Tube**: Projects light from a single line in every direction, out to a defined Range. This light only works in Realtime Mode.

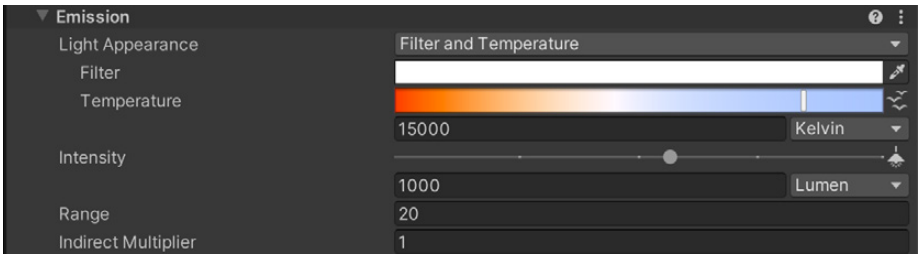
— **Disc**: Projects light from a disc shape in the local positive Z direction, out to a defined Range. This light only works in Baked Mode.



HDRP area light shapes

Color and temperature

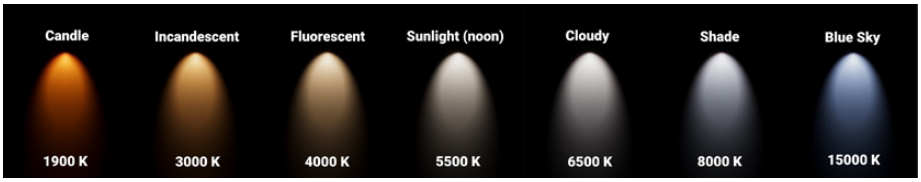
All HDRP Light types have **Emission** properties that define the light's appearance.



Modify Light Appearance properties under Emission

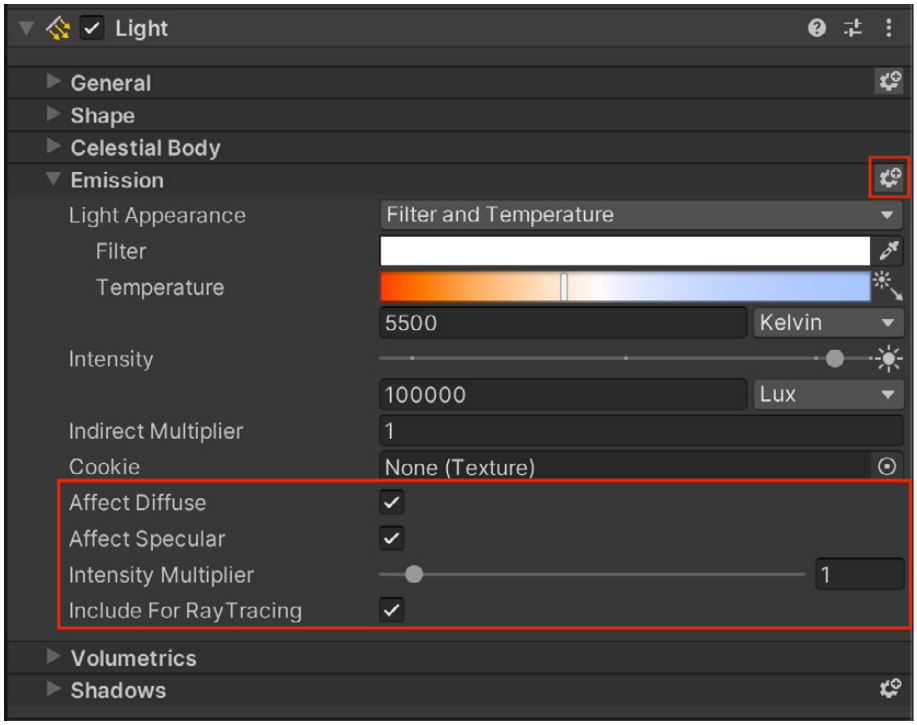
You can switch the **Light Appearance** to **Color** and specify an RGB color. Otherwise, change this to **Filter and Temperature** for more physically accurate input.

Color temperature sets the color based on degrees Kelvin. See the Lighting and Exposure Cheat Sheet for reference.

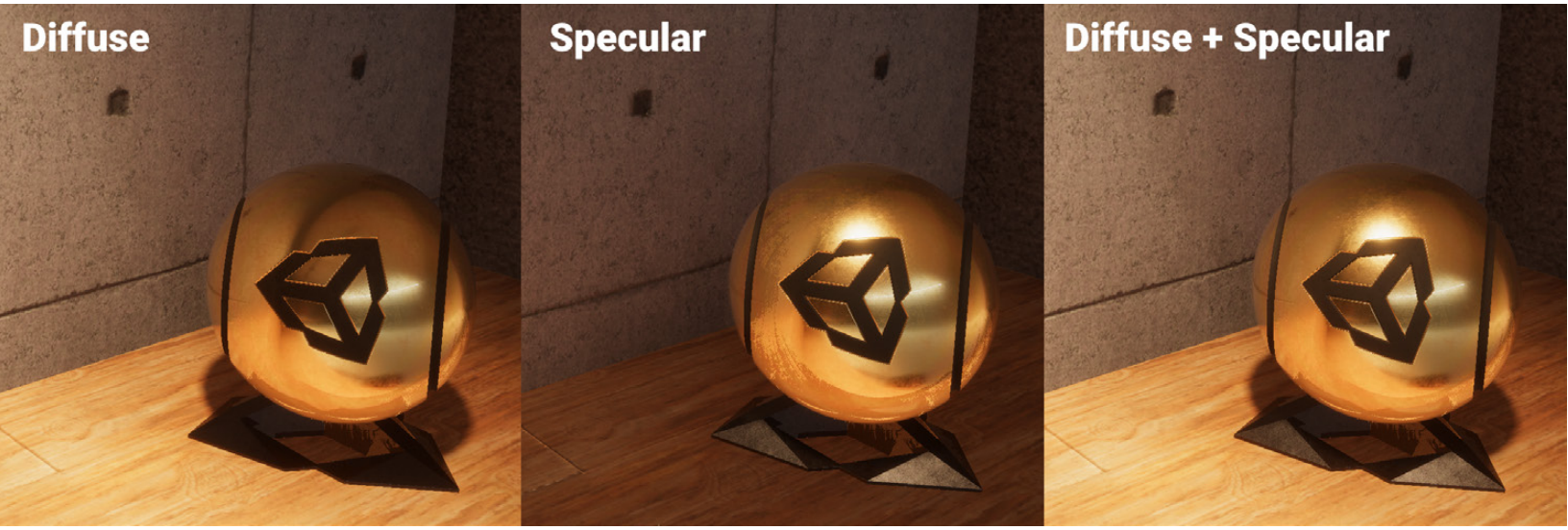


Color temperature on the Kelvin scale

You can also add another color that acts like a **Filter**, tinting the light with another hue. This is similar to adding a color gel in photography.



Each Light has additional properties.



Toggle diffuse and specular lighting for additional control.

Additional properties

HDRP also includes some advanced controls under the **More Options** button at the top right of the Inspector properties. Click it to reveal additional controls.

These include toggles for **Affect Diffuse** and **Affect Specular**. In cutscene or cinematic lighting, for example, you can separate Lights that control the bright shiny highlights independently from those that produce softer diffuse light.

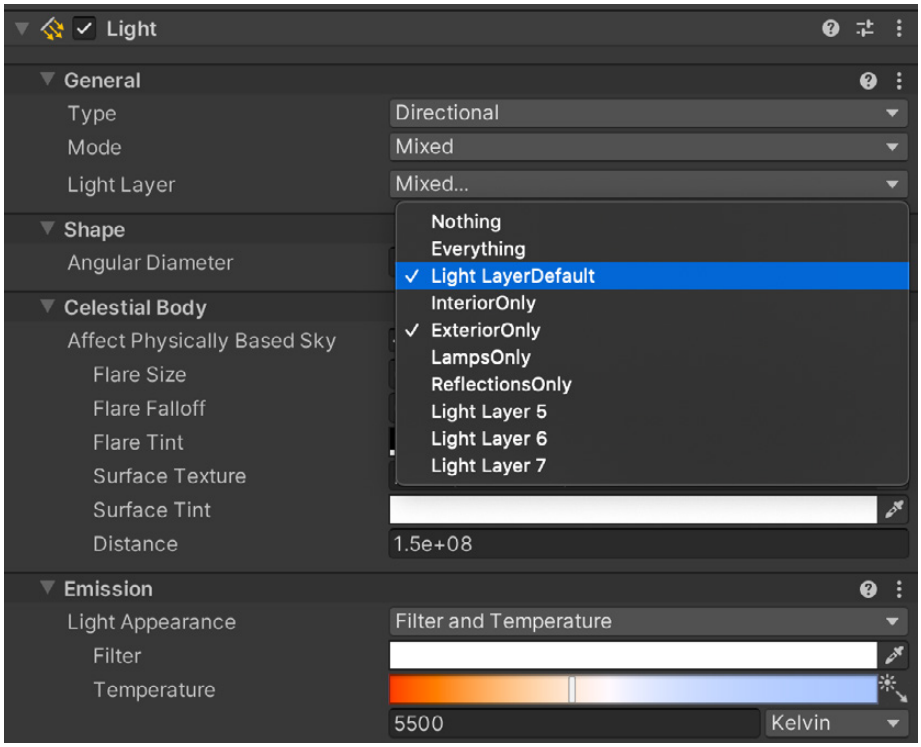
You can also use the **Intensity Multiplier** to adjust the overall intensity of the light without actually changing the original intensity value. This is useful for brightening or darkening multiple Lights at once.

Light Layers

HDRP allows you to use **Light Layers** to make Lights only affect specific meshes in your Scene. These are LayerMasks that you can associate with a Light component and MeshRenderer.

In the Light properties, click the **More Options** button. This displays the **Light Layer** dropdown under **General**. Choose which LayerMasks you want to associate with the Light.

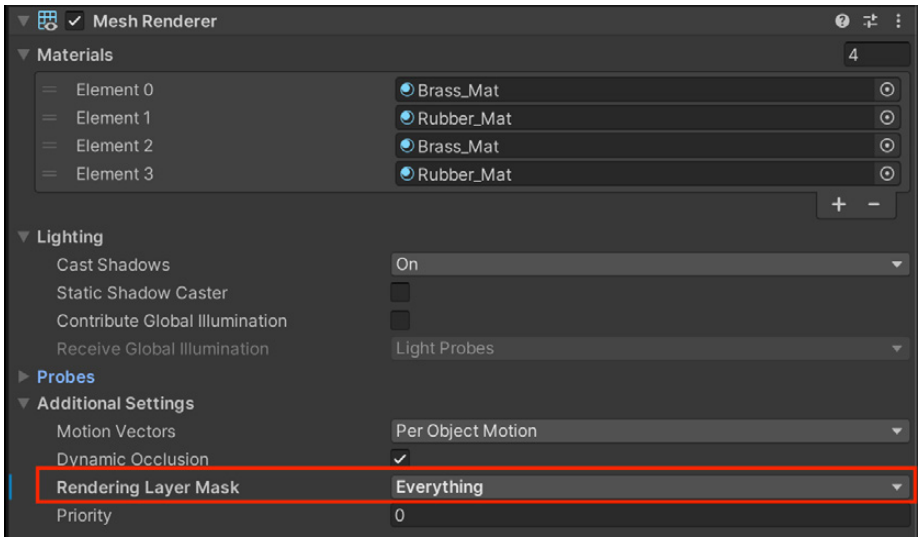
> In the Light properties, select **Show Additional Properties** from the **More Items** menu (☰). This displays the **Light Layer dropdown** under **General**. Choose what LayerMasks you want to associate with the Light.



Select a Light Layer

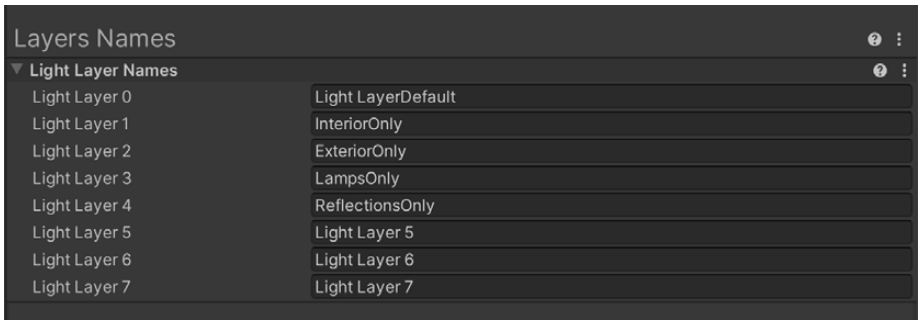
Next, set up the MeshRenderers with the **Rendering Layer Mask**. Only Lights on the matching LayerMask will affect the mesh. This feature can be invaluable for fixing light leaks, making sure that lights only strike their intended targets. It can also be part of the workflow to set up cutscene lighting, so that characters only can receive dedicated cinematic lights.

For example, if you wanted to prevent the lights inside of a building from accidentally penetrating the walls to the outside, you could set up specific Light Layers for the interior and exterior. This ensures that you have fine-level control of your Light setups.



Set the Rendering Layer Mask so that only specific Lights affect the mesh.

To set up your Light Layers, go to the **HDRP Default Settings**. The **Layers Names** section lets you set the string name for **Light Layer 0** to **7**.



Layers Names in the HDRP Default Settings

For more information, including the full list of Light properties, see the Light component documentation.

Physical Light Units and intensity

HDRP uses Physical Light Units (PLU) for measuring light intensity. These match real-life SI measurements for illuminance, including candela, lumen, lux, and nits. Note that PLU expects 1 unit in Unity to equal 1 meter for accuracy.

Units

Physical Light Units can include units of both *luminous flux* and *illuminance*. Luminous flux represents the total amount of light *emitted* from a source, while illuminance refers to the total amount of light received by an object (often in luminous flux per unit of area).

