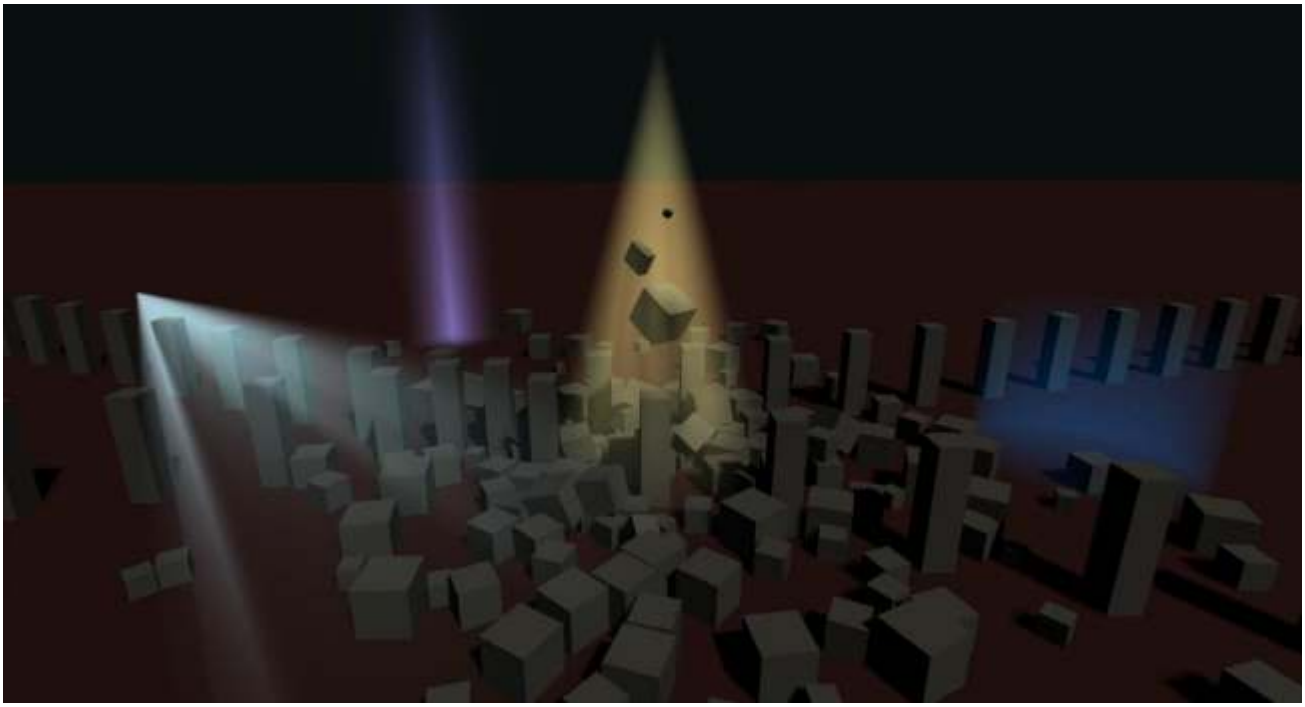


VOLUMETRIC LIGHTS

Universal Rendering Pipeline Edition

Quick Start Guide

Kronnect

Contents

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 has been created from scratch for Universal Rendering Pipeline. It won't work with other pipelines.

Requirements

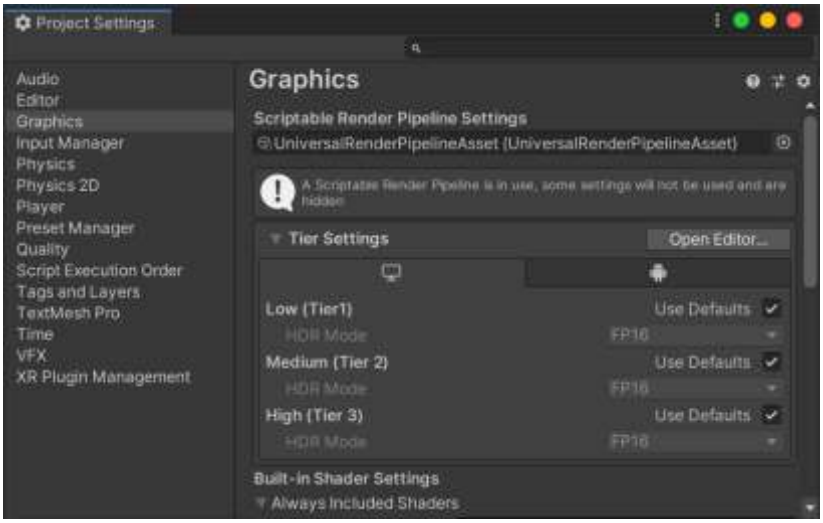
Volumetric Lights for Universal Rendering Pipeline requires:

- Unity 2019.3 or later
- Universal Rendering Pipeline 7.1.8 or later
- Universal Rendering Pipeline asset must have "Depth Texture" enabled.

