Buto

Volumetric Lighting and Fog for Unity URP
A Game-Ready Asset by OccaSoftware

Need Help?

If you run into any issues or have any questions, please don't hesitate to contact me by email, on Twitter, or on Discord.
occasoftware@gmail.com
@occasoftware
OccaSoftware on Discord

Table of Contents

Introduction

Buto enables you to easily add real-time stylized volumetric fog to your scene. One Render Feature, One Material. That’s it.

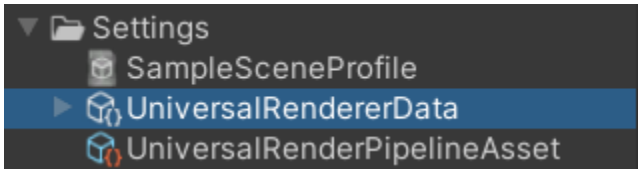
Designed for Unity 2021.3 LTS Universal Render Pipeline (URP).

Features

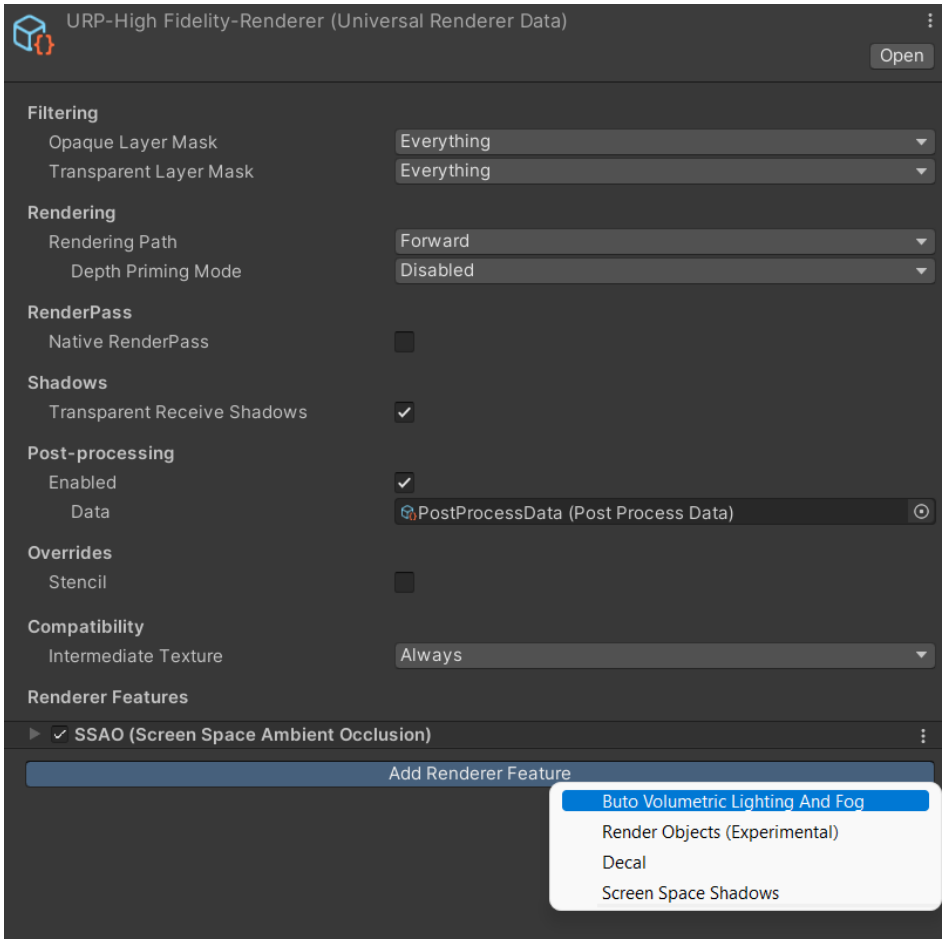
1. Physically-based volumetric lighting and fog generates an accurate simulation of atmospheric particles.
2. Optimized, high-performance rendering at half scale resolution with intelligent depth-aware upscaling.
3. Built-in volumetric noise gives depth and texture to the particle distribution.
4. Particle density exponentially decreases over height resulting in atmospheric height fog.
5. Analytic height fog is rendered behind the nearby ray-marched volumetric fog to guarantee long-range visual consistency.
6. Distance-based Color Ramps, Color Ramp influence, and Light and Shadow intensity sliders give you creative stylized fog options.

