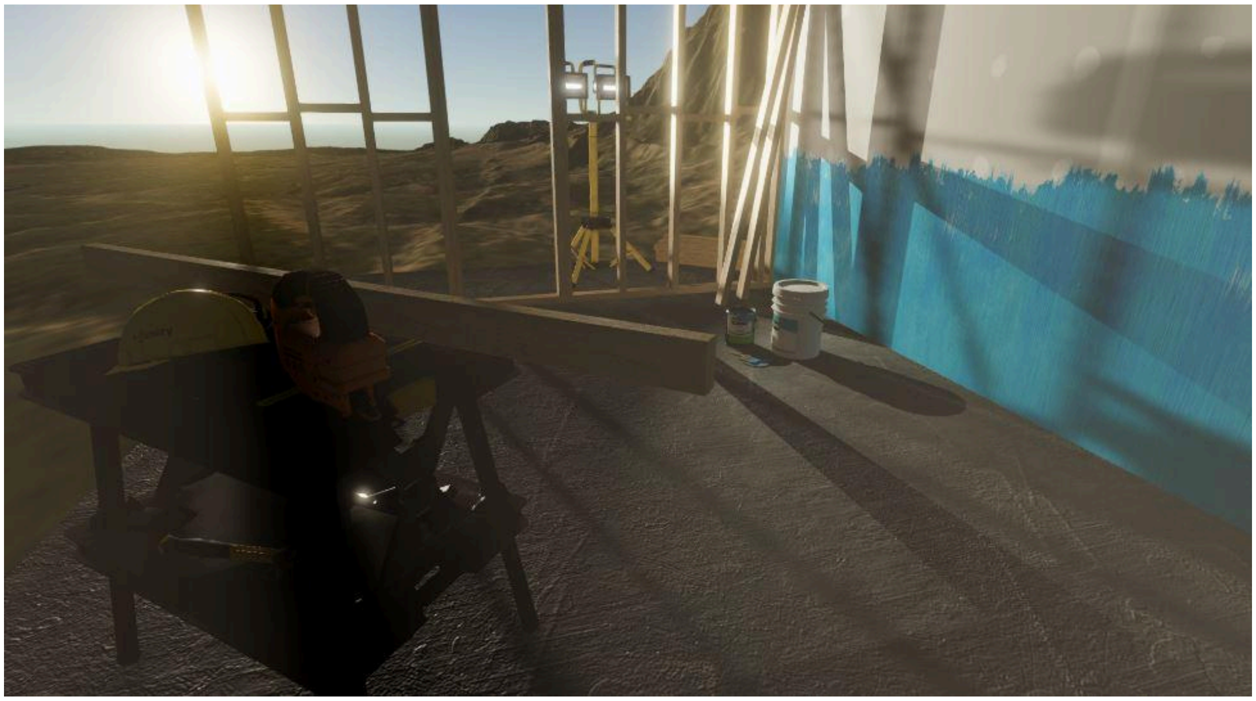


Godray Post Processing

By Joseph Y.

Overview

The godrays post processing is a post processing effect that uses screen space to mimic the look of godrays on screen. It is designed to run on the Unity URP pipeline and Unity 6, but theoretically, it could run on Unity 2022 but it is untested. This effect utilizes URP's render feature to integrate it into the render pipeline and HLSL and Shaderlab to create the shader effect.

Installation

Before you begin:

1. Ensure you use the Universal Render Pipeline (URP) and Unity 6 (6000.0.25f1).
2. Ensure you have installed the necessary packages, such as Shader Graph.
3. Ensure that you are using forward rendering in your universal renderer.
4. Ensure that the color space is set to linear in the project settings.
5. Ensure that under the project settings > graphics > Render Graph, you enable Compatibility Mode on (meaning you have Render Graph Disabled).

The effect is contained in a Unity package you can import into Unity. To do this, navigate to Assets > Import Packages, select the Unity package, and open it. A popup will appear showing the contents of the package. Choose to import everything from the package and click OK. They will be imported into the Unity project under the Godrays folder.
After importing, you should see several specific items, 2 C# scripts (1 containing our render feature and pass, and the other for its volume component), 3 shader files (2 for the gaussian blur, and 1 for the godray effect), and 1 text/HLSL file containing helper methods for the shader files.
To apply the effect:

1. Navigate to the universal renderer being used in your project and locate the "Add Render Feature" button.
2. Click the "Add Render Feature" button and choose the Godray render feature.
3. [Optional] Create a game object or go to the object containing your post-processing volume/volume.

4. [Optional] Under the volume component, click on **Add Override** and select Godray. If done correctly, the volume component should be updated with a new volume override that says Godray Volume.

You can adjust the render pass event by navigating to the Godray render feature in the universal renderer data. Click on the dropdown beside the setting and change the event to your preference, such as "after post processing".