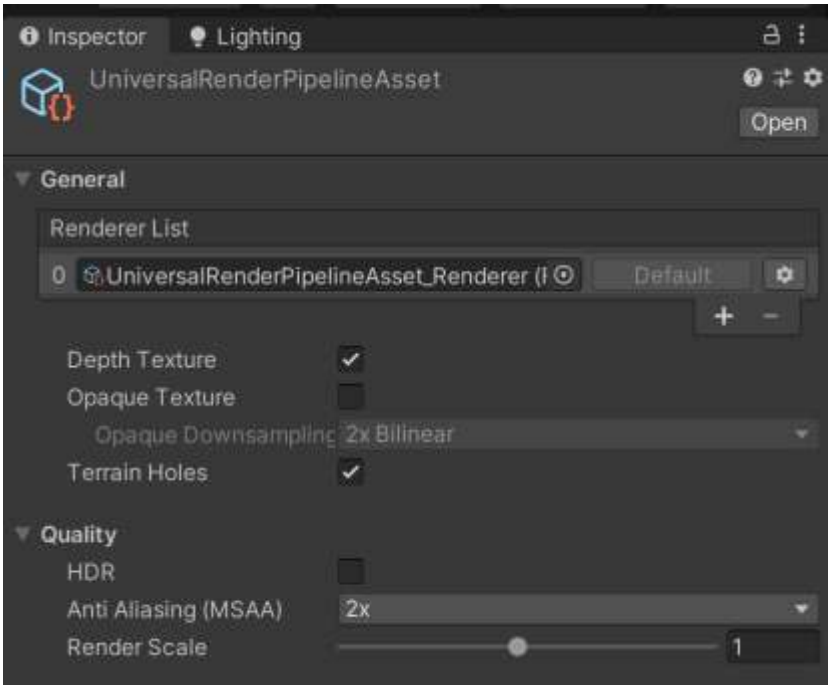
Setup Instructions

Step 1: Install Universal Rendering Pipeline
Go to Windows -> Package Manager. Select Universal RP (7.3.1 or later preferably) and click "Install".

Step 2: Assign the Universal Rendering Pipeline asset
Go to Project Settings -> Graphics and assign a Universal Rendering Pipeline asset. You can use the asset include in the demo folder (Volumetric Lights /Demo /URP Pipeline Settings folder).



Step 3: Configure Universal Rendering Pipeline asset
Double click the Universal Rendering Pipeline asset to show its properties.
Then enable “Depth Texture”. Enabling MSAA is also recommended:

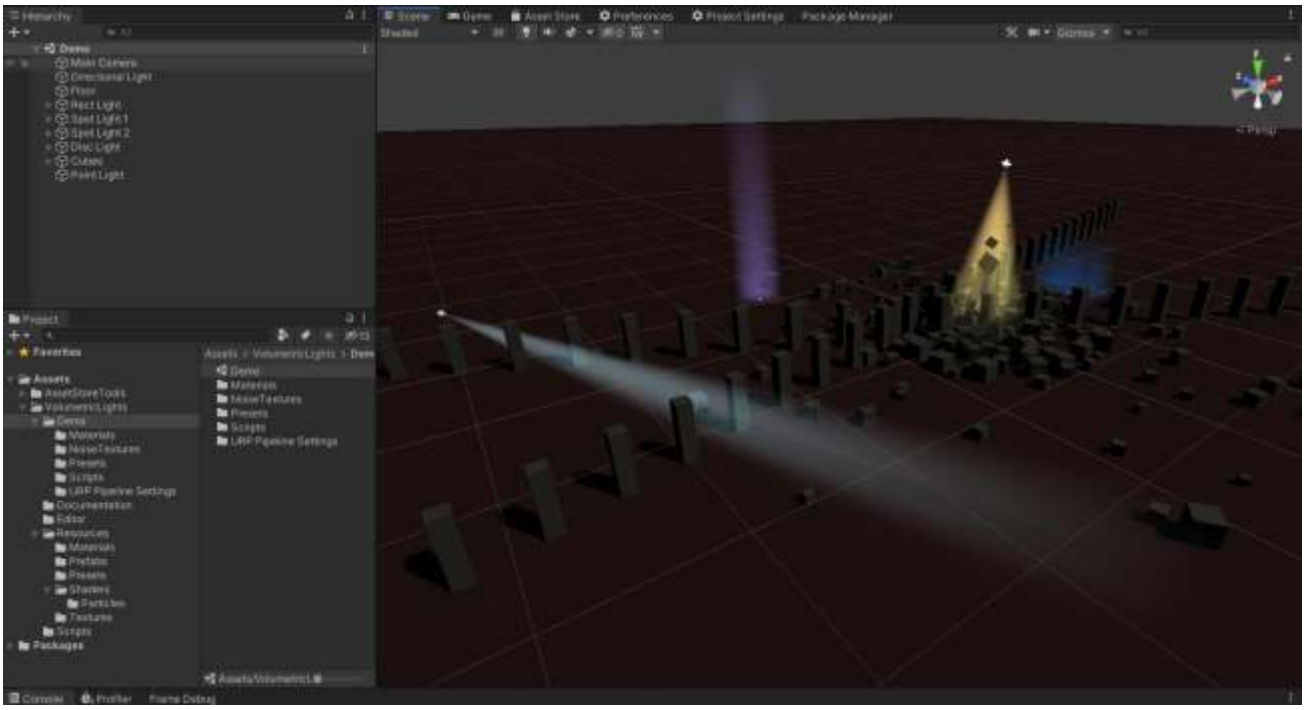


Important note!
Check both “Project Settings / Graphics” and “Project Settings / Quality” sections since you can define URP settings overrides in Quality levels. You need to enable Depth Texture option in all URP settings used in any Quality Level.

Now you can use Volumetric Lights!

Demo Scenes

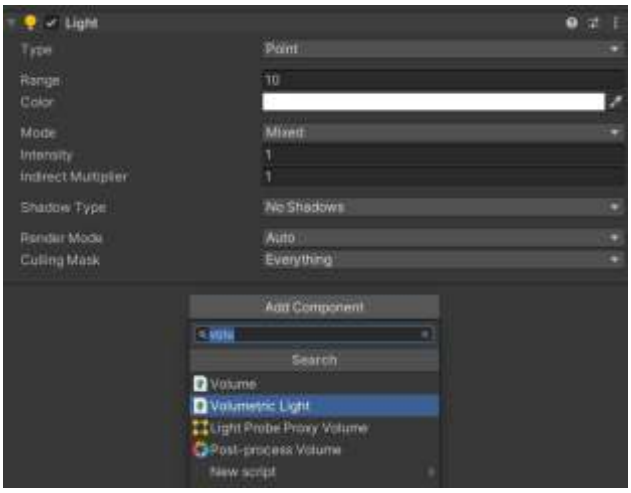
The asset includes two demo scenes to let you quickly familiarize with the different features. Please make sure you have completed the Setup explained in previous section.



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

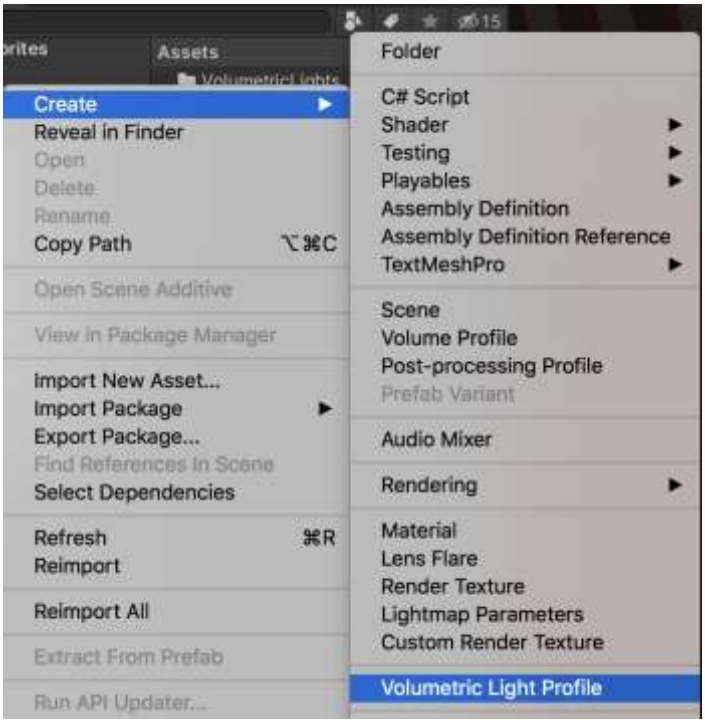
Adding Volumetric Effect to existing lights