Because commercial lighting and photography may express units differently depending on application, Unity supports multiple Physical Light Units for compatibility:

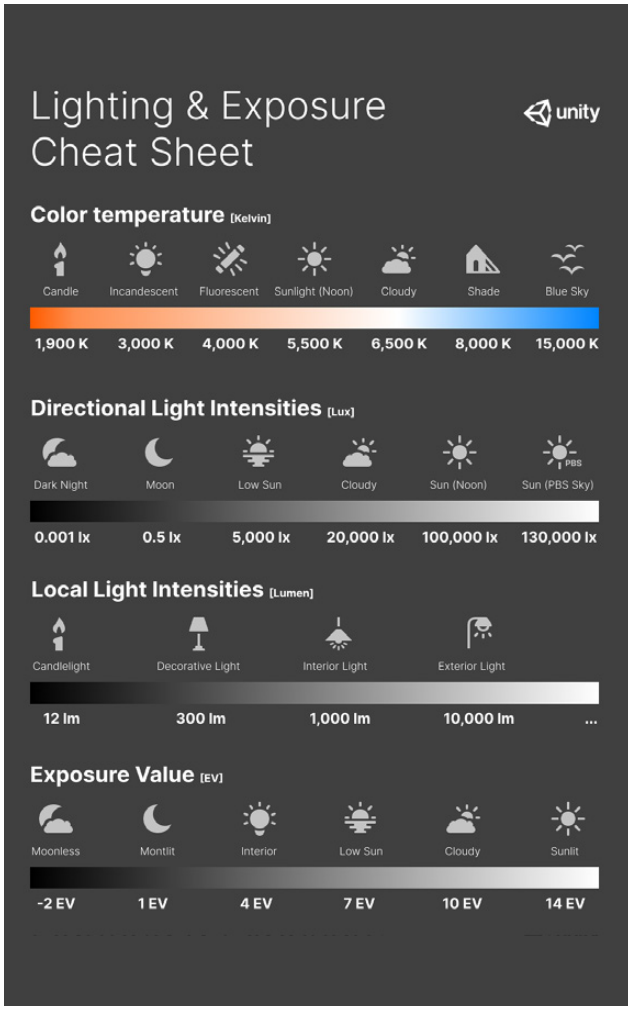
— **Candela**: One unit is equivalent to the luminous flux of a wax candle. This is also commonly called *candlepower*.

— **Lumen**: This is the SI unit of luminous flux defined to be the 1 candela over a solid angle (steradian). You will commonly see lumens on commercial lightbulb specifications. Use it with Unity spot, point, or area lights.

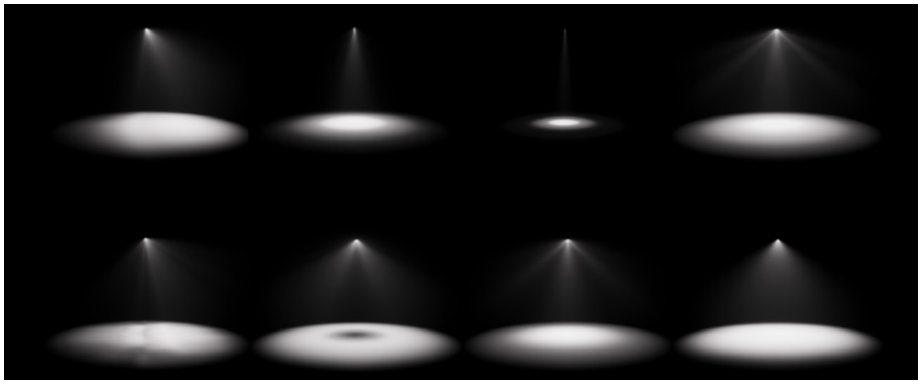
— **Lux**: A light source that emits 1 lumen onto an area of 1 square meter has an illuminance of 1 lux. Real-world light meters commonly read lux, and you will often use this unit with directional lights in Unity.

— **Nits**: This is a unit of luminance that is the equivalent of 1 candela per square meter. Display devices and LED panels (televisions or monitors, for example) often measure their brightness in Nits.

— **EV_{100}**: This uses an intensity corresponding to **EV_{100}**, which is an exposure value with 100 ISO film (see exposure value formula above). Incrementing the exposure results in the doubling of the lighting, due to the logarithmic behavior.



Guidance for lighting and exposure levels



IES profiles applied to various lights

For recreating a real lighting source, switch to the unit listed on the tech specs and plug in the correct luminous flux or luminance. HDRP will match the Physical Lighting Units, eliminating much of the guesswork when setting intensities.

Click the icon to choose presets for **Exterior**, **Interior**, **Decorative**, and **Candle**. These settings provide a good starting point if you are not explicitly matching a specific value.

Common lighting and exposure values

The following cheat sheet contains the color temperature values and light intensities of common real-world light sources. It also contains exposure values for different lighting scenarios.

You can find a complete table of common illumination values in the Physical Light Units documentation.

IES Profiles and Cookies

Make your point, spot, and area lights more closely mimic the falloff of real lights using an **IES profile**. This works like a light cookie to apply a specific manufacturer's specs to a pattern of light. IES profiles can give your lights an extra boost of realism.

Import an IES profile from **Assets > Import New Asset**. The importer will automatically create a Light Prefab with the correct intensity. Then just drag the Prefab into the Scene view or Hierarchy and tweak its color temperature.

Here are some sources for IES profiles:

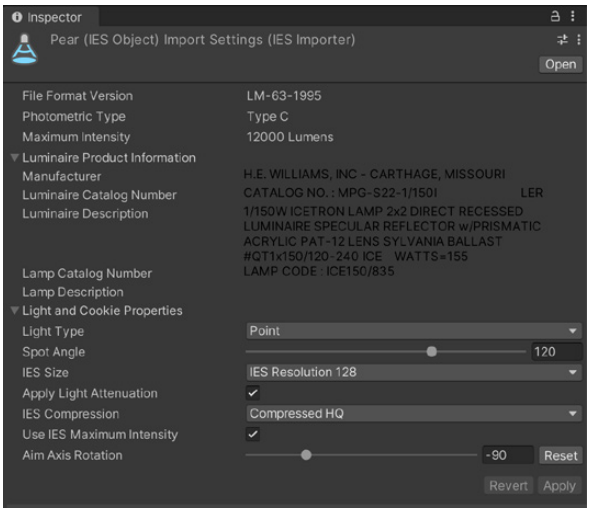
Real-world manufacturers

— Philips
— Lithonia Lighting
— Efficient Lighting Systems
— Atlas
— Erco
— Lamp
— Osram

Artist sources

— Renderman

For information on the IES profile importer, see the documentation.



IES Profile and Import Settings

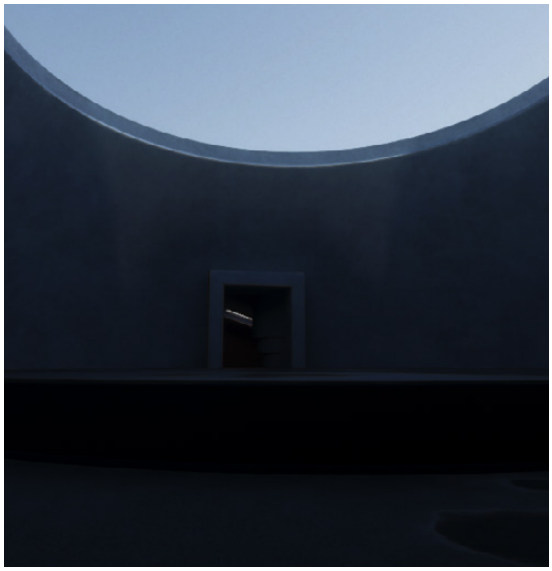
Environment lighting

In the real world, light reflects and scatters around us. The sky and ground contribute to environment lighting as random photons bounce between the atmosphere and earth and ultimately arrive at the observer.

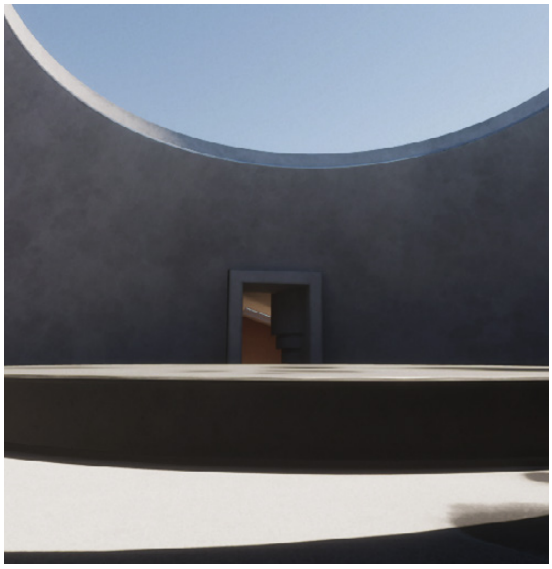
In HDRP, you can use the **Visual Environment** override to define the sky and general ambience for a scene.

Use **Ambient Mode: Dynamic** to set the sky lighting to the current override that appears in the Visual Environment's **Sky > Type**. Otherwise, **Ambient Mode: Static** defaults to the sky setup in the **Lighting** window's **Environment** tab.

Even with your other light sources disabled, the SampleScene receives general ambient light from the Visual Environment.



Environment lighting only – with the sun directional light disabled, the sky still provides ambient light.



Direct sunlight combined with the environment lighting

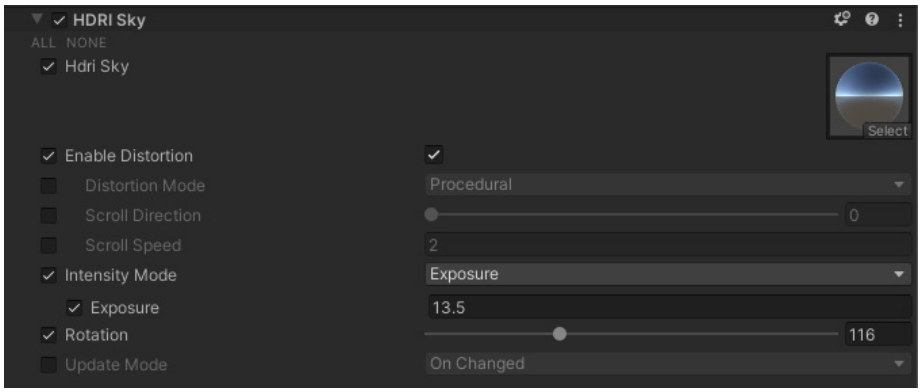
Adding the key light of the sun completes the general illumination of the scene. The environment light helps fill in the shadow areas so that they don't appear unnaturally dark.

HDRP includes three different techniques for generating skies. Set the **Type** to either **HDRI Sky**, **Gradient Sky**, or **Physically Based Sky**. Then, add the appropriate override from the Sky menu.

Applying a Visual Environment sky is similar to wrapping the entire virtual world with a giant illuminated sphere. The colored polygons of the sphere provide a general light from the sky, horizon, and ground.

HDRI Sky

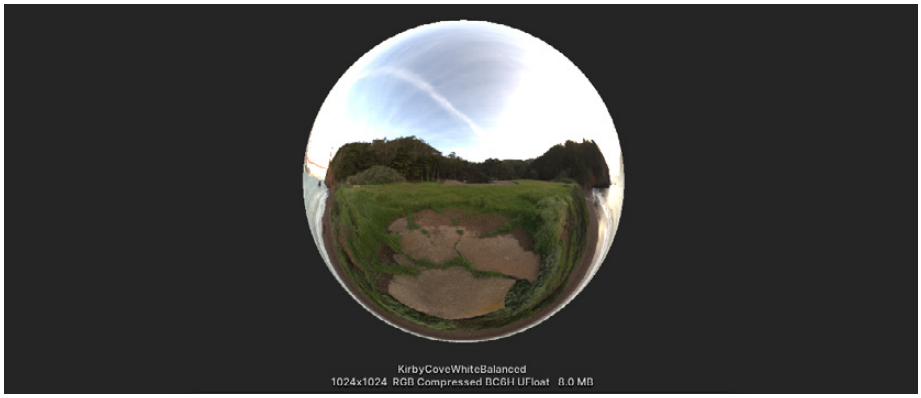
HDRI Sky allows you to represent the sky with a cubemap made from high-dynamic range photographs. You can find numerous free and low-cost sources of HDRIs online. A good starting point is the Unity HDRI Pack in the Asset Store. If you're adventurous, we also have a guide to shooting your own HDRIs.



HDRI Sky

Once you've imported your HDRI assets, add the **HDRI Sky** override to load the **HDRI Sky** asset. This lets you also tweak options for **Distortion**, **Rotation**, and **Update Mode**.

Because the sky is a source of illumination, specify the **Intensity Mode**, then choose a corresponding **Exposure/Multiplier/Lux** value to control the strength of the environmental lighting. Refer to the Lighting and Exposure Cheat Sheet above for example intensity and exposure values.

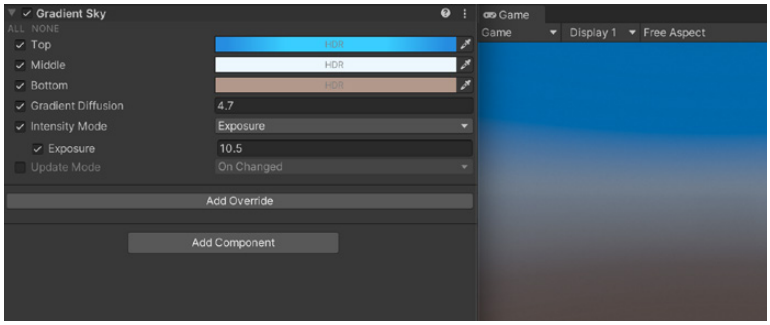


An HDRI Sky applied as a cubemap to the interior of a sphere

Gradient Sky

Choose Gradient Sky in the Visual Environment to approximate the background sky with a color ramp. Then add the **Gradient Sky** override. Use the **Top**, **Middle**, and **Bottom** to determine colors for the gradient.