Using the effect

You can go to a volume game object with the Godray volume on or the Godray render feature (if you are not using the Godray Volume) to adjust it. Do note that if you are using the volume to control the effect, you cannot use the render feature to adjust it since the volume takes control over the settings in the render feature. Here are the details about the parameters:

1. **Overall Scattering:** Affects the overall intensity of the godrays/light scattering effect. Lower will make the godrays brighter & foggier while higher will make it more blended with the scene.
2. **Min Scattering:** Affects minimum amount of intensity in the godrays/light scattering. Higher will make the godrays brighter & foggier and lower will make it more blended with the scene. Dependent on the values of Max Scattering.
3. **Max Scattering:** Affects maximum amount of intensity in the godrays/light scattering. Lower will make the godrays brighter & foggier and higher will make it more blended with the scene.
4. **Intensity:** Affects the intensity of the godrays. Is separate from the first 3 parameters, used for extra fine tuning. Higher will make the godrays brighter & foggier and lower will make it more blended with the scene.
5. **Steps:** Affects the amount of steps it takes within the code to create the godrays. Higher will make the godrays look nicer and accurate at the cost of a bit more performance while lower will make the godrays less accurate for slightly more performance.
6. **Max Distance:** Affects the max distance it will compute for the godrays. Higher will allow the godrays to go further at the cost of a bit more performance while lower will not allow the godrays to go further for slightly more performance.

7. **Down Sample:** Affects the compression on the godrays, off will be uncompressed (accurate & nicer godrays at the cost of performance) while quarter renders godrays at the quarter of the screen’s resolution (compressed but more performance).
8. **Phase Functions:** Affects one of the formulas used to compute the godrays, can be switched for a different look. Henyey Greenstein is overall less intensive/bright while Schlick is overall more intensive/bright.
9. **Gaussian Amount:** Affect how strong is the gaussian blur on the godray’s blending with the scene. Higher means stronger while lower means weaker. You typically don’t need to set it at high values.
10. **Gaussian Samples:** Affects the amount samples allocated to the gaussian blue. Having more samples will make the blurring/blending better at the cost of some minor performance.

Due to how it works, the effect only works on directional lights, and only one at that, which is the main light. The godrays use the sun as our reference for the projection of rays of sunlight, replicating how sunlight works in real life. So to adjust for godrays in the scene, you will need to rotate the directional lighting object to best suit your needs. If it is not possible to rotate the directional lighting in a way that would give you the godrays that you wanted, you can use the volumetric spotlight shader or other solutions as an alternative.

If an object or mesh is blocking the sun or its light rays, it will present as dark or normal looking, while areas where it is not blocking the sun will be brighter than the parts blocking the sun and its light rays. The godrays will also blend in between the blocked and unblocked areas.

The godray’s colors are determined by the color of the directional light. So to change the godray’s color, you will need to go to the directional light object, and change its color.

The effect can run on forward, forward+, or deferred rendering, which can be found in your project’s universal renderer data.

Known Issue

It is known that the godrays can clash with the other post processing where the godrays could be overridden or overshadowed. This is in the case as well where other effect can override the skybox, making the sun look very hard to see. Changing the godray's effect and the other post processing effect's render pass event can resolve this. Changing the rendering from forward to deferred can also help as well. There are also some cases where changing the godray effect and other effect's render pass events can make all rendering of opaque objects invisible (sometimes, it only affects the scene view, leaving the game view alone). This can be solved by changing it to deferred rendering (in the cases I tested, it does fix it but it may not in other cases that I have not known).

If you have multiple suns or sun-like objects (like a moon), the effect could misinterpret it. If you are okay with the look of godrays from 2+ suns, then it should be ok. But this issue can be the case if you have a custom skybox shader that implements something like a moon and does not cover the bottom half of the skybox, in that case, it can cause godrays that are unintended and may leak through objects. If you are using Unity's default skybox shader, you should be ok, this is more of a warning note for those using custom skybox shaders.