First Setup

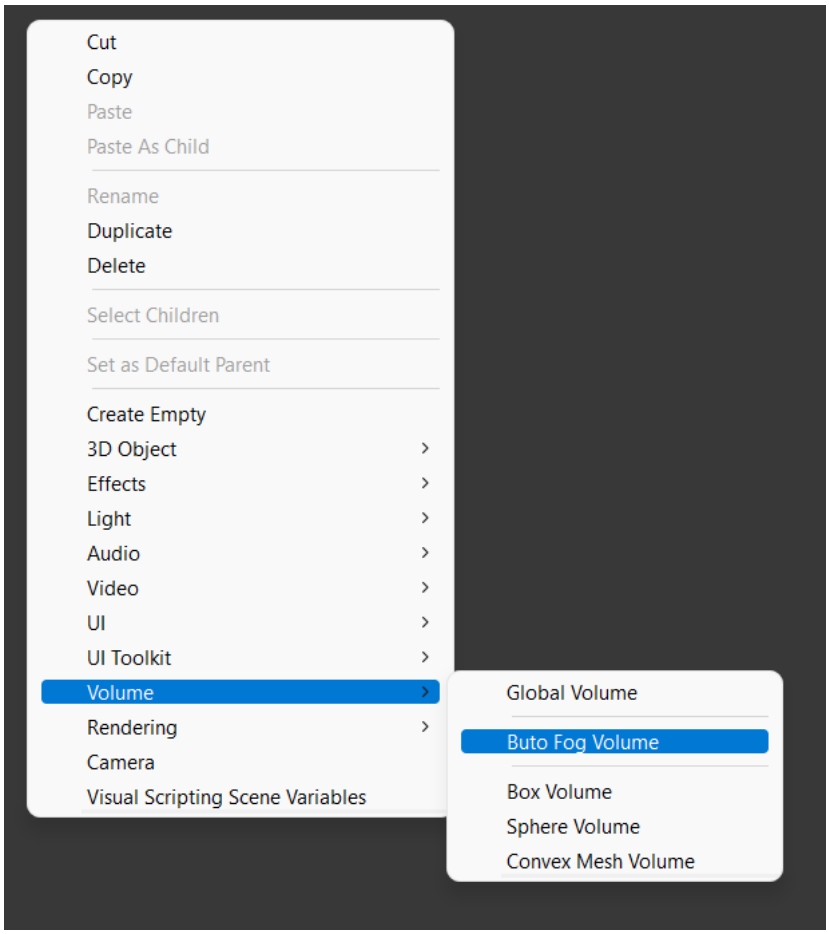
Navigate to your Universal Renderer Data asset. Click on it.



In the Universal Renderer Data Inspector, click “Add Renderer Feature” and choose “Buto Volumetric Lighting and Fog”