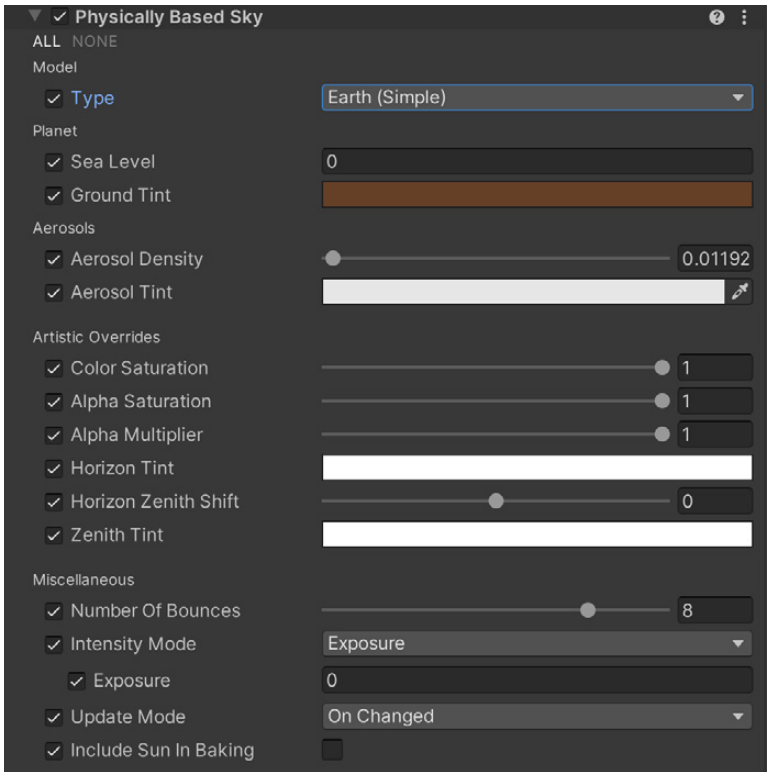
Blend the color ramp with **Gradient Diffusion**, and dial the **Intensity** for the strength of the lighting.



The Top, Middle, and Bottom colors blend into a Gradient Sky.

Physically Based Sky

For something significantly more realistic than a gradient, you can use the **Physically Based Sky** override. This procedurally generates a sky that incorporates phenomena such as Mie scattering and Rayleigh scattering. These simulate light dispersing through the atmosphere, recreating the coloration of the natural sky. Physically Based Sky requires a directional light for accurate simulation.



Physically Based Sky override

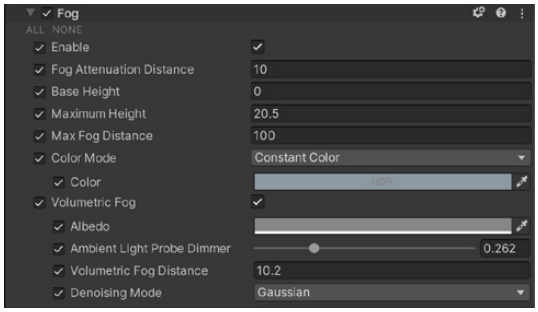
A procedurally generated sky from the *Fountainebleau* Demo

Use the Base Height and Maximum Height to make a low-hanging fog.

Fog and atmospheric scattering

Smoke, fog, and haze are traditional tools of cinematography. They can help add depth and dimension to stage lighting or create an atmospheric mood. You can use fog for a similar advantage in HDRP.

Its opacity depends on the object's distance away from the camera. Fog can also hide the camera's far clipping plane, blending your distant geometry back into the scene.



Fog override

Global fog

HDRP implements global fog as a **Fog** override. Here, the fog fades exponentially with its distance from the camera and its world space height.

Set up the Fog override on a Volume in your Scene. The **Base Height** determines a boundary where constant, thicker fog begins to thin out traveling upward. Above this value, the fog density continues fading exponentially, until it reaches the **Maximum Height**.

Likewise, the **Fog Attenuation Distance** and **Max Fog Distance** control how fog fades with greater distance from from the camera. Toggle the **Color Mode** between a **Constant Color** or the existing **Sky Color**.

Fog Attenuation Distance

Enable **Volumetric Fog** to simulate atmospheric scattering. Make sure to check **Fog** and **Volumetrics** in the **Frame Settings** (either under the camera or in HDRP Default Settings) under Lighting. Also, enable **Volumetric Fog** in the HDRP Pipeline Asset.

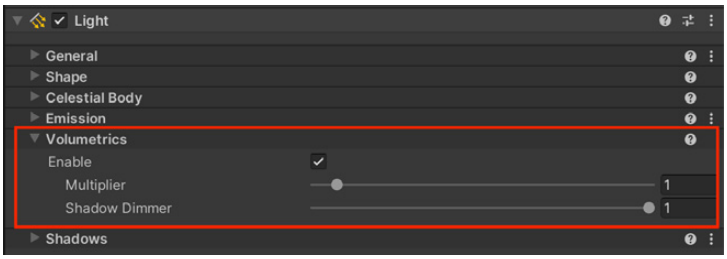
Volumetric Fog Distance sets the distance (in meters) from the Camera's Near Clipping Plane to the back of its volumetric lighting buffer. This fills the atmosphere with an airborne material, partially occluding GameObjects within range.

Volumetric Fog more accurately holds out the foreground geometry.

Volumetric Lighting

Volumetric Lighting can simulate rendering dramatic sunbeams, like crepuscular rays behind the clouds at sunset or passing through foliage.

Each Light component (except area lights) has a **Volumetrics** group. Check **Enable**, then set the **Multiplier** and **Shadow Dimmer**. A **Real-time** or **Mixed Mode** light will produce 'god rays' within Volumetric Fog. The Multiplier dials the intensity, while the Shadow Dimmer controls how shadow casting surfaces cut into the light.



Volumetrics in the Light component

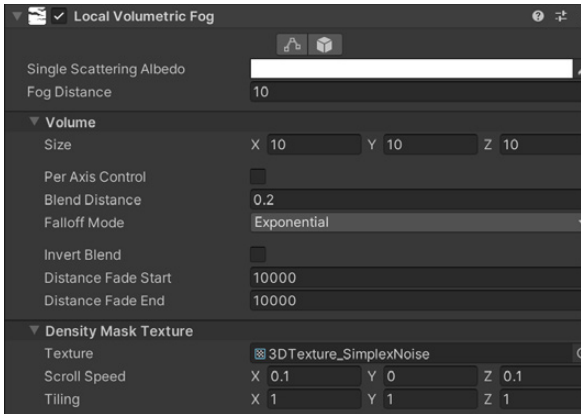
Room 2 in the Sample Scene features a skylight and Volumetric Fog. The frame of the glass case carves volumetric shadows out of the sunbeams from the ceiling. Dial up the Shadow Dimmer and exaggerate the Multiplier to intensify the effect.

Volumetric Lighting applied to the ceiling spotlights

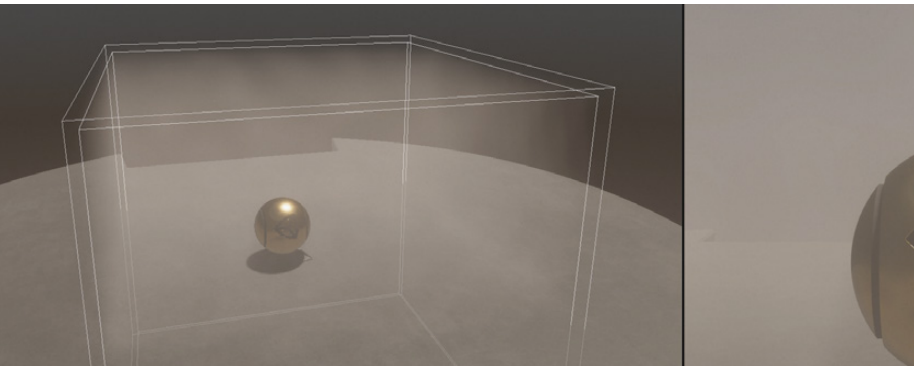
Local Volumetric Fog

If you want more detailed fog effects than the Fog override can provide, HDRP additionally offers Local Volumetric Fog (called a Density Volume component before HDRP 12).

This is a separate component, outside the Volume system. Create a **Local Volumetric Fog** GameObject from the menu (**GameObject > Rendering > Local Volumetric Fog**) or right-click over the Hierarchy (**Rendering > Local Volumetric Fog**).



The Local Volumetric Fog component



Local Volumetric Fog appears in a bounding box.

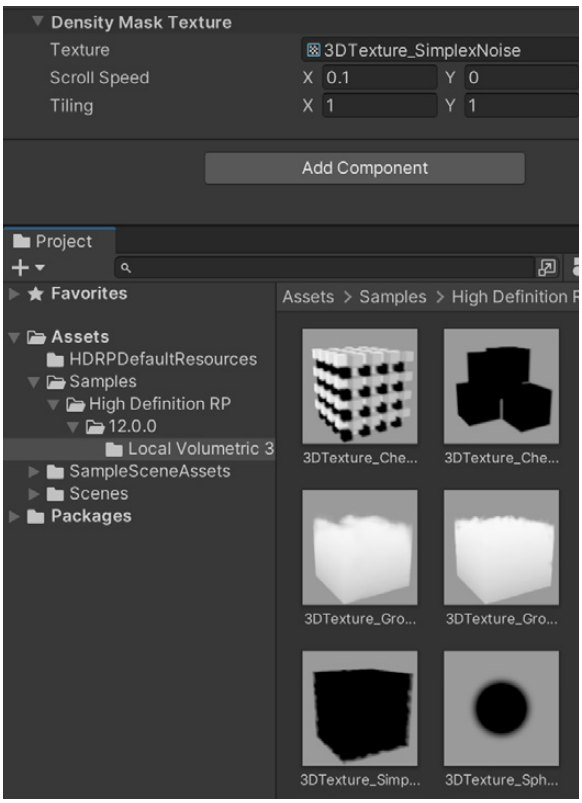
Volumetric Lighting dramatically illuminates the Volumetric Fog areas.

This generates a fog-filled bounding box. Adjust the size, axis control, and blending/fading options.

By default, the fog is uniform, but you can apply a 3D **Texture** to the Texture field under the **Density Mask Texture** subsection. This gives the user more flexibility to control the look of the fog. Download examples from the Package Manager's **Local Volumetric 3D Texture Samples** or follow the documentation procedures to create your own Density Masks.

Add some **Scroll Speed** for animation and adjust the **Tiling**. Your Volumetric Fog then can gently roll through the scene.

Note: HDRP voxelizes Local Volumetric Fog to enhance performance. However, the voxelization can appear very coarse. To reduce aliasing, use a Density Mask Texture and increase the Blend Distance to soften the fog's edges.



Density Mask Texture from the Local Volumetric 3D Texture Samples

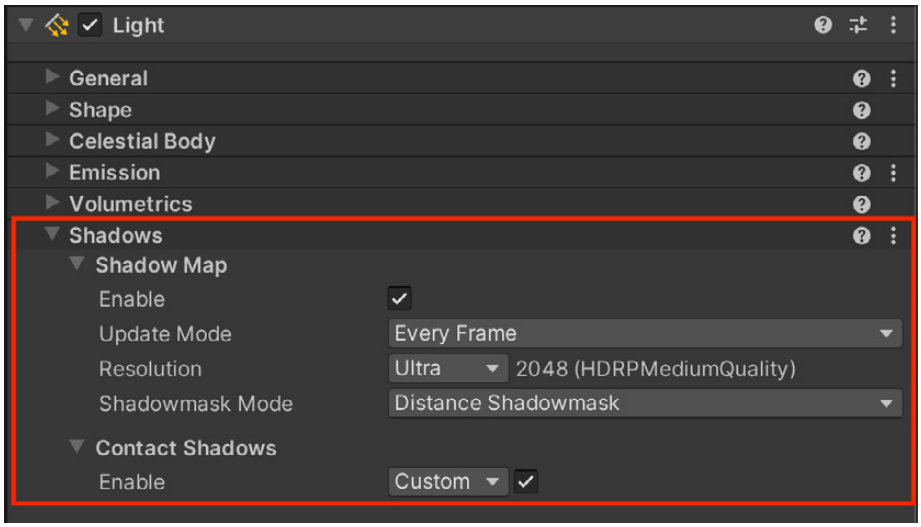
Shadows

We can't perceive light without darkness. Well-placed shadows in your scene add as much interest as the lighting itself and can imbue your scenes with extra depth and dimension. HDRP includes a number of features to fine-tune your shadows and prevent your renders from looking flat.

Shadow maps

Shadows render using a technique called **shadow mapping**, where a texture stores the depth information from the light's point of view.

Locate the Shadows subsection of the Light component to modify your shadow mapping **Update Mode** and **Resolution**. Higher resolutions and update frequency settings cost more resources.



Shadow settings per light

Shadow Cascades

For a directional light, the shadow map covers a large portion of the scene, which can lead to a problem called *perspective aliasing*. Shadow map pixels close to the camera look jagged and blocky compared to those farther away.

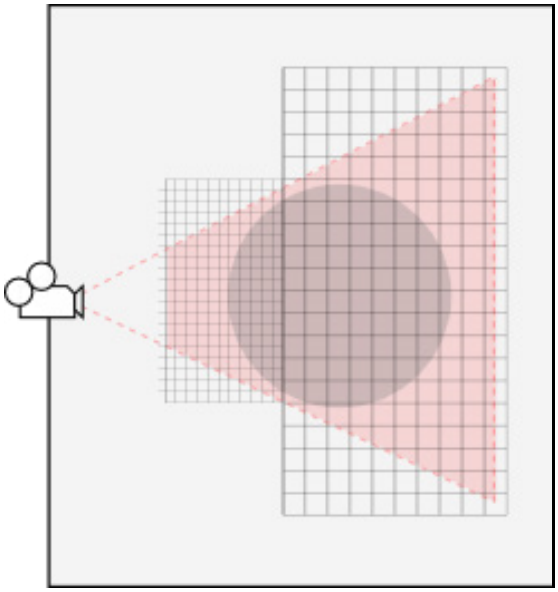
Perspective aliasing with blocky shadows

Shadow Cascades reduce perspective aliasing.