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:

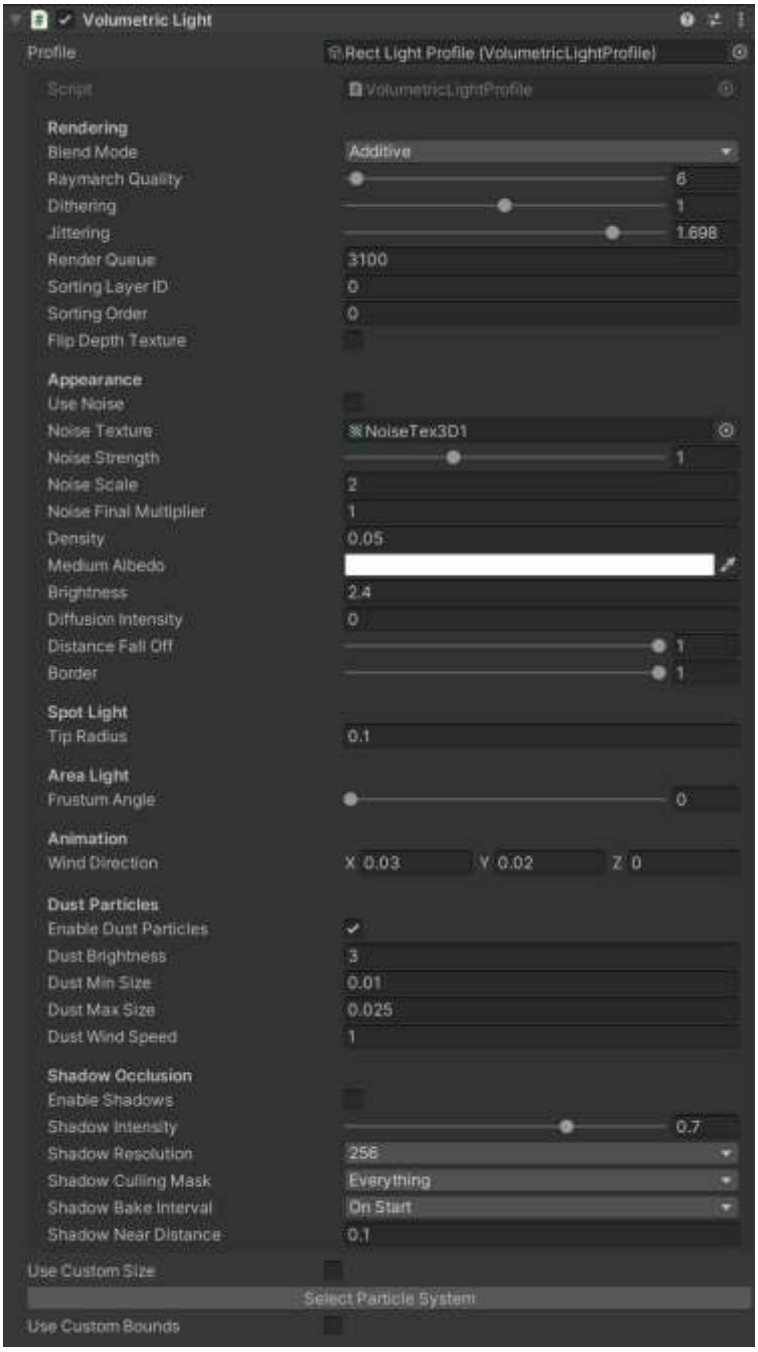


Adding new Volumetric Lights to the scene

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.

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.
- **Min Step Size**: determines the minimum step size. Increase to improve performance (accuracy will be reduced as well).
- **Max Steps**: determines the maximum number of steps or samples taken along each ray. This is a maximum value which can be lowered to improve performance by ensuring there're no rays that take too many samples.
- **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.
- **Flip Depth Texture**: enable only if the light effect doesn't render correctly or clips incorrectly with the existing geometry due to an inverted depth texture bug in URP. This can occur is MSAA if off, HDR is off and Render Scale is set to 1 in the URP asset. If you enable MSAA or HDR, the issue should not occur.
- **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.
- **Cast Direct Light**: this option can only be used in deferred rendering mode (enable it on the camera). It will illuminate the objects reached by the volumetric effect and can be used along other effects like translucency (see the Church demo scene for an example).