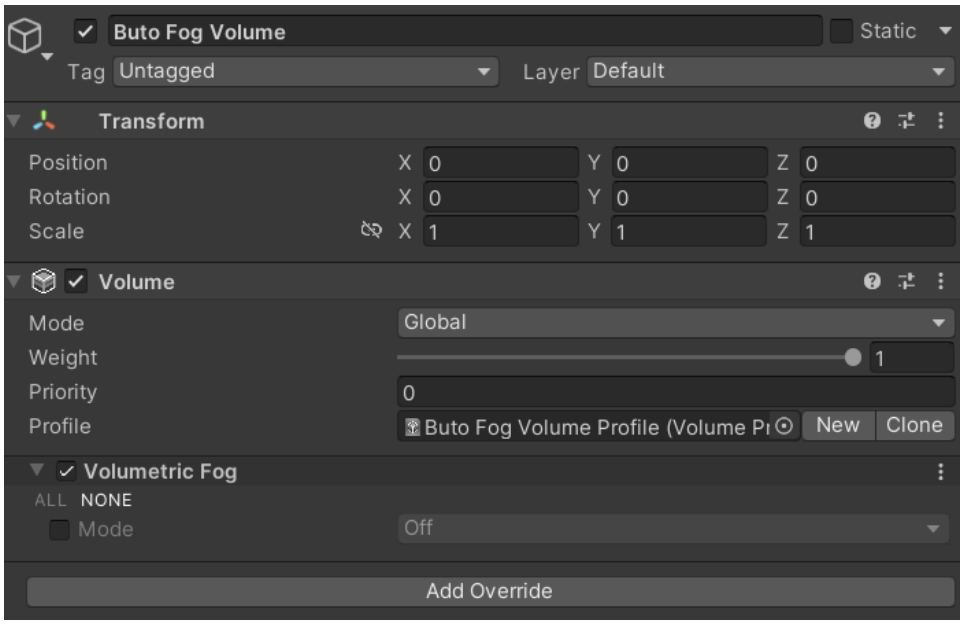


In your Hierarchy, right click, then go to Volume, Buto Fog Volume

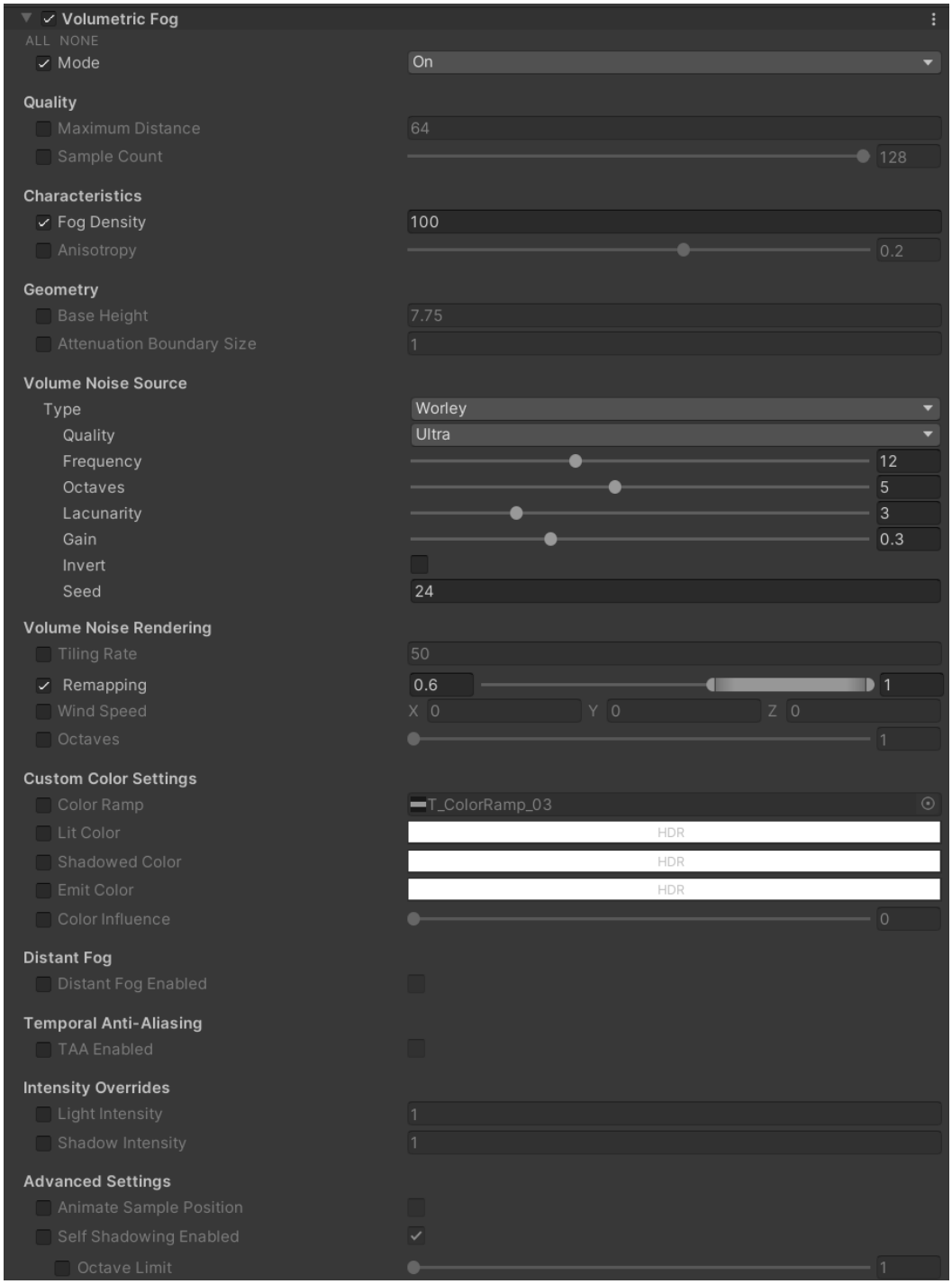


Create a new Volume Profile.

Then click Add Override -> Volumetrics -> Buto Volumetric Fog.



Configure the Volumetric Fog override according to your scene requirements.