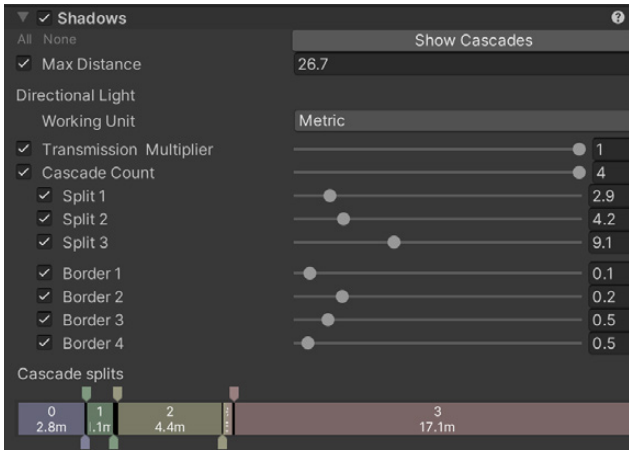
Unity solves this with cascaded shadow maps. It splits the camera frustum into zones, each with its own shadow map. This reduces the effect of perspective aliasing.

HDRP gives you extra control over your Shadow Cascades with the **Shadows** override. Use the cascade settings per Volume to fine-tune where each cascade begins and ends.

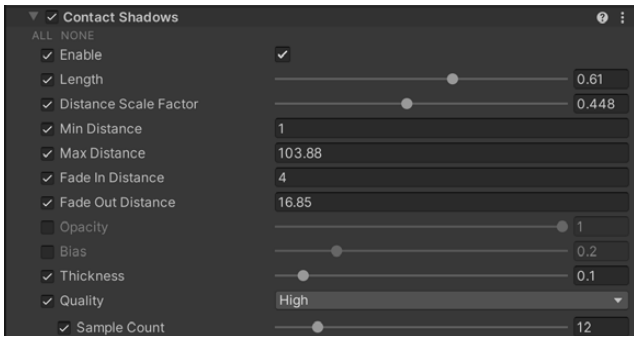
Toggle the **Show Cascades** button to visualize the cascade splits more easily. With some tweaking, you can keep perspective aliasing to a minimum.



Shadow Cascades break the camera frustum into zones, each with its own shadow map.



Shadow override



Contact Shadows override

Contact Shadows

Shadow maps often fail to capture the small details, especially at discernible edges where two mesh surfaces connect. HDRP can generate these Contact Shadows using the **Contact Shadows** override.

Contact Shadows are a screen space effect and rely on information within the frame in order to calculate. Objects outside of the frame do not contribute to Contact Shadows. Use them for shadow details with a small onscreen footprint.

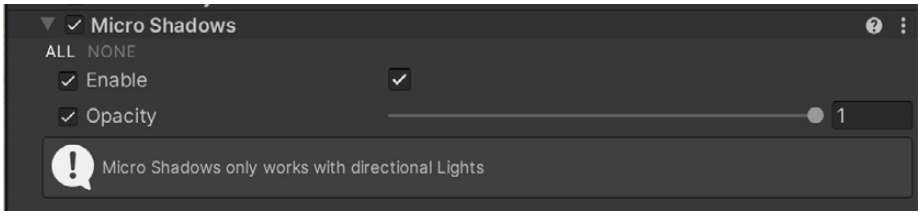
Make sure that you enable **Contact Shadows** in the **Frame Settings**. You can also adjust the **Sample Count** in the Pipeline Asset under **Lighting Quality Settings**.

Click Show Cascades to visualize the cascade splits.

Micro shadows

HDRP can extend even smaller shadow details into your Materials. **Micro shadows** use the normal map and ambient occlusion map to render really fine surface shadows without using the mesh geometry itself.

Simply add the **Micro shadows** override to a Volume in your Scene and adjust the Opacity. Micro shadows only work with directional lights.



Micro shadows override

Micro shadows add extra contrast to this bed of foliage.

Adjust the settings for the shadow thickness, quality, and fade.

Reflections

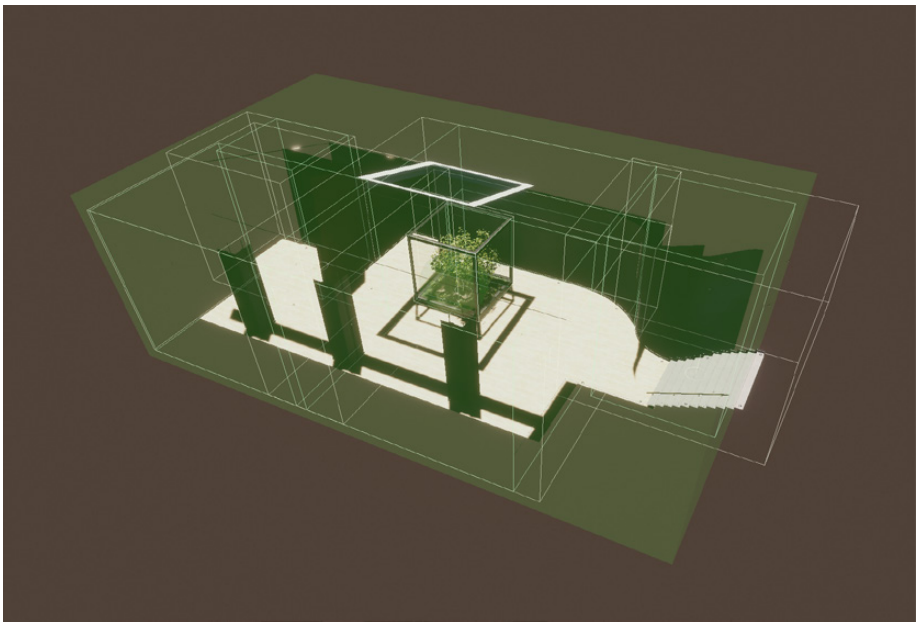
Reflections help integrate your GameObjects with their surrounding environment. Though we normally associate reflections with smooth and shiny surfaces, even rough materials need to receive correct reflections in a PBR workflow. HDRP offers multiple techniques to generate reflections:

— Screen Space Reflections

— Reflection Probes

— Sky reflections

Each reflection type can be resource intensive, so select the method that works best depending on your use case. If more than one reflection technique applies to a pixel, HDRP blends the contribution of each reflection type. Bounding surfaces called Influence Volumes partition the 3D space to determine what objects receive the reflections.

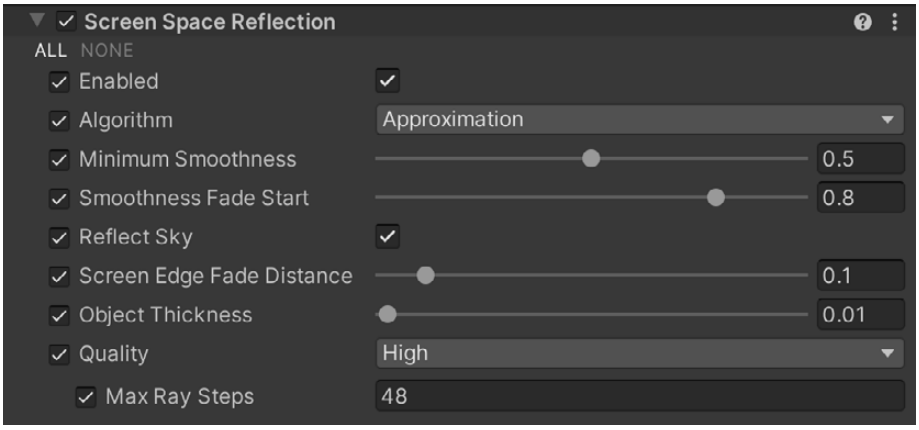


Influence Volumes determine where Reflection Probes create reflections.

Screen Space Reflections

Screen Space Reflections use the depth and color buffer to calculate reflections. Thus they can only reflect objects currently in camera view and may not render properly at certain positions on screen. Glossy floorings and wet planar surfaces are good candidates for receiving Screen Space Reflections.

Screen Space Reflections ignore all objects outside of the frame, which can be a limitation to the effect.



Screen Space Reflection override

Screen Space Reflections

Make sure you have **Screen Space Reflection** enabled in the **Frame Settings (HDRP Default Settings** or the **Camera's Custom Frame Settings**) under **Lighting**. Then, add the **Screen Space Reflection** override to your Volume object.

Material surfaces must exceed the **Minimum Smoothness** value to show Screen Space Reflections. Lower this value if you want rougher materials to show the SSR, but be aware that a lower Minimum Smoothness threshold can add to the computation cost. If Screen Space Reflection fails to affect a pixel, then HDRP falls back to using Reflection Probes.

Use the **Quality** dropdown to select a preset number of **Max Ray Steps**. Higher Max Ray Steps increase quality but come with a cost. As with all effects, balance performance with visual quality.

Reflection Probes

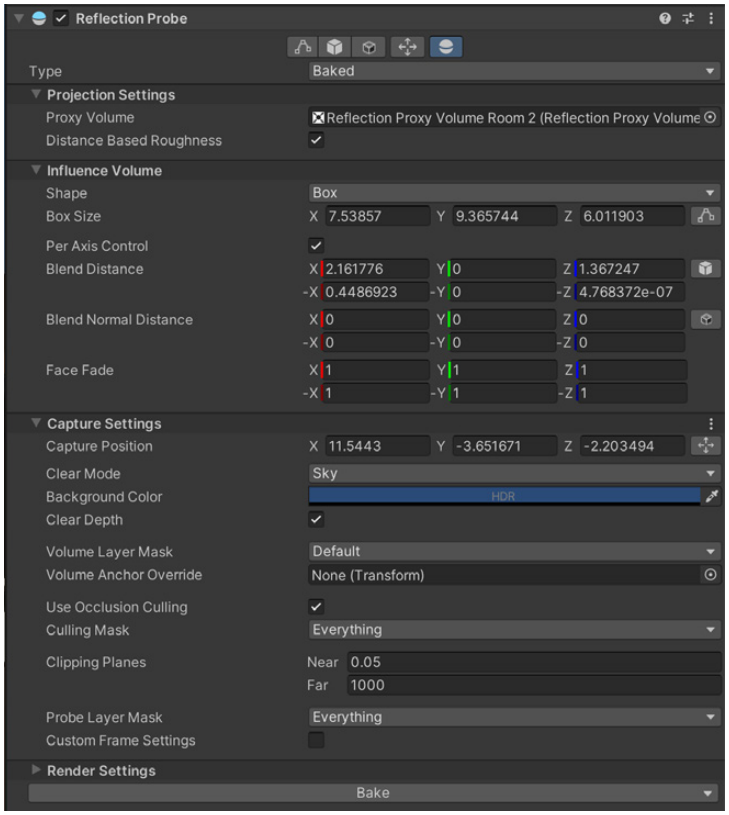
Reflection Probes generate reflections using an image-based technique. A probe captures a spherical view of its surroundings in all directions and stores the result in a cubemap texture. A shader uses that cubemap to approximate a reflection.

Each Scene can have several probes and blend the results. Localized reflections then can change as your camera moves through the environment.

Set the **Type** of each probe to **Baked** or **Real-time**:

— *Baked probes* process the cubemap texture just once for a static environment.

— *Real-time probes* create the cubemap at runtime in the Player rather than in the Editor. This means that reflections are not limited to static objects, but be aware that real-time updates can be resource intensive.



Reflection Probe component

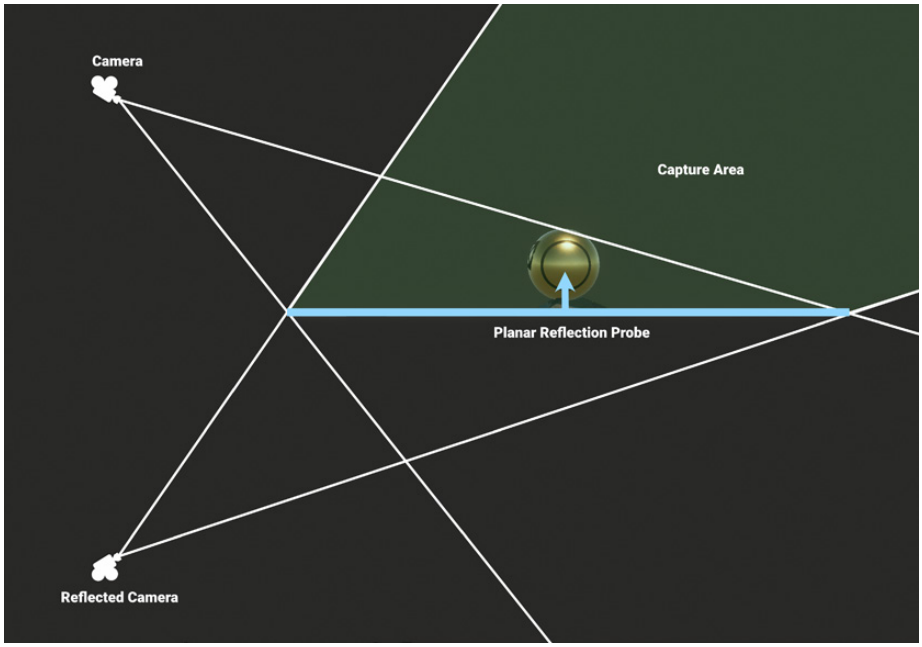
The **Influence Volume** determines the 3D boundaries where GameObjects will receive the reflection, while the **Capture Settings** let you customize how the Reflection Probe takes a snapshot of the cubemap.

