

VOLUMETRIC LIGHTS

Built-in Pipeline Edition

Quick Start Guide



Contents

Introduction	3
Requirements.....	3
Demo Scene	4
Adding Volumetric Effect to existing lights	5
Adding new Volumetric Lights to the scene.....	6
Settings Description	7
Rendering section	7
Appearance section	8
Dust particles.....	8
Shadow Occlusion.....	9
Other Settings.....	9
Scripting Support.....	10
Accessing properties.....	10
Creating volumetric lights at runtime	10
Performance Tips	11
Orthographic Camera Support	11
Support.....	11

Introduction

Thanks for purchasing!

Volumetric Lights is a powerful, yet lightweight, volumetric lighting solution for Unity that adds realistic ambient to your scenes.

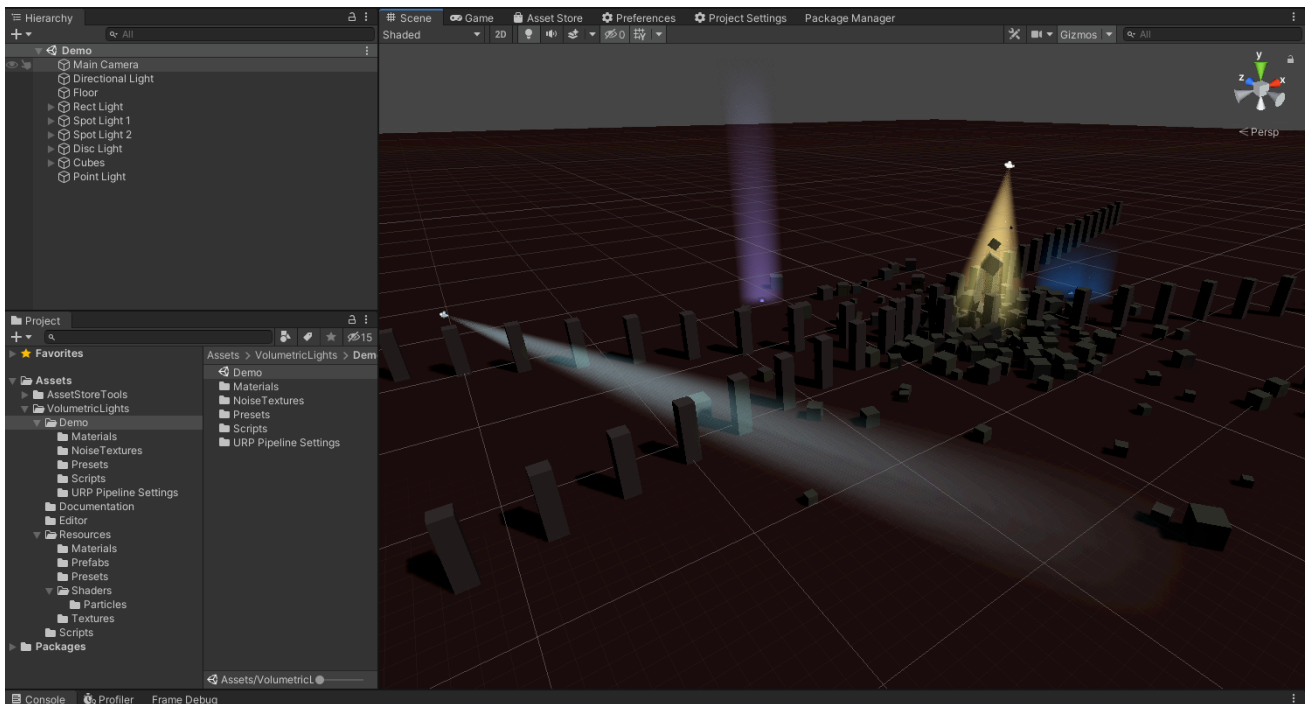
This version of Volumetric Lights works only with built-in pipeline. You'll find a version designed for Universal Rendering Pipeline in the corresponding folder of the asset.

Requirements

Volumetric Lights for built-in pipeline requires Unity 2019.3 or later.