On Performance

There are four major parameters that will drive more performant results. They are described here below.

Volume Noise Rendering: Octaves
Increasing octaves will cause Buto to sample the volumetric noise multiple times. This is performance-intensive. Lower your Volume Noise Rendering Octaves for a performance gain.

Volume Noise Source: Quality
A larger 3D Texture requires more time to sample and requires more VRAM. Lower your quality setting for a performance gain.

Sample Count
Increasing sample count will cause Buto to take more samples for a given pixel. Although increasing this does give smoother results, it does have a significant performance impact. Lower your sample count for a performance gain.

Self Shadowing Enabled
Because fog is a dense medium, light rays will bounce away and areas of fog will be shadowed by fog that was in between it and the main directional light source. This does significantly increase the level of realism, but does also have a high performance cost. Disable it for a performance gain.

Self Shadowing Enabled: Octave Limit
By default, the Self Shadowing effect is limited to a single Volume Noise Rendering Octave. You can override this setting for highly realistic results, but it does have a **very high performance cost.** Lower this setting for a performance gain.

Distant Fog
Distant Fog is calculated in a second pass after the Volumetric Fog. It is a minor performance cost. Disable it for a minor performance boost.

Temporal Anti-Aliasing
Temporal Anti-Aliasing can give you smoother results under low camera motion. However, it does require Render Texture tracking and merging. This can cause a

performance drop if you do not also lower your Sample Count to get the benefits of the TAA pass. Disable TAA without raising your Sample Count for a performance boost.

On Volume Noise Sources

What is a Volume Noise Source

Buto enables you to use Volumetric Noise, or 3D Noise, in order to define the structure of the fog in your scene. This noise is tiled repeatedly across a given cube domain. In effect, it describes the density of the fog in 3-dimensional space. The base fog value is multiplied by the 3D Noise value sampled at a given point in space. Therefore, when the 3D Noise is close to a value of 0, the fog will also be close to 0. In the same way, when the 3D Noise is close to a value of 1, the fog will also be close to the base fog value.

You can choose to go without a 3D Noise texture by setting your Volume Noise Source as '**None**'.
If you do choose to use a 3D Noise texture, you have two options:

1. You can use your own pre-computed 3D Texture. These can be challenging to compute, but it does give you the most control over the final look of the fog.
2. You can use our built-in noise generation algorithms to create 3D Textures according to a set of input parameters that we have exposed for you.

Using Your Own Pre-Computed Texture

If you choose to use your own pre-computed texture, you will need to render your noise as a 2D Texture Array, then import it to Unity, and then convert it to a Texture3D using a helper script (*not included in Buto*). Buto expects the input Texture to be a 4-channel texture with data in the red, green, blue, and alpha channels. However, Buto automatically resamples this texture data by assigning decreasing importance to each channel and then summing the result together. See the table below for an example of how this works.

Channel	Input Value	Sample Weight	Sampled Value
R	.5	0.53	0.265
G	.8	0.27	0.216
B	.3	0.13	0.039
A	.6	0.07	0.042

Then we calculate the final value by summing up the sampled values as follows:
0.265 + 0.216 + 0.039 + 0.042 = 0.562

We use this method rather than using a pre-compressed single-channel texture because it gives better visual results overall and also gives better results when remapping the value range to a smaller, clamped range.

Using Buto's Built-In Noise Generation System

Buto's editor includes a helpful noise generation system that will procedurally create Volumetric Noise, bake it to a 3D Texture, and associate the 3D Texture to the corresponding Unity volume component asset.

As of this writing, Buto's editor tool includes the following Volume Noise Types:

1. Perlin
2. Worley
3. Perlin-Worley
4. Billow
5. Curl

You can provide a **Seed** value to randomize the results. The results for each noise type will be constant for a given seed.

You can define the **Quality** setting to limit the size of the texture asset, and you can **Invert** the results to achieve double the number of base noise types.

Using the **Frequency** slider allows you to describe how many noise cells will be contained within a given texture. The **Octaves** slider enables you to layer the noise with

itself, and the **Lacunarity** and **Gain** parameters describe the rate of frequency change from layer to layer and the rate of strength loss from layer to layer.

This is a new system, so you may encounter some unexpected issues. If you do, please contact me at occasoftware@gmail.com so that I can help troubleshoot and resolve them.