Update Log

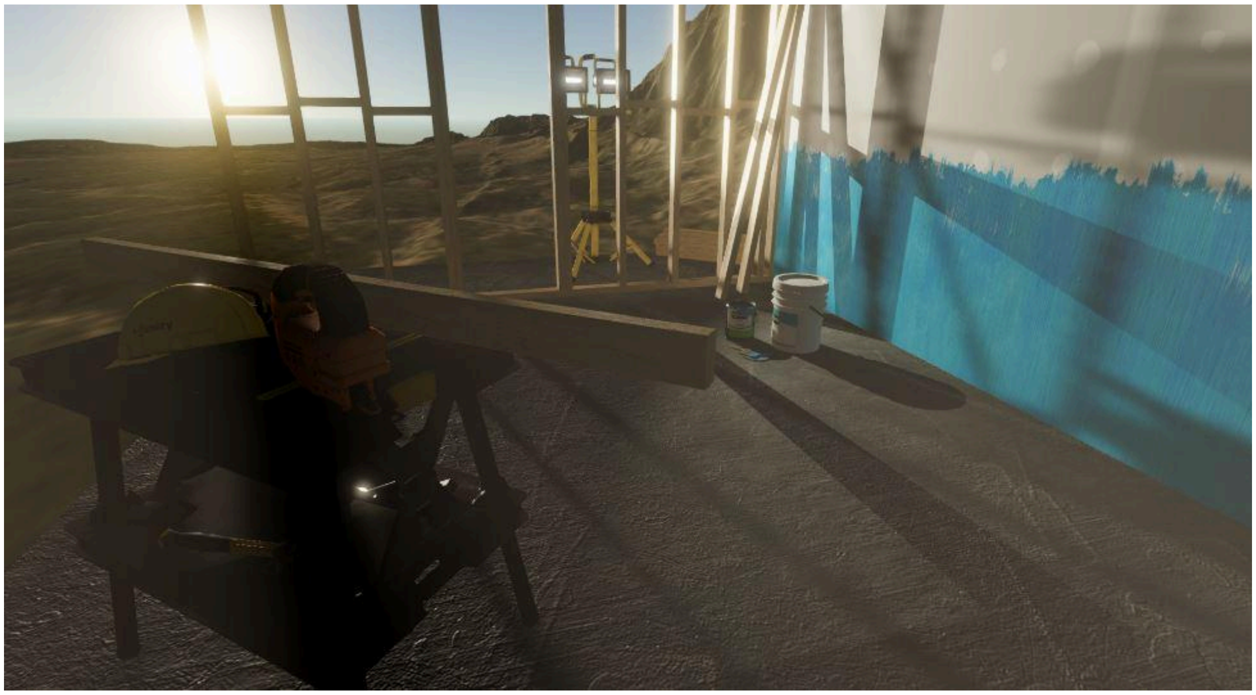
V0.9: The base implementation of the post processing.

Credits

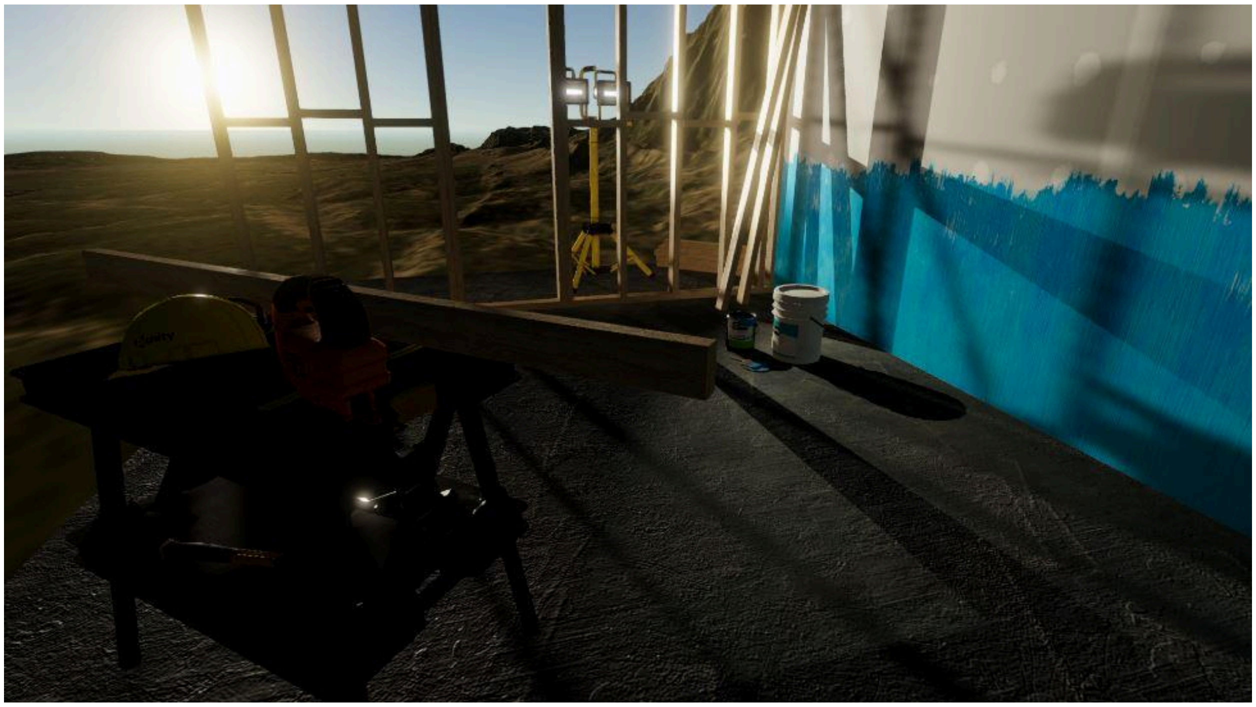
Some of the shader code is based on Mr-222's Unity Volumetric Rendering Repo's Volumetric Lighting code.
(https://github.com/Mr-222/Unity_Volumetric_Rendering/tree/main)

"Mossy/Grassy Landscape" (https://skfb.ly/6RYvL) by Šimon Ustal is licensed under Creative Commons Attribution (http://creativecommons.org/licenses/by/4.0/). No changes made. (The mountain landscape that you see in the screenshots for demonstration purposes)

Example Screenshots



What it looks like with the godrays on



What it looks like without the godrays