Planar Reflection Probe

A Planar Reflection Probe allows you to recreate a flat, reflective surface, taking surface smoothness into account. This is perfect for a shiny mirror or floor.

Though a Planar Reflection Probe shares much in common with a standard Reflection Probe, it does work slightly differently. Rather than capture the environment as a cubemap, it recreates the view of a camera reflected through the probe's mirror plane.

The probe then stores the resulting mirror image in a 2D RenderTexture. Drawn to the boundaries of the rectangular probe, this creates a planar reflection.

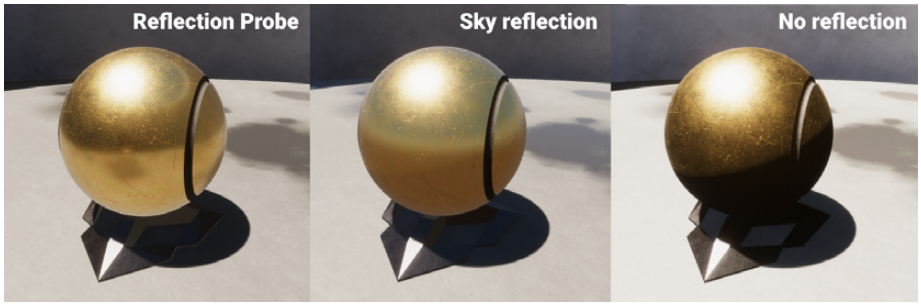


A Planar Reflection Probe captures a mirror image by reflecting a camera through a plane.

Planar Reflection Probes applied to three different materials

Sky reflection

When an object is not affected by a nearby Reflection Probe, it will fallback to the sky reflection.



A Reflection Probe shows the surrounding room whereas a sky reflection reflects the Gradient Sky.

Reflection hierarchy

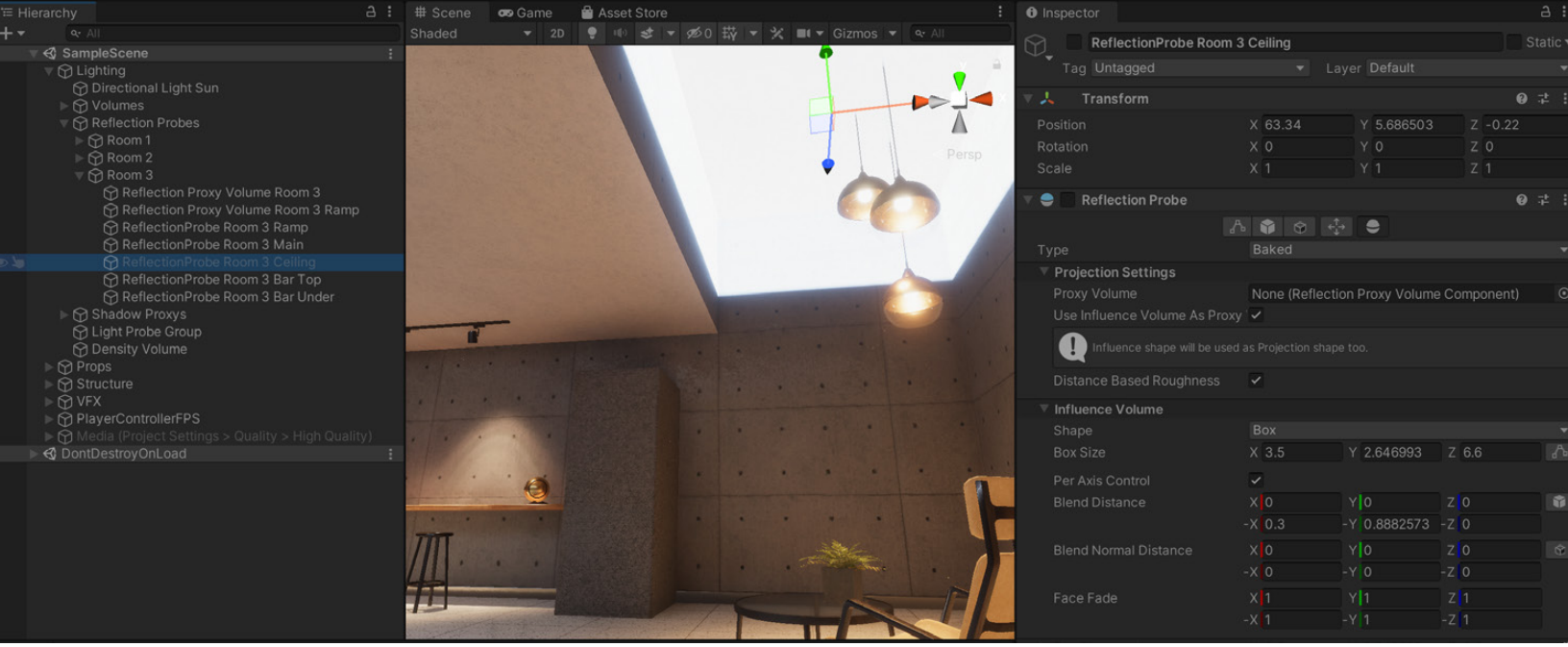
To produce the highest-quality reflections, HDRP uses the technique that gives the best accuracy for each pixel and allows it to blend with the other techniques. HDRP checks the three reflection methods (SSR, Reflection Probes, sky) using a weighted priority. This specific sequence for evaluating reflections is called the **Reflection hierarchy**.

When one technique does not fully determine the reflection at a pixel, HDRP falls back to the next technique. In other words, Screen Space Reflection falls back to the Reflection Probes, which in turn fall back to sky reflections.

It's important to set up your Influence Volumes for your Reflection Probes properly. Otherwise, you could experience light leaking from unwanted sky reflections.

This is apparent in Room 3 of the SampleScene. Disabling one of the Reflection Probes or shifting its Influence Volume forces the reflection to fall back to the sky. This causes the bright HDRI sky to overpower the scene with its intense reflections.

For more details about determining Reflection Hierarchy, see the Reflection in the HDRP documentation page.

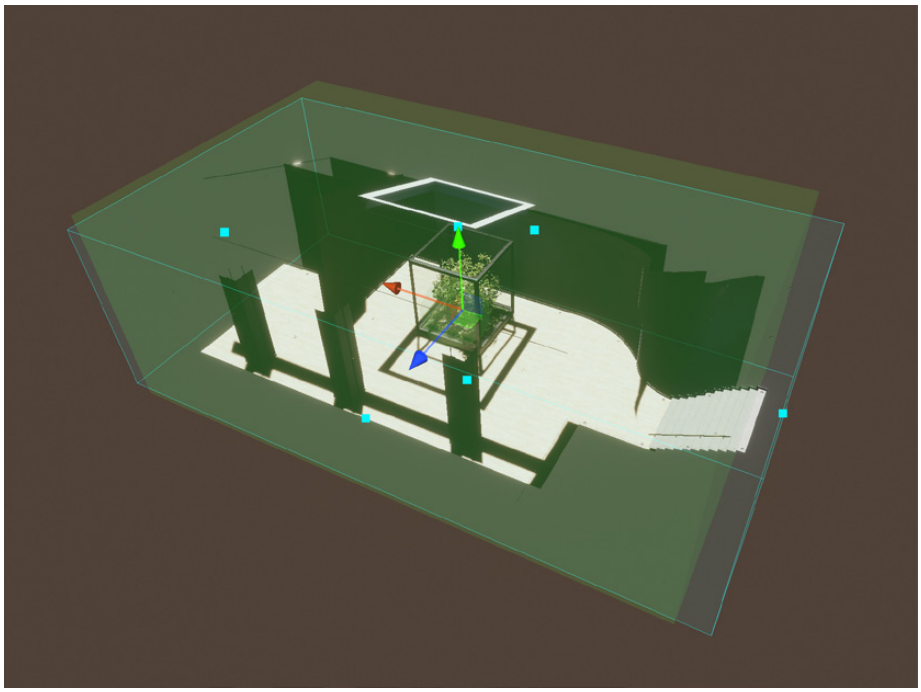


Disabling the Reflection Probe on the Room 3 ceiling causes unwanted light leaking.

Reflection Proxy Volumes

Because the capture point of a Reflection Probe is fixed and rarely matches the Camera position near the Reflection Probe, there may be a noticeable perspective shift in the resulting reflection. As a result, the reflection might not look connected to the environment.

A Reflection Proxy Volume helps you partially correct this. It reprojects the reflections more accurately within the proxy Volume, based on the Camera position.



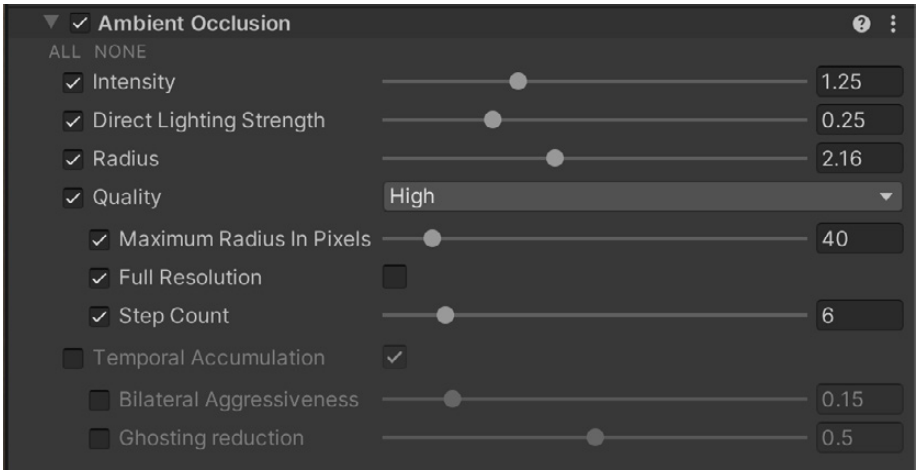
A Reflection Proxy Volume reprojects the cubemap to match the room's world space position.

Real-time lighting effects

HDRP also features some real-time lighting effects available through the Volume system. Select a local or global Volume, then add the appropriate effect under **Add Override > Lighting**.

Screen Space Ambient Occlusion

Ambient occlusion simulates darkening that occurs in creases, holes, and surfaces that are close to one another. Areas that block out ambient light appear occluded.



Ambient Occlusion override

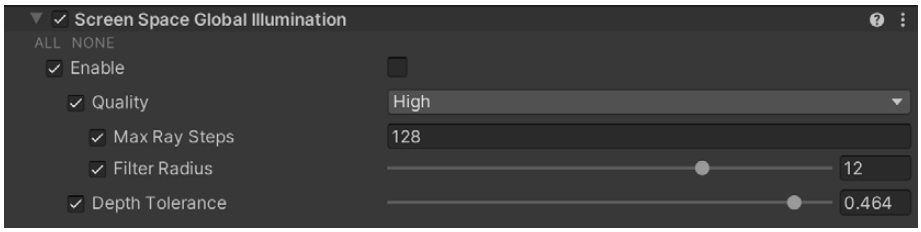
Ambient occlusion visualization in the Auto Showroom project

Though you can bake ambient occlusion for static geometry through Unity's Lightmapper, HDRP adds an additional **Screen Space Ambient Occlusion**, which works in real time. Because this is a screen space effect, only information from within the frame can contribute to the effect produced. SSAO ignores all objects outside of the camera's field of view.

Enable **Screen Space Ambient Occlusion** in the **Frame Settings** under **Lighting**. Then, **Add Override** on a local or global Volume and select **Lighting > Ambient Occlusion**.

Screen Space Global Illumination

Screen Space Global Illumination (SSGI) uses the depth and color buffer of the screen to calculate bounced, diffuse light. Much like how lightmapping can bake indirect lighting into the surfaces of your static level geometry, SSGI more accurately simulates how photons can strike surfaces and transfer color and shading as they bounce.



Screen Space Global Illumination override

In Room 2 of the Sample Scene, we can see the green of the moving tree leaves transfer through bounced light onto the wall with SSGI enabled.

Screen Space Global Illumination captures the bounced light from the foliage in real time.

Like other effects that depend on the frame buffer, the edges of the screen become problematic, since objects outside the camera's field of view cannot contribute to the global illumination.

SSGI is enabled under the **Frame Settings** under **Lighting**, and it must be enabled in the Pipeline Asset's **Lighting** section as well.

Note: We recommend that you use Unity 2021.2 or above with Screen Space Global Illumination. HDRP 12 includes significant improvements to SSGI quality.

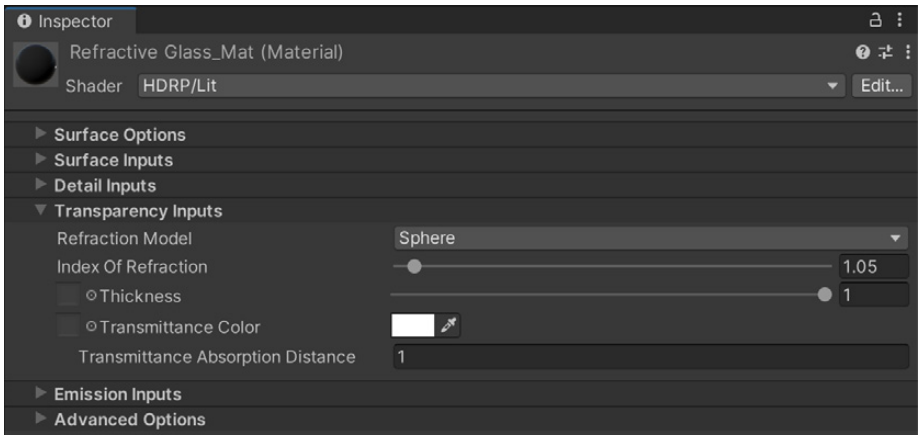
Screen Space Refraction

The Screen Space Refraction override helps to simulate how light behaves when passing through a denser medium than air. HDRP's Screen Space Refraction uses the depth and color buffer to calculate refraction through a transparent material like glass.