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 Bake Mode** (only for point lights): point light shadows are expensive – this option let you choose between computing shadows on a half sphere oriented to player (faster) or a cubemap (slower). If cubemap option is selected, you can choose to spread the rendering across several frames (1 face per frame) or render all 6 faces every frame.
- **Shadow Orientation** (only for point lights): baked shadows captured in a half sphere (180º) for point lights requires orienting the camera. This option let you specify a fixed direction for the alignment of such half sphere or automatically orient it to the player camera.
- **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.
- **Translucency**: when enabled, colors from transparent objects that have the "Volumetric Lights Translucency" script attached will be used in the effect.

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.

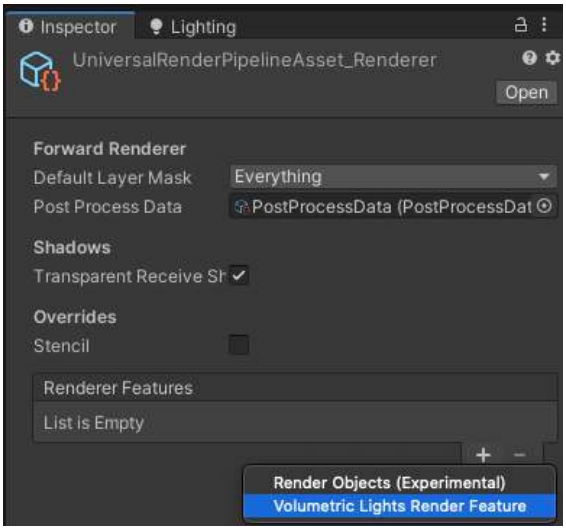
Other Settings

- **Use Custom Size**: let you override light properties like range or size.

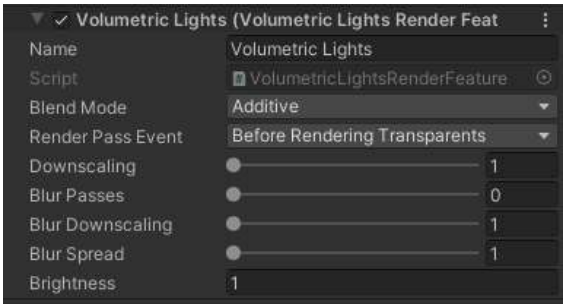
- **Use Custom Bounds**: let you customize the boundary of the light effect effectively clipping the volumetric effect. This is useful to clamp the volumetric effect to certain corners for example. When this option is enabled, you will see the actual bounds in the Scene View and you can use the handlers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

Global Settings

To achieve smoother results, add the Volumetric Lights Render Feature to the Forward Renderer of the URP asset. Note: this feature is not necessary on mobile devices as they use high-density screens.



This feature will render all volumetric lights to an off-screen buffer and perform screen-space blur before composing it to the camera buffer. The following settings are available:



- **Blend Mode**: the blending mode for the final composition.
- **Brightness**: global brightness for the final composition (this value multiplies to the actual brightness of each individual light).
- **Downscaling:** reduces the resolution of the intermediate light buffer. This option will improve performance.
- **Blur Passes**: number of blur iterations. The higher number, the softer the result.
- **Downscaling**: reduces the size of the blur buffers. This option improves performance at quality expense.
- **Blur Spread**: increases the kernel radius of the blur pass. Can produce softer results in combination with the blur passes and downscaling.

Transparency and Semi-transparency support

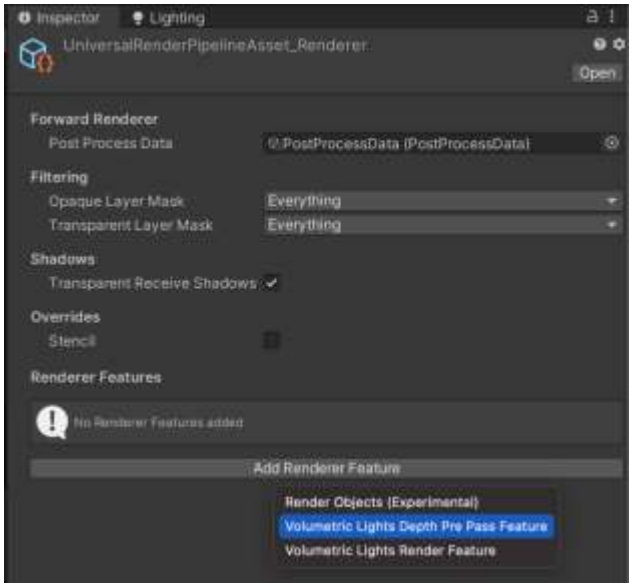
Volumetric Lights relies on the scene depth buffer to compute where objects are. Since transparent objects do not write to depth buffer, they might get overdrawn by the volumetric light effect.