Be aware that these pre-baked 3D Textures may be generated on-the-fly during runtime, which can cause your game to freeze while the texture is generated and saved. In order to avoid these freezes during gameplay, I recommend that you load the textures ahead of time. One method of doing this is by triggering each fog volume during your load scene. If you continuously encounter these freezes and pre-loading the volumes doesn't work or is otherwise inconvenient, please contact me at occasoftware@gmail.com so that we can work together to find a solution that works for you.

Additional Fog Types

You may want different types of fog depending on your scene. Simply add a Global Volume to the scene and configure accordingly. You can save Volumetric Fog presets as Volume Profiles.

Note that a single scene can contain any number of Buto Fog Volumes. By using the built-in Unity Volume system, you can configure local Box and Sphere volumes as well.

Using Volumetric Point Lights and Fog Density Masks

Configuring Volumetric Point Lights

1. On any Game Object, add the Buto Light component
2. Buto will now treat this Game Object as a Volumetric Point Light
3. You can add a Light component to the same Game Object and have the Buto Light inherit the characteristics of the Light component. The easiest method is as follows:
    a. Toggle on the property, Inherit Data From Light Component in the Buto Light component.

    b. Click the button, Check or Add Light Component. A light component will be added if one is not already present on the Game Object.
    c. Configure the Light component's Intensity and Color.
4. Note that Volumetric Point Lights do not account for shadow attenuation
5. When more than 8 Buto Lights are available in the scene, Buto will automatically cull all but the closest 8 lights. Note that this incurs a slight performance cost.

Configuring Fog Density Masks

1. On any Game Object, add the Fog Density Mask component
2. Buto will now treat this Game Object as a Fog Density Mask
3. You can have as many as 8 Fog Density Masks render concurrently in a scene.
4. When more than 8 Fog Density Masks are available in the scene, Buto will automatically cull all but the closest 8 masks. Note that this incurs a slight performance cost.
5. You can set Fog Density Masks to act multiplicatively with a global Fog, or to act as the only fog sources in your scene.

When the Fog Density Mask is set to **Multiplicative**, the fog within the **Radius** of the mask is multiplied by the **Density Multiplier** according to the **Falloff** rate.

When the Fog Density Mask is set to **Exclusive**, Buto will only render fog that is within the **Radius** of the mask. When there is **more than one Exclusive** Fog Density Mask active in the scene at the same time, Buto will render fog that is within the **Radius** of **either mask**.

Custom Color Ramps

Buto enables you to include a Custom Color Ramp to define how the fog colors are defined over distance. This Custom Color Ramp is combined multiplicatively with the simple fog color settings included in the asset. Although the Custom Color Ramp does not interpolate between Unity's Volume Components, the simple fog color settings will. In this way, you can at least define grayscale ramps that give you a level of dynamic control when moving between volumes. However, you can always leverage local volumes and disable/enable the fog to avoid sudden discontinuities when switching to new Color Ramps. Below, I'll describe in more detail how to create a custom color ramp.

Creating a custom Color Ramp

Your Color Ramp texture can have any dimensions. However, be aware that it is sampled as follows:
X coordinate is selected by the relative distance from the camera to the end of the volumetric fog.
Y coordinate is pre-determined for each color type. For each x coordinate,

- Shadow Color is selected from the center of the bottom third of the image,
- Lit Color is selected from the center of the center third of the image,
- Emission Color is selected from the center of the top third of the image.

For example,

The Color Ramp is sampled using Point filtering. Increase the resolution of the texture in order to leverage smoother gradients. Note that this is typically unnecessary. Buto accumulates fog data gradually over distance, which naturally lends a smooth gradient to the results.

I recommend using Adobe Color to identify color ramps and using GIMP to create it.

If you'd like an easier method to create new Color Ramps, do submit a feature request by email to occasoftware@gmail.com. You can use the following subject line:
Subject: New Feature Request (Buto): Easier Color Ramp Creation

Support

Contact

If you encounter any issues at all, please contact me at occasoftware@gmail.com.
You can also reach out on Discord or Twitter.

Thanks for your support!

- Join the OccaSoftware Newsletter. Be the first to know about new releases, sales, and receive exclusive Unity tips.
- Learn more about OccaSoftware - The Best Assets for Unity.
- Check out our FAQs.
- Read the latest OccaSoftware News.
- Watch our newest release videos and tutorials on Youtube.

Like this asset? We also recommend the following other assets from OccaSoftware.

- Altos - Volumetric Clouds, Skybox, Day Night Cycle, Sun, Moon, and Stars
- Toon Kit - Toon Shader for Unity
- LSPP - Volumetric Lighting, God Rays, and Light Shafts