Screen Space Refraction override

To enable this effect through the **HDRP/Lit shader**, make sure your material has a Surface Type of **Transparent**.

Then choose a **Refraction Model** and **Index of Refraction** under **Transparency Inputs**. Use the **Sphere** Refraction Model for solid objects. Choose **Thin** (like a bubble) or **Box** (with some slight thickness) for hollow objects.



Transparency Inputs to control refraction

Screen Space Refraction

Post-processing

Modern high-end graphics would be incomplete without post-processing. While we can't always "fix it in post," it's difficult to imagine our rendered images without the filters and full-screen image effects that make them more cinematic. Thus, HDRP comes bundled with its own built-in post-processing effects.

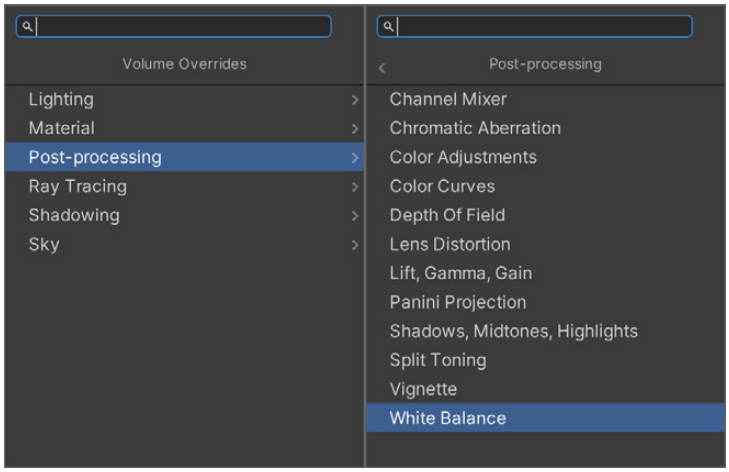
Post-processing effects make your renders more cinematic.

HDRP post-processing uses the Volume system to apply the image effects to the camera. Once you know how to add overrides, the process of applying more post effects should already be familiar.

Post-processing overrides

Many of these post-processing overrides for controlling color and contrast may overlap in functionality. Finding the right combinations of them may require some trial and error.

You won't need *every* effect available. Just add the overrides necessary to create your desired look and ignore the rest.



Post-processing overrides

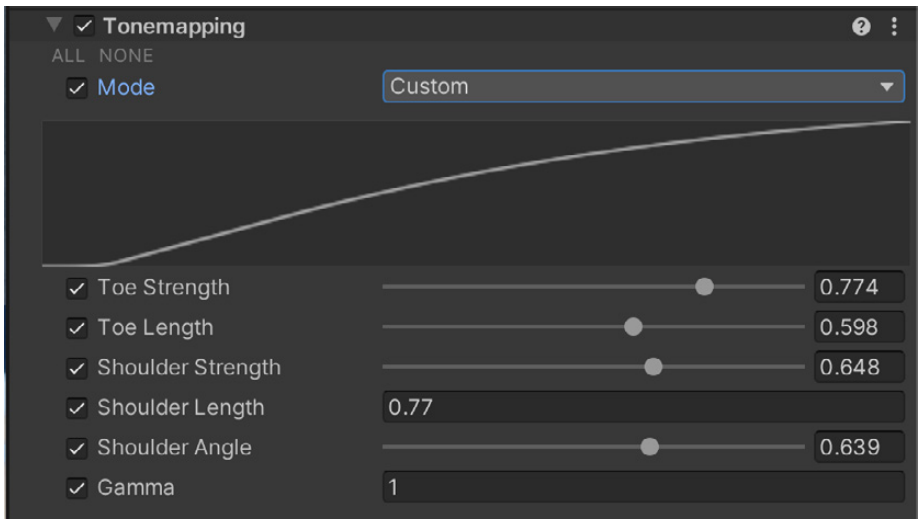
Refer to the Volumes in the SampleScene for example usage.

Tonemapping

Tonemapping is a technique for mapping high-dynamic-range colors to the more limited dynamic range of your screen. It can enhance the contrast and detail in your renders.

ACES versus Neutral tonemapping

If you want a filmic look, set the **Mode** to the industry standard **ACES** (Academy Color Encoding System). For something less saturated and contrasted, select **Neutral**. Experienced users also have the option of choosing **Custom** and defining the tonemapping curve for themselves.



Tonemapping with a custom curve

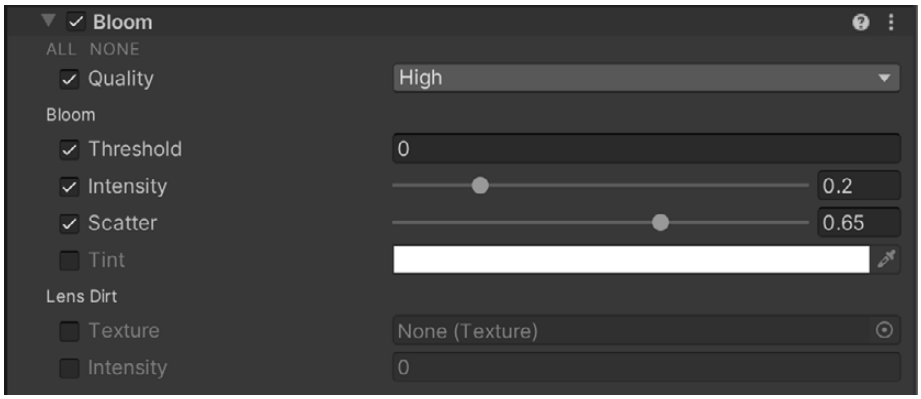
Shadows, Midtones, Highlights

The Shadows, Midtones, Highlights override separately controls the tonal and color range for shadows, midtones, and highlights of the render. Activate each trackball to affect the respective part of the image. Then, use the **Shadow** and **Highlight Limits** to prevent clipping or pushing the color correction too far.

Shadows, Midtones, Highlights

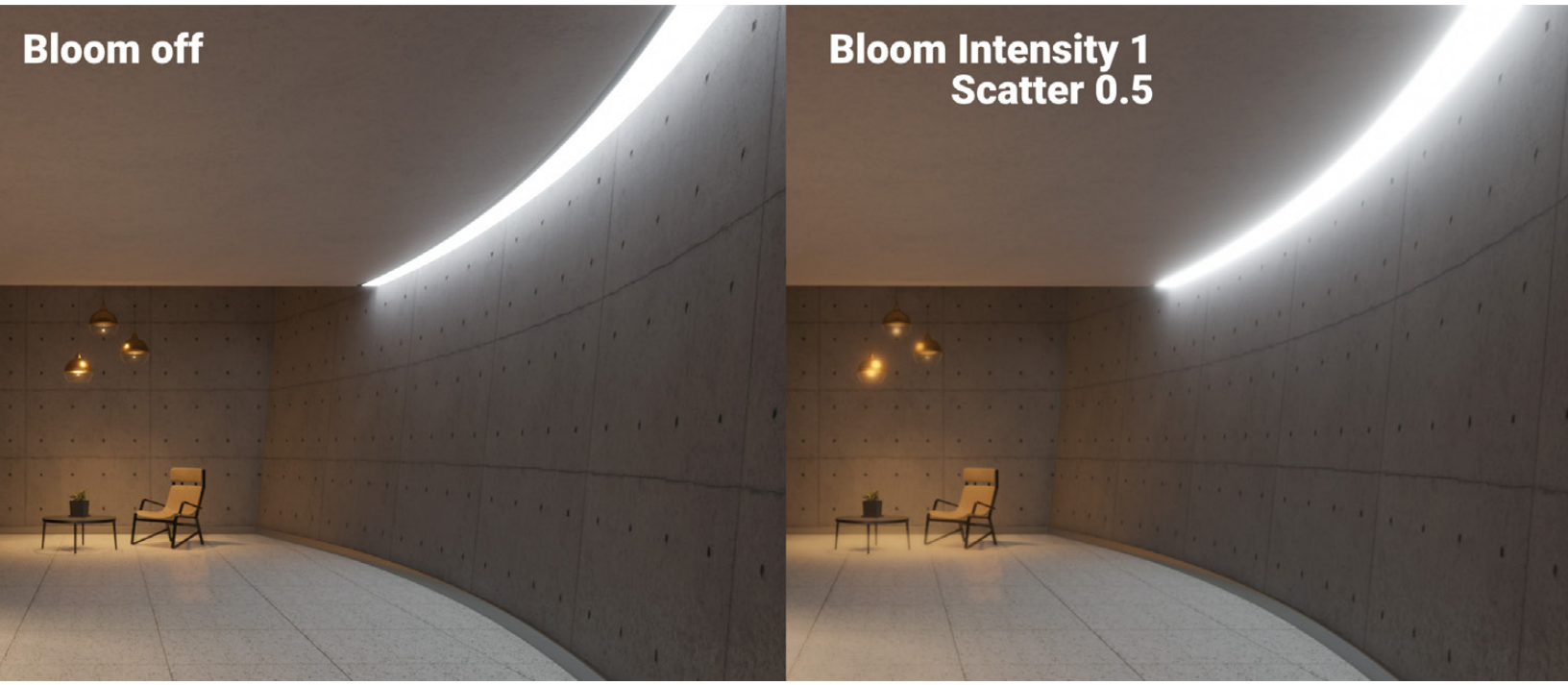
Bloom

Bloom creates the effect of light bleeding around the light source. This conveys the impression that the light source is intensely bright and overwhelming the camera.



Bloom override

Adjust the **Intensity** and **Scatter** to adjust the Bloom's size and brightness. **Lens Dirt** applies a texture of smudges or dust to diffract the Bloom effect.

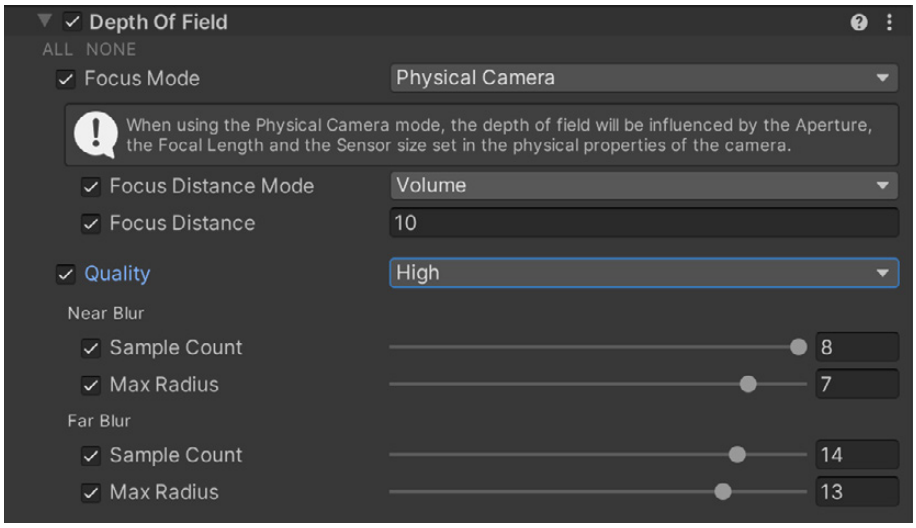


The effect of Bloom

Depth of Field

Depth of Field simulates the focus properties of a real camera lens. Objects nearer or farther from the camera's focus distance appear to blur.

When Depth of Field is active, an out-of-focus blur effect called a bokeh can appear around a bright area of the image. Modify the camera aperture's shape to change the appearance of the bokeh (see Additional Physical Camera Parameters above).



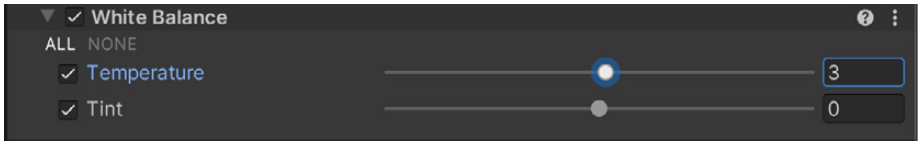
Depth of Field override

Depth of Field simulates the focus distance of real cameras.

White Balance

The White Balance override adjusts a Scene's color so that the color white is correctly rendered in the final image. You could also push the **Temperature** to shift between yellow (warmer) and blue (cooler). **Tint** adjusts the color cast between green and magenta.

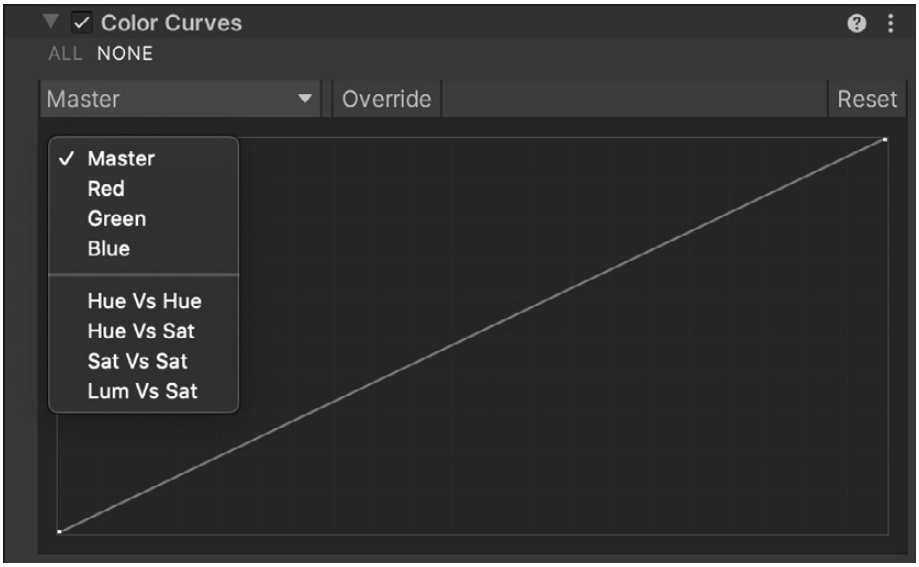
In the HDRP Sample Project, the local Volumes include White Balance overrides for each room.