There're two ways to solve this issue:

1) Render queues:
   You can change the render queue of the volumetric light, so it has a lower value than your transparent objects. Note that transparent queue starts at 3000. With this solution, volumetric light effect will render always before your transparent objects, regardless the position in the scene.

   If this option works for you, perfect! This solution is fast and easy to implement.

2) Alternatively, you can add the "Volumetric Light Depth Pre-Pass Render Feature" to the URP renderer:



Once you add the feature, you will see 3 parameters:

- **Transparent Layer Mask**. Specify which layer contains the transparent objects that you want to be considered when rendering the volumetric light effect.
- **Alpha Cutout Layer Mask**: if you have semi-transparent objects in the scene, you can use this mask to also include them but only the solid parts. Note that this option can be expensive.

Build Optimizations

Volumetric Lights uses a few shader keywords to optimize the execution of the effects. They're needed to enable the desired functionality. However, each keyword multiplies the number of shader variants so you may want to disable or remove some of them manually if you want to optimize your build. As a result, building the shaders for the first time can take a few minutes. Once it's built, Unity caches the compilation so next builds are much faster.

To reduce the number of shader variants and speed up the build, you can remove some keywords related to VolumetricLights options that you're not using in your project.

Locate and edit the file VolumetricLightURP.shader inside VolumetricLights/Resources/Shaders folder. The following lines declare the keywords. Feel free to comment out or remove any of these lines or keywords to reduce the number of shader variants:

```
#pragma multi_compile _ VF2_DEPTH_PREPASS        // used only if depth prepass is enabled
#pragma multi_compile_local _ VL_NOISE           // used if Noise option is enabled
#pragma multi_compile_local _ VL_BLUENOISE       // used if Blue Noise option is enabled
#pragma multi_compile_local _ VL_DIFFUSION       // used by the Diffusion Intensity option
#pragma multi_compile_local _ VL_PHYSICAL_ATTEN        // used by Quadratic attenuation
#pragma shader_feature_local VL_CUSTOM_BOUNDS   // used by Custom Bounds option
```

The following keywords are required per light type. If you don't use a specific light type, you can remove that keyword:

```
#pragma multi_compile_local VL_SPOT VL_SPOT_COOKIE VL_POINT VL_AREA_RECT VL_AREA_DISC
```

If you don't use shadows in the volumetric lights, you can remove this line. Or if you don't use the cubemap based shadows (only for point lights), you can remove the VL_SHADOWS_CUBEMAP keyword:

```
#pragma multi_compile_local _ VL_SHADOWS VL_SHADOWS_CUBEMAP
```

Important: if you upgrade Volumetric Lights to a newer version, any changes done to the VolumetricLightURP.shader will be lost! Remember to remove any unwanted keywords again after upgrading.

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:

- Reduce Raymarch Quality: reducing this value will indeed reduce the number of texture fetches per pixel.
- Increase “Min Step Size” value.
- Decrease “Max Steps” value.
- 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.
- Use the Downscaling option in the Volumetric Light Renderer Feature to improve overall performance.
- Disable “Use Blue Noise” option.
- 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” whenever possible.
- Reduce Render Scale in the Universal Rendering Pipeline asset. This will reduce the resolution of the framebuffer, improving the overall performance. This setting is especially useful on mobile devices since they use very high DPI screens and can afford to use a lower resolution.

Known Issues & Notes

Inverted Depth Bug

Universal Rendering Pipeline 7.2.0 has a bug which produces an incorrect (flipped) depth texture. This bug occurs when HDR is OFF, MSAA is x1 and Render Scale =1. More details here: https://forum.unity.com/threads/inverted-depth-texture-and-unresolved-shadowmap.824703/#post-5466879

Solution:
Option 1) Enabling HDR, MSAA or changing Render Scale (any of the three) produces the correct depth texture.
Option 2) Enable “Flip Depth Texture” option in Volumetric Light component.

VR rendering issues

Due to URP not setting inverted view-projection matrices correctly in VR, an alternate world space reconstruction mode needs to be enabled. To do so, locate and edit the Options.hlsl file and uncomment this line:
//#define USE_ALTERNATE_RECONSTRUCT_API

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.