OVERCLOUD

Manual

Version 1.05

Table of contents

1. Introduction

Hi, and thanks for purchasing OverCloud.
This document will help you get started using the plugin, cover all of its features, and go through some of the technical aspects so that you can optimize it for your use case.
If you have questions or feature requests, please visit the Unity forum thread or shoot me an email at fewes17@gmail.com (although preferably you should use the forum thread, as this is helpful to others as well).

2. Getting started

2.1. Installing

(If you installed OverCloud from the asset store, this should be set up propely already)

1. Place the OverCloud directory in the Assets folder (or any sub-folder) of your Unity project.

OverCloud is made to be portable, and so can be placed in any sub-folder of the Assets folder.

2.2. Initializing

Before you begin, it is recommended you set your projects color space to Linear if you haven't done so already (the option for this can be found under *Edit* > *Project Settings* > *Player* > *Other Settings* > *Color Space*). It is also recommended you use some form of tonemapping image effect (an easy option is to use Unity's post processing stack and enable *Color Grading*).

To initialize OverCloud, drag the OverCloud prefab into a scene. Upon first usage, OverCloud will generate new 3D noise textures. This will take a little while but only has to be done once. The noise textures are stored in *Assets/Resources/CloudNoiseGen/OverCloud* and cannot be moved from there.

To enable OverCloud rendering in a camera, add a OverCloudCamera component to it. The OverCloudCamera component present on the main camera also controls the scene view rendering. For more information and rendering settings, see the OverCloudCamera section.

The OverCloud prefab comes with its own directional lights for the Sun and the Moon, parented to the prefab. If you wish to use your own, drag the Light references to the

OverCloud component and add a OverCloudLight component to both lights. A moon light is not required, but recommended. For more information, see the OverCloudLight section.

If “Atmosphere” and “Override Skybox Material” is enabled, OverCloud will override the Skybox material with its own. For more information, see the Atmosphere section.

2.3. Compatibility

OverCloud is compatible with several other assets, but some require manual code changes. Please refer to the compatibility webpage for details.

3. Components

This section goes through all the relevant components and what they are used for.
For a description of each parameter, check the tooltip by hovering the mouse over the name of the parameter.

3.1. OverCloud

OverCloud is the main component used to drive the sky, atmosphere, lighting and time of day system. It controls most effects on a global level and is required for the plugin to run. Only one OverCloud instance can be active at any given time.

3.1.1. Components

References to key components and resources used by OverCloud.

3.1.2. Volumetric Clouds

Settings related to the rendering of the volumetric cloud plane.

3.1.2.1. Compositor

The compositor is a world-space texture which describes the cloud coverage. The volumetric clouds use this as the base for the 3D noise. It is also used to generate cloud shadows and ambient occlusion.

3.1.2.2. Shape (3D Noise)

Settings controlling the 2-pass 3D noise used to give the clouds their volumetric look. The noise is applied as erosion of the base volume.

3.1.3. 2D Clouds

Here it is possible to add and customize 2D cloud planes which extend all the way to the horizon. These clouds are rendered between the geometry and transparent pass, are always sorted based on camera distance and use depth-fading (commonly referred to as *soft particles*) and near-camera fading.

3.1.4. Atmosphere

Settings related to the rendering of the atmospheric scattering and sky elements.

3.1.5. Lighting

Settings related to global lighting effects. The *Clouds Lighting* settings apply to both the volumetric clouds and the 2D cloud planes (with the exception of *Albedo*).

3.1.6. Weather Effects

Settings related to the weather effects, such as wind, rain and lightning.

3.1.7. Time Of day

Settings related to the dynamic time of day (TOD). The time of day system is location-based (latitude + longitude) and controls the position of the sun and moon (optional).

3.1.8. Weather Presets

Authoring and activation of custom weather presets. These are what allows you to fade between different weather types. To change the currently active weather preset from script, use *OverCloud.SetWeatherPreset*.

3.2. OverCloudCamera

The OverCloudCamera component needs to be added to any camera you want OverCloud to render in. It also allows you to control visual quality per-camera.
Having an OverCloudCamera component active on the main camera will also enable rendering in the scene camera.

3.3. OverCloudLight

The OverCloudLight component is a utility class used to drive the color and intensity of the sun and moon directional lights. Alternatively, it can be used to light the volumetric clouds with a point light (only one can be active at any given time).

3.4. OverCloudFogLight

The OverCloudFogLight component allows you to add volumetric lighting to a point or spot light. The opacity of the effect is based on the fog density inside the light volume, but it can also be overridden using the *Minimum Density* parameter.

3.5. OverCloudProbe

The OverCloudProbe component can be used to sample the cloud density and rain coverage at a specific position. Use this to detect if a Transform is inside (or under) the cloud volume.