White Balance

Color Curves

Grading curves allow you to adjust specific ranges in hue, saturation, or luminosity. Select one of the eight available graphs to remap your color and contrast.



Color Curves

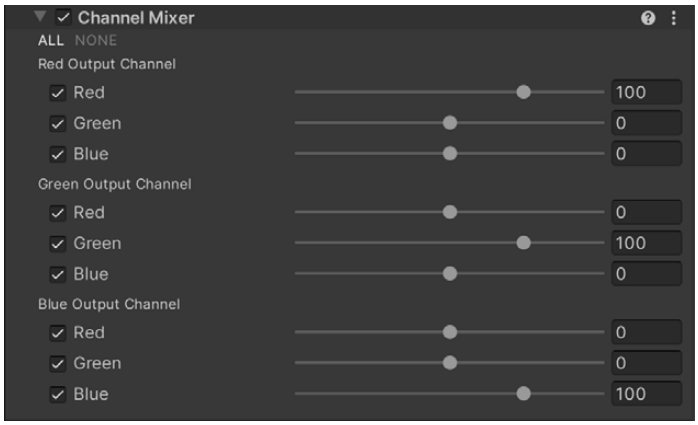
Color Adjustments

Use this effect to tweak the overall tone, brightness, hue, and contrast of the final rendered image.

Color Adjustments

Channel Mixer

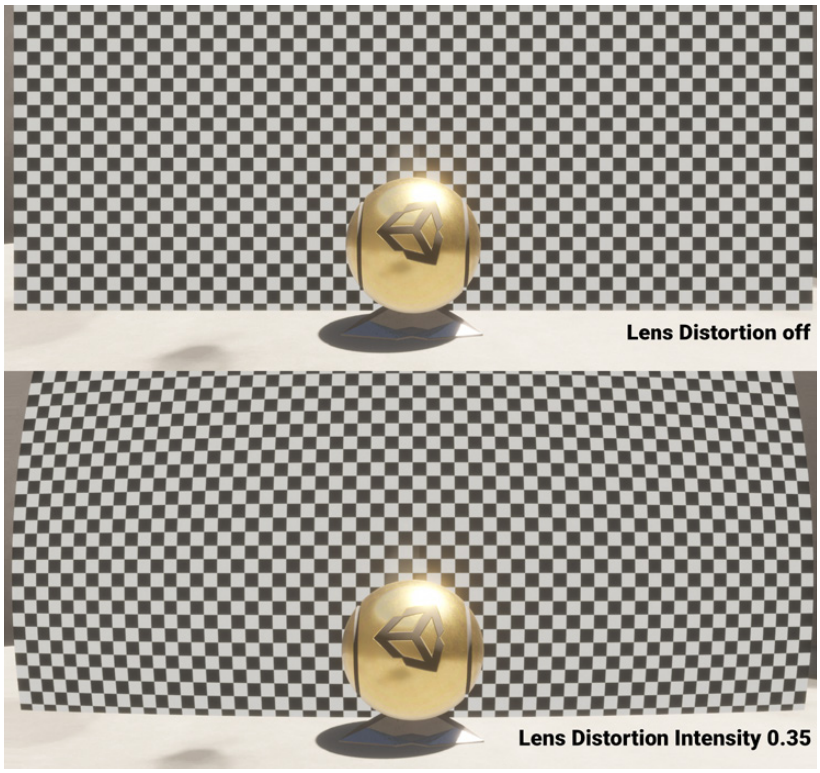
The Channel Mixer lets one color channel impact the “mix” of another. Select an RGB output, then adjust the influence of one of the inputs. For example, dialing up the Green influence in the Red Output Channel will tint all green areas of the image with a reddish hue.



Channel Mixer

Lens Distortion

Lens Distortion simulates radial patterns that arise from imperfections in the manufacture of real-world lenses. This results in straight lines appearing slightly bowed or bent, especially with zoom or wide-angle lenses.



Lens Distortion warps the image in a radial pattern.

Vignette

Vignette imitates an effect from practical photography, where the corners of the image darken and/or desaturate. This can occur with wide-angle lenses or result from an appliance (a lens hood or stacked filter rings) blocking the light. This effect can also be used to draw the attention of the viewer to the center of the screen.

A Vignette darkens the edges of the frame.

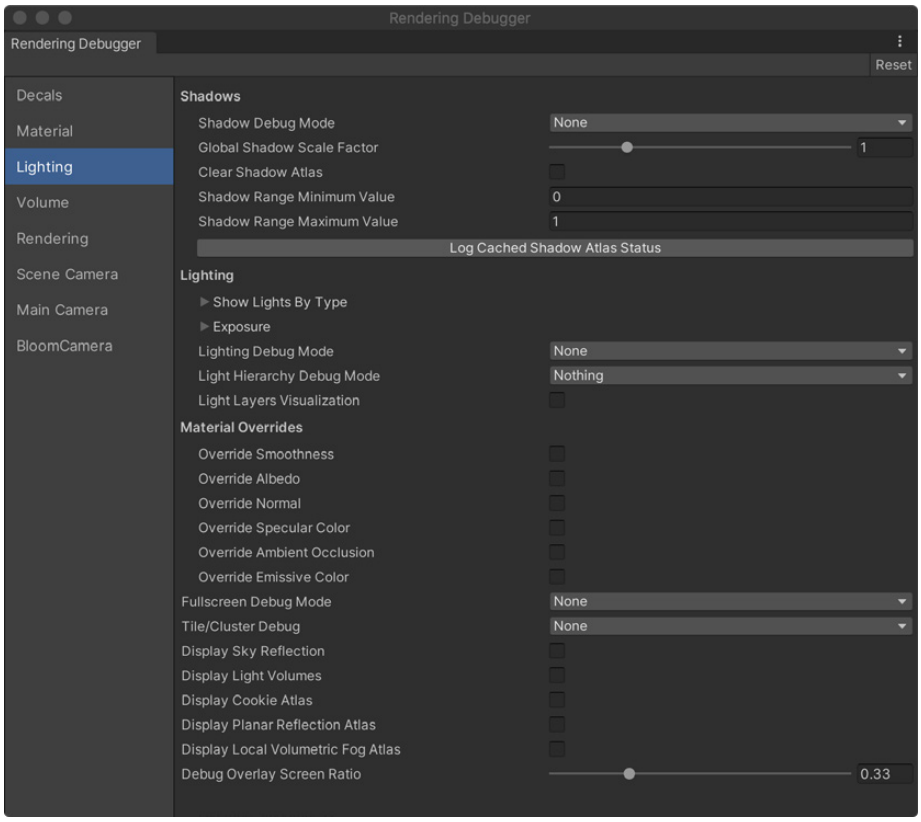
Motion Blur

Real-world objects appear to streak or blur in a resulting image when they move faster than the camera exposure time. The Motion Blur override simulates that effect.

To minimize performance cost, reduce the Sample Count, increase the Minimum Velocity, and decrease the Maximum Velocity. You can also reduce the Camera Clamp Mode parameters under the Additional Properties.

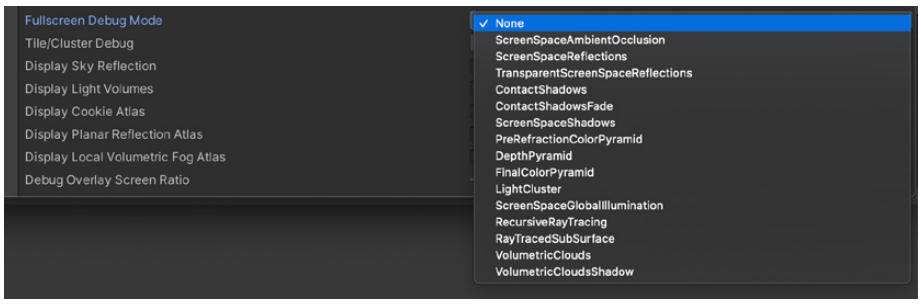
Rendering Debugger

The Rendering Debugger window (**Window > Analysis > Rendering Debugger**) contains debugging and visualization tools specific to the Scriptable Render Pipeline. The left side is organized by category. Each panel allows you to isolate issues with lighting, materials, volumes, cameras, and so on.



Rendering Debugger

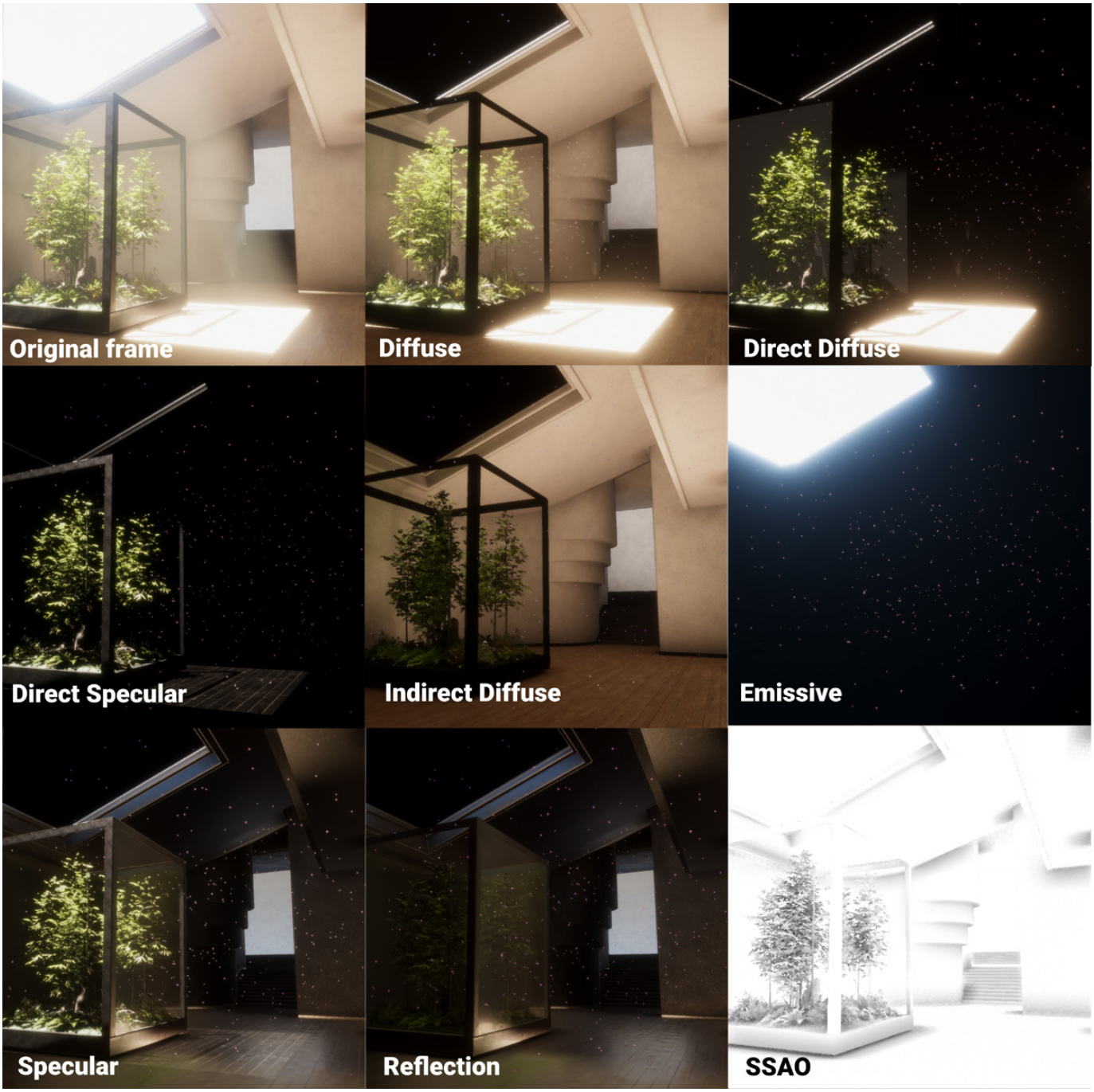
The Debugger can help you troubleshoot a specific rendering pass. On the Lighting panel, you can enter **Fullscreen Debug Mode** and choose features to debug.



Fullscreen Debug Mode options

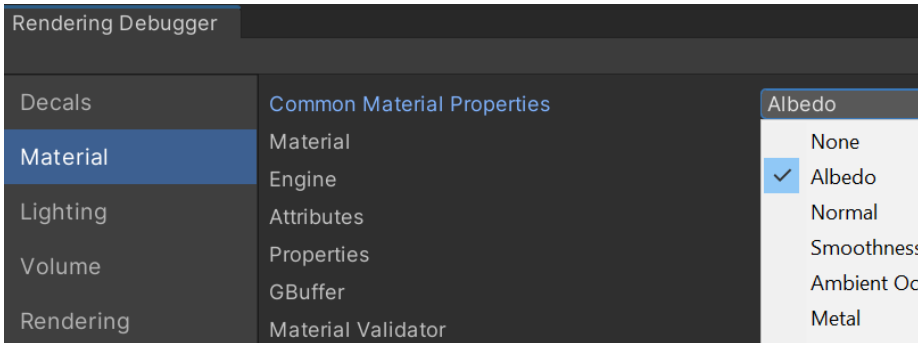
These Debug modes let you play “pixel detective” and identify the source of a specific lighting or shading issue. The panels on the left can show you vital statistics from your cameras, Materials, Volumes, and so on, to help optimize your render.

With the fullscreen Debug mode active, the Scene and Game views switch to a temporary visualization of a specific feature. This can serve as a useful diagnostic.