Demo Scene

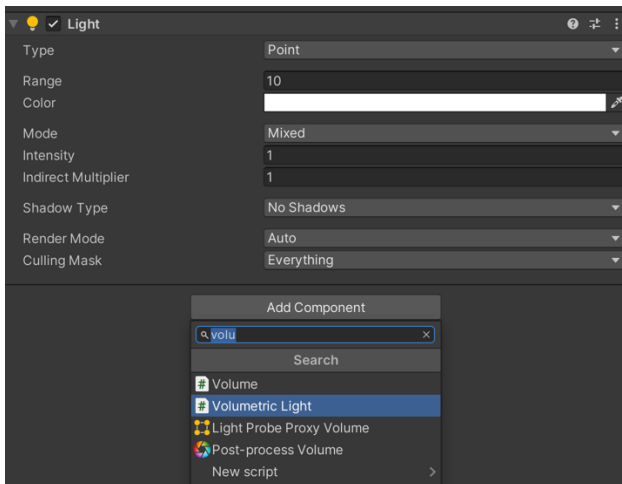
The asset includes a demo scene to let you quickly familiarize with the different features.



Demo running on iPhone 7: https://youtu.be/s_hdefX4P9k

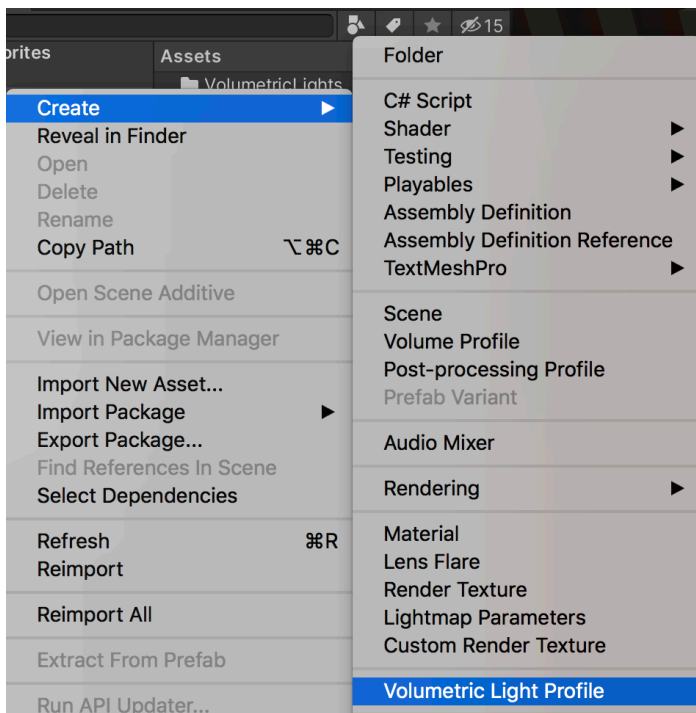
Adding Volumetric Effect to existing lights

Just add the “Volumetric Light” component to any of your lights. Please note that volumetric lights asset is currently compatible with point, spot and area lights.



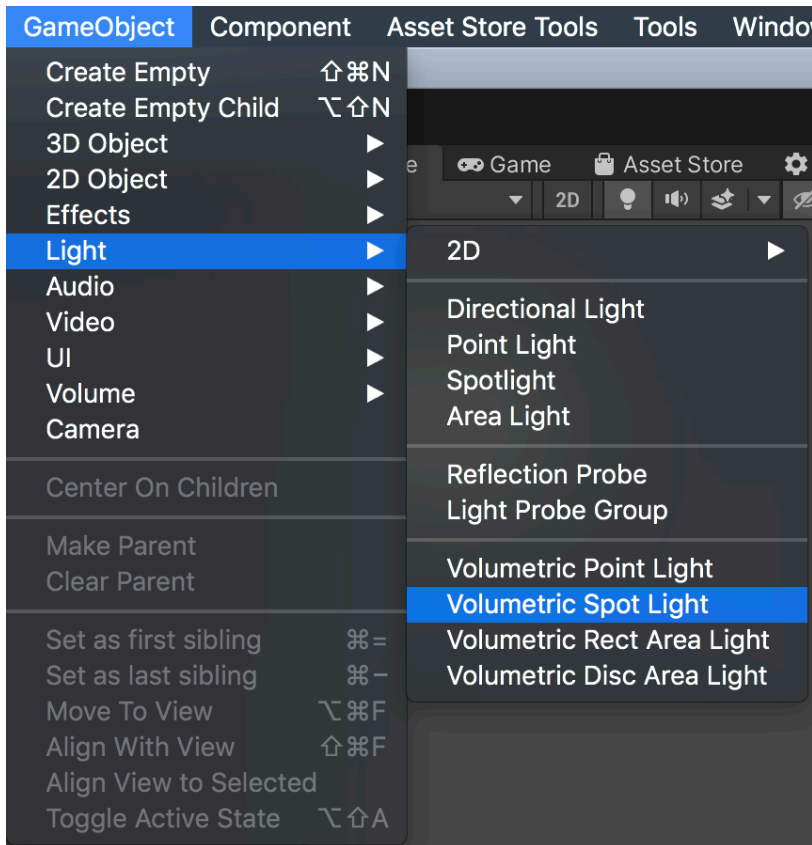
Optional: Create or assign a fog profile

You can create a new volumetric light profile at any moment and assign it to the component. Right click in on the Project panel and select Create -> Volumetric Light Profile:



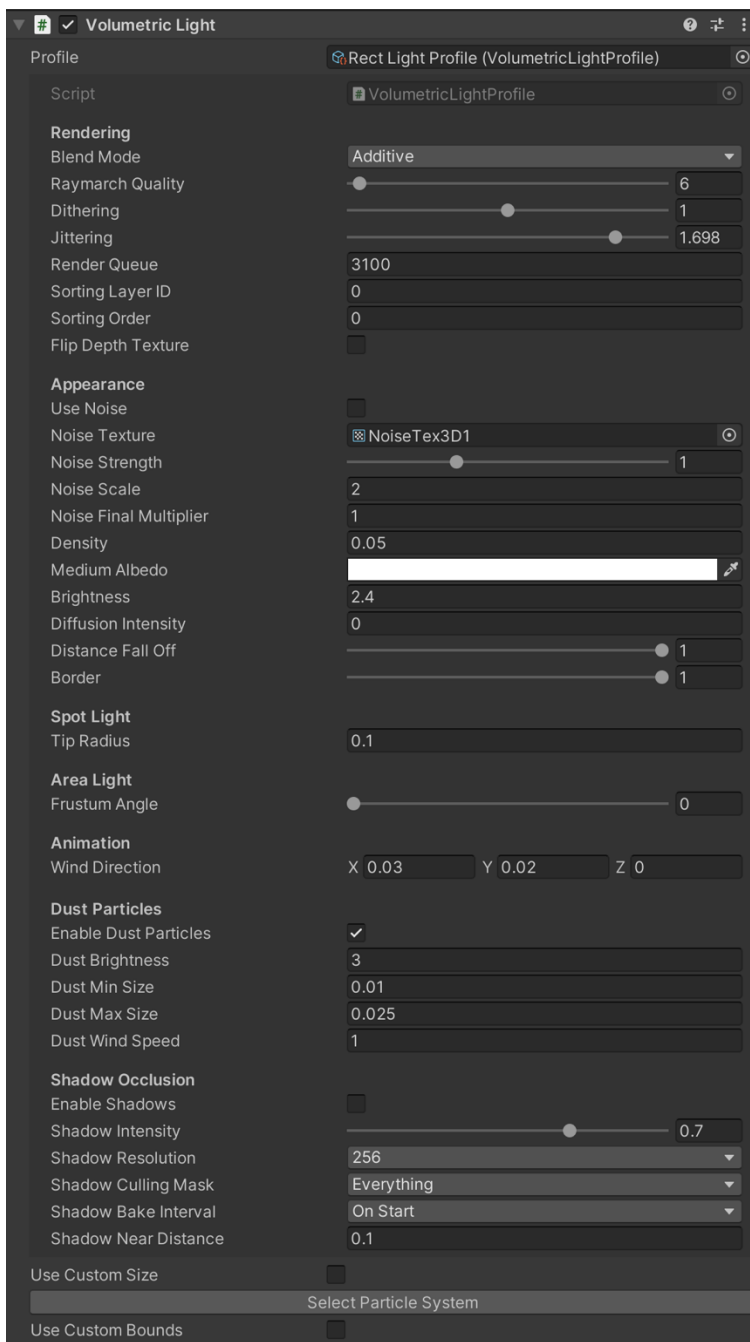
Adding new Volumetric Lights to the scene

You can create a normal light and add the “Volumetric Light” script to it as described in the previous section or you can just select the menu option **GameObject -> Light -> Volumetric *** Light** as shown in the picture below:



Settings Description

The following options are visible when you add a Volumetric Lights component:



Profile: this field shows which profile is being used (if any).

Rendering section

- **Blend Mode:** blend or additive. Blend does the traditional alpha blending with the background while additive adds the light intensity to the existing pixels.