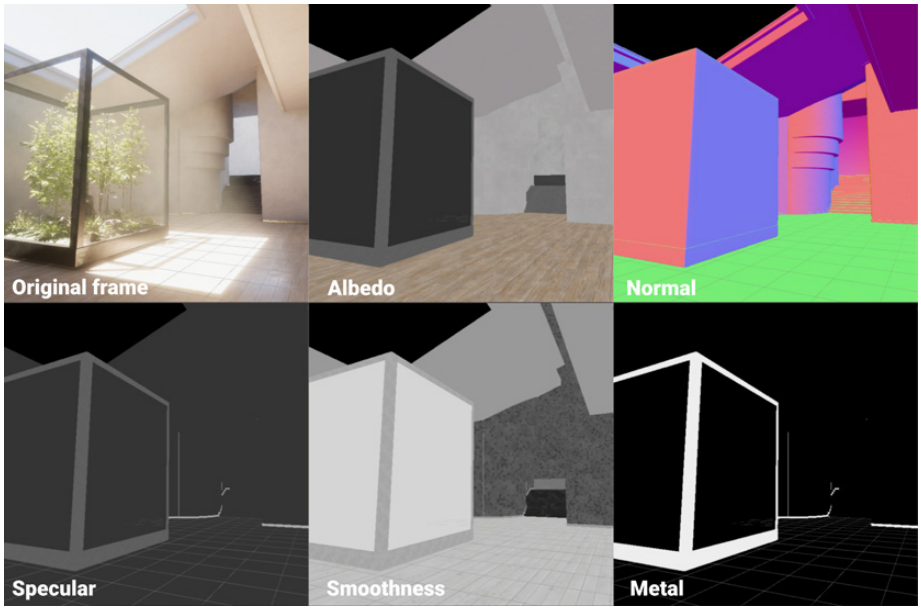


Lighting Debug Mode or Fullscreen Debug Mode can help you understand the sources of illumination in your scene.

You can also debug several common material properties. On the Material screen, select from the Common Material Properties: albedo, normal, smoothness, specular, and so on.



Common Material Properties



Use the Render Pipeline Debugger to troubleshoot materials.

See the Render Pipeline Debugger documentation for complete details.

Ray tracing

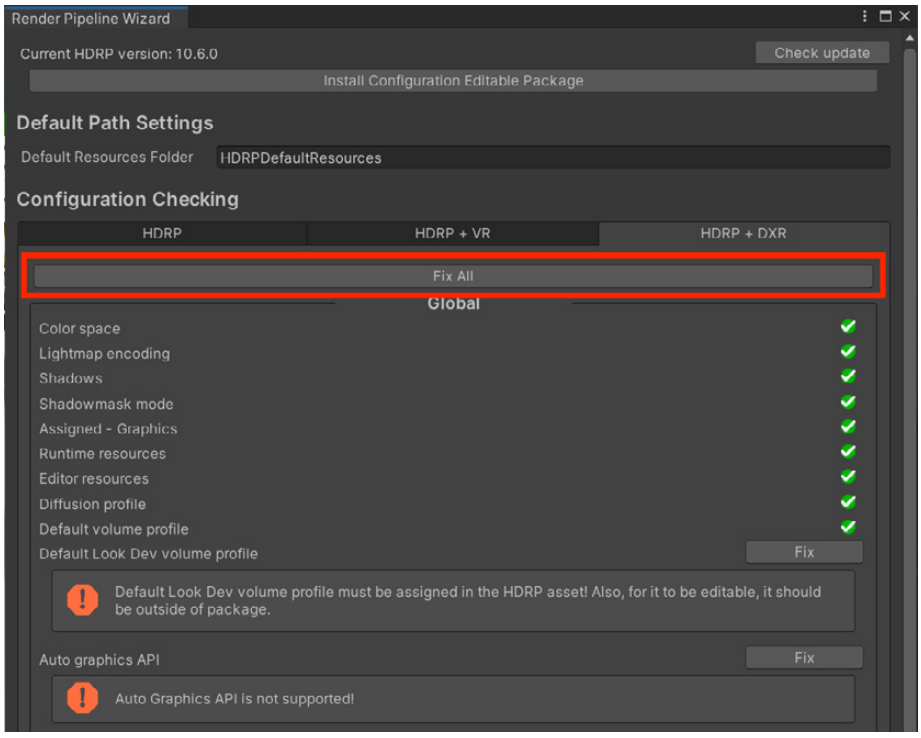
Ray tracing is a technique that can produce more convincing renders than traditional rasterization. While it has historically been expensive to calculate, recent developments in hardware acceleration have now made ray tracing possible for real-time applications.

HDRP includes preview support for ray tracing with select GPU hardware and the DirectX 12 API. See Getting started with ray tracing for a specific list of system requirements.

Setup

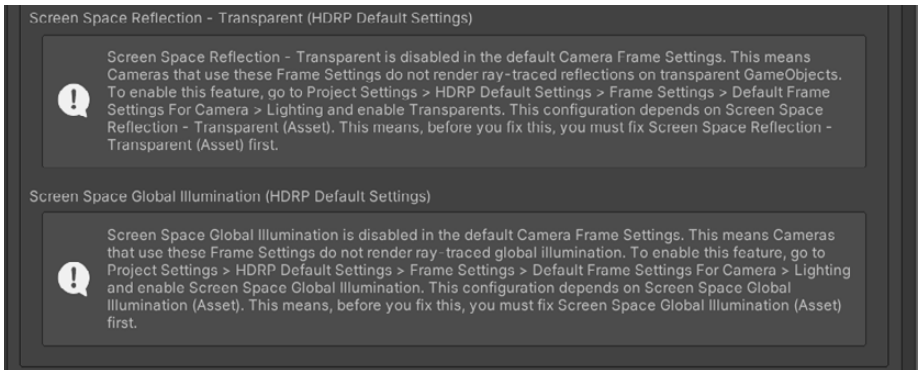
In order to enable ray tracing (in Preview), you need to change the default graphics API of your HDRP project to DirectX 12.

Open the **Render Pipeline Wizard** (**Window > Render Pipeline > HD Render Pipeline Wizard**).[2] Click **Fix All** in the **HDRP + DXR** tab, then restart the Editor. Follow the Pipeline Wizard prompts to activate any disabled features.



Enable ray tracing in the Render Pipeline Wizard.

[2] In Unity 2021, the HDRP Wizard is located under Window > Rendering > HDRP Wizard



Follow the instructions to fix any disabled features.

You can also set up ray tracing manually.

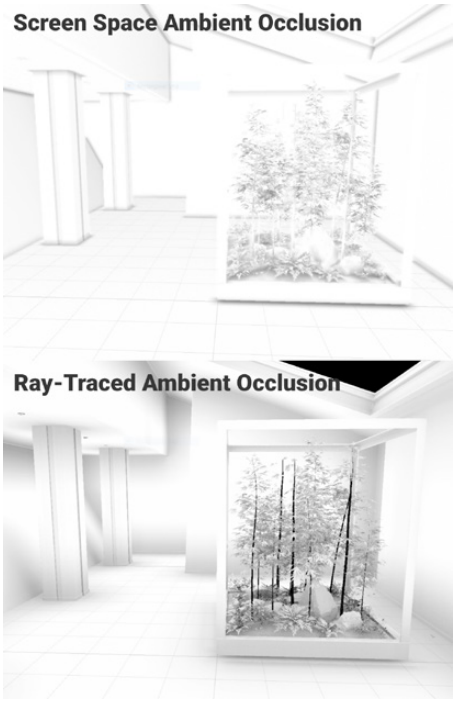
Once you enable ray tracing for your project, check that your **HDRP Global** or **Camera Frame Settings** also has ray tracing activated. Make sure you are using a compatible 64-bit architecture in your **Build Settings** and validate your scene objects from **Edit > Rendering > Check Scene Content for HDRP Ray Tracing**.

Overrides

Ray tracing adds some new Volume overrides and enhances many of the existing ones in HDRP:

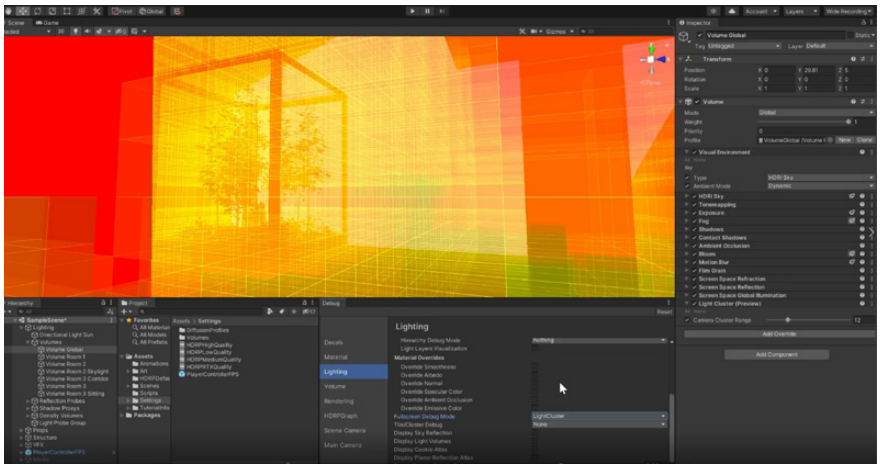
— **Ambient Occlusion**: Ray-Traced Ambient Occlusion replaces its screen-space counterpart. Unlike SSAO, Ray-Traced Ambient Occlusion allows you to use off-screen geometry to generate the occlusion. This way, the effect does not disappear or become inaccurate toward the edge of frame.

— **Light Clusters**: Ray tracing divides your Scene into a grid of 3D cells. HDRP uses these Light Clusters to determine the local lighting whenever a ray hits a surface. It can then compute light bounces for certain effects (Ray-Traced Reflections, Ray-Traced Global Illumination, and so on).



Screen Space Ambient Occlusion vs Ray-Traced Ambient Occlusion

Use **HDRP Debug** mode to visualize the Light Clusters as a diagnostic. Light clusters highlighted in red indicate where the light count has reached the **Maximum Lights per Cel**l in the HDRP asset. Adjust this setting to reduce unwanted light leaking or artifacts.

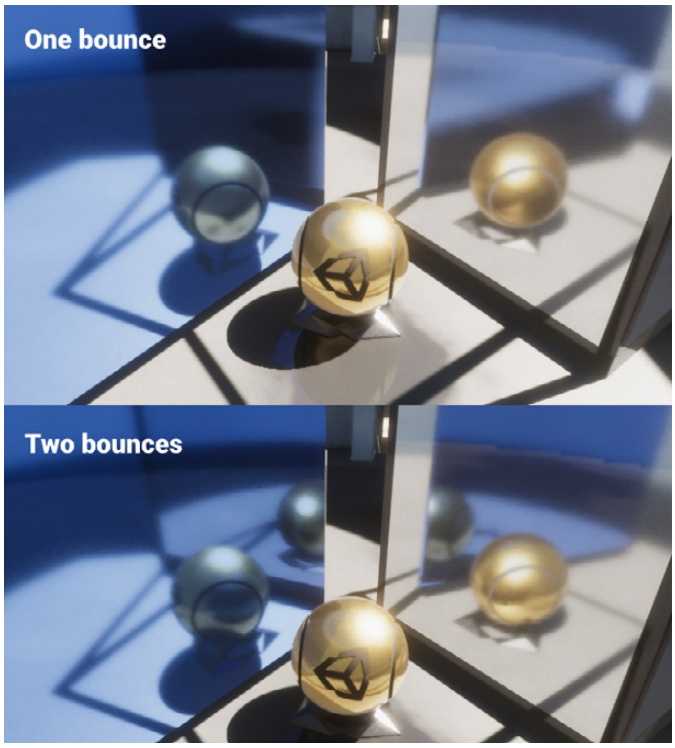


Ray tracing light clusters in Debug mode

— **Global Illumination**: This is an alternative to Screen Space Global Illumination and Light Probes for simulating bounced, indirect lighting. Ray-Traced Global Illumination calculates in real time, and it allows you to avoid the lengthy offline process of baking lightmaps while yielding comparable results.

Use the Quality setting for complex interior environments that benefit from multiple bounces and samples. Performance mode (limited to one sample and one bounce) works well for exteriors, where the lighting mostly comes from the primary directional light.

Ray-traced Global Illumination shows bounced lighting in real time.



Ray-traced Reflections render within the smooth, mirror-like surfaces.

— **Reflections**: With ray-traced reflections, you can achieve higher quality reflections than using Reflection Probes or Screen Space Reflections. Off-screen meshes appear correctly in the resulting reflections.

  Adjust the **Minimum Smoothness** and **Smoothness Fade Start** values to modify the threshold that smooth surfaces receive ray-traced reflections. Increase the **Bounces** if necessary (e.g., two mirrors reflecting each other), but be aware of the performance cost.

— **Shadows**: Ray-traced shadows for directional, point, spot, and rectangle area lights can replace shadow maps from any opaque GameObjects. Directional lights can also cast ray-traced shadows from transparent or translucent GameObjects.

  Ray tracing allows for Percentage-Closer Soft Shadows (PCSS), where shadows soften as their distance from the caster increases. This produces very natural-looking shadows.

  HDRP's directional lights can also generate semi-transparent, colored shadows. In this example, a glass surface casts a realistically tinted shadow on the floor.

— Watch Activate ray tracing with HDRP for a walkthrough of the High Definition Render Pipeline's ray-tracing features in Preview. See the ray tracing documentation on the HDRP microsite for more information.

Ray-traced shadows soften as they fall farther from the caster to achieve a different effect than with shadow mapping.

More resources

— For a video introduction to HDRP's features, we recommend watching Achieving high-fidelity graphics with HDRP to accompany this guide.

— If you're migrating from the Built-In Render Pipeline, see this chart for a detailed feature comparison between the two render pipelines.

— The HDRP documentation includes a complete breakdown of every feature in the pipeline.

Next steps:

We hope this guide inspires you to try HDRP for your next project.

Remember that the 3D Sample Project is available from the Unity Hub when you want to explore further. Be sure to check out the additional resources listed below, and you can always find tips on the Unity Blog or HDRP community forum.

At Unity, we want to empower artists and developers with the best tools to build real-time content. Lighting is both an art and a science – and where these meet, it's a little bit like magic.


Unity®