- **Raymarch Quality:** influences the number of iterations executed in the raymarching loop. The lower value, the better performance.
- **Raymarch Min Step:** determines the minimum step size. Increase to improve performance (accuracy will be reduced as well).
- **Dithering:** reduces banding effect.
- **Jittering:** reduces banding effect (especially useful when shadows are enabled).
- **Use Blue Noise:** enables blue noise for jittering computation. It adds an additional texture fetch but reduces moiré pattern caused by regular jittering algorithm. Use only if you get a noticeable improvement.
- **Render Queue / Sorting Layer / Sorting Order:** the volumetric effect renders in world space in the transparent queue. These settings allow you to customize the order of the rendering.
- **Always On:** enabling this option will force the volumetric effect to be visible regardless of the light enable state. This option is useful to create fake volumetric lights, where only the volumetric effect is visible but the light component itself doesn't add any overhead since you can disable it.

Appearance section

- **Use Noise:** uses an internal noise texture instead of assuming an homogeneous medium.
- **Noise Texture:** the asset will use a default 3D noise texture located in Resources/Textures folder. You can find more noise textures in Demo/NoiseTextures folder. You can also provide your own textures.
- **Density:** density of the medium used during the raymarching loop. The greater value, the opaquer effect. A low-density value used in a very large light space can impact performance as the ray march loop needs to run many steps.
- **Diffusion Intensity:** controls the intensity of the diffusion of the light when looking directly to the source through the medium.
- **Distance Falloff / Border:** controls the falloff of the volume. Border has no effect on point lights.

Dust particles

The component will use a particle system with a custom particle shader that's aware of the lighting volume geometry.

Shadow Occlusion

This option adds shadow support to the effect.

- **Shadow Intensity:** amount of light cancelling due to shadows.
- **Shadow Resolution:** resolution of the shadow map. Usually you don't need a very high value to create a good effect. Try to use the lower texture size as possible.
- **Shadow Culling Mask:** specifies which objects cause shadows.
- **Shadow Bake Interval:** specifies if shadow map is computed during start only or on every frame. Note that if the light is moved or rotated, the shadow map will be automatically recomputed.
- **Shadow Near Distance:** specifies the near clip plane of the camera used to create the shadow map. This setting allows you to skip objects that are too near to the light.
- **Use Default RT Format:** when enabled (by default), Volumetric Lights will use a default render texture format when rendering shadows. If disabled, it will render to a single depth buffer which is a bit faster but can cause trouble with some shaders that use Grab Pass command.

Pro tip! When **Shadow Bake Interval** is set to **OnStart**, you can force shadows update anytime calling `ScheduleShadowCapture()` method of the `VolumetricLight` component attached to the light.

Other Settings

- **Use Custom Size:** let you override light properties like range or size.
- **Use Custom Bounds:** let you customize the boundary of the light effect effectively clipping the volumetric effect. This is useful to clamp the volumetric effect to certain corners for example. When this option is enabled, you will see the actual bounds in the Scene View and you can use the handlers of the cube to adjust it size.

Scripting Support

Accessing properties

All properties shown in the inspector are accessible through the volumetric light component. First you need to get a reference to the main script which can be done easily using:

using VolumetricLights;

...

VolumetricLight vl = lightGameObject.GetComponent<VolumetricLight>;

(where lightGameObject is the gameobject of the light)

Then, you can set any property like density or wind speed/direction:

vl.density = 0.2f;

vl.windDirection = Vector3.right;

etc.

Call **vl.UpdateMaterialProperties()** to issue a refresh after setting the new values.

Creating volumetric lights at runtime

Use this code to create a new volumetric light using code:

VolumetricLight vl = lightGameObject.AddComponent<VolumetricLight>();

Performance Tips

Volumetric Lights uses an extremely optimized ray-marching algorithm to provide “volumetric sense” effect in front of your player. If you need to improve performance, you can try the following options:

- If Shadow Occlusion is used, make sure the shadow culling mask only includes objects that you want to cast shadows. By default, shadow culling mask doesn't include Transparent FX layer as volumetric lights are set to that layer to avoid being rendered again by the occlusion cameras.
- Disable “Use Blue Noise” option.
- Reduce Raymarch Quality: reducing this value will indeed reduce the number of texture fetches per pixel.
- Reduce the range of the volumetric effect by using the “Use Custom Size” and specifying new range or size
- Set Shadow Bake Interval to “On Start”.

Orthographic Camera Support

To enable fully support of orthographic camera, locate and edit the Options.hlsl file and uncomment this line:

```
//#define ORTHO_SUPPORT
```

Support

Please visit kronnect.com for questions, support and more info.

If you have any suggestion to improve the asset, please contact us. We'll gladly consider it.