3.6. OverCloudReflectionProbe

The OverCloudReflectionProbe component enables rendering of OverCloud in Reflection Probes. It also offers controls for when and how to update the reflection probe, whether to spread the workload over multiple frames or not, and functionality for saving the cubemap to a file (using horizontal face layout).

Please note that any reflection probe rendered using the OverCloudReflectionProbe component will not have properly convolved mip-maps. This is because Unity only offers functionality to calculate regular mip maps from script. As such, rougher reflections might look "off". If you simply wish to update the cubemap and don't care about having the atmosphere/clouds in it, use an OverCloudReflectionProbeUpdater **instead.**

3.7. OverCloudReflectionProbeUpdater

The OverCloudReflectionProbeUpdater component enables the updating of a Reflection Probe whenever the environment changes.

4. Rendering

4.1. Adding cloud shadows to deferred rendering path

In the default mode, cloud shadows are automatically injected in the screen-space shadow mask. Unfortunately, this mask is faded based on the *Project Settings* > *Quality* > *Shadow Distance* when sampled and there is no way (that I know of) to get around that fade.
As an alternative, in deferred rendering you can swap out the deferred shader to get cloud shadows which render as far as OverCloud allows.
To do this, go to *Project Settings* > *Graphics* and under *Built-in Shader Settings* change *Deferred* from *Built-in Shader* to *Custom*, and select the *Hidden/Internal-DeferredShading* found in *OverCloud/Resources/Shaders*.
Also, make sure that *OverCloud* > *Lighting* > *Cloud Shadows* > *Mode* is set to *External*, or you will get double shadowing.

4.2. Adding OverCloud support to a transparent shader.

For the atmosphere/fog there are two options:

1. Sample the atmosphere in the vertex shader, interpolate and apply in the fragment shader.
2. Sample and apply the atmosphere in the fragment shader.

4.2.1. Method 1

1. Add `OVERCLOUD_COORDS(TEXCOORDXX)` to your interpolator struct (usually v2f) and swap out XX for an available interpolator. The Atmosphere struct uses three interpolators so slot XX-XX+2 will be occupied.
2. Add `OVERCLOUD_TRANSFER(worldpos, o)` to your vertex function. `o` is your output struct variable.
3. Add `OVERCLOUD_FRAGMENT(color, screenUV)` to the end of your fragment function. `screenUV` can be calculated from the screen position (usually called `screenPos`): `float2 screenUV = i.screenPos.xy / i.screenPos.w`; Alternatively you can use `OVERCLOUD_FRAGMENT_LITE(color)` if you don't care about using the scattering mask.

4.2.2. Method 2

1. Add `OVERCLOUD_FRAGMENT_FULL(color, screenUV, worldPos)` to the end of your fragment shader.

If you want to support alpha blending with the volumetric clouds using the cloud depth buffer, multiply your alpha with the attenuation function like so:

```
color.a *= CloudAttenuation(distanceToCamera, screenUV, blendFactor);
```

`distanceToCamera` is the distance from the camera to the fragment in world units.
`screenUV` is again the screen-space UVs.
`blendFactor` controls the strength of the attenuation effect. It is best to play around with this value until it looks good.

4.3. Useful shader functions and variables

To include these functions in your shaders it is easiest to include the file *OverCloudInclude.cginc* in the base directory.

4.3.1. Defined in OverCloudCore.cginc

`float CloudShadows (float3 worldPos)`
Returns cloud shadows for a world-space position. Uses the dominant OverCloudLight direction (sun or moon).

`float CloudShadows (float3 worldPos, float3 lightDir)`
Returns cloud shadows for a world-space position. Uses the provided light direction.

`float CloudOcclusion (float3 worldPos)`
Returns cloud ambient occlusion for a world-space position.

`float BelowClouds (float height)`
Returns a float value which smoothly fades from the center of cloud plane to below it

`float RainSurface (float3 worldPos, float3 worldNormal)`
Returns a float value which defines how wet a world-space position should be.

`float3 _OC_LightDir`
The direction of the current dominant OverCloudLight (sun or moon).

`float3 _OC_LightColor`
The color of the current dominant OverCloudLight (sun or moon).

`float _OC_GlobalWindTime`
The current wind time. Used to drive the position of the clouds. Affected by the compositor timescale and weather preset wind multiplier.

`float3 _OverCloudOriginOffset`
The current world-space origin offset. Se below.

4.3.2. Defined in Atmosphere.cginc

`Atmosphere OverCloudAtmosphere (float3 worldPos)`
Calculates fog, scattering and extinction for a world-space position.

`void EvaluateAtmosphere (inout Atmosphere atm, float scatteringMask)`
This needs to be called after `OverCloudAtmosphere` for the result to be correct.
The reason these are split up is so the atmosphere can be sampled in the vertex shader.

`void ApplyAtmosphere (inout float3 color, Atmosphere atm)`
Applies the result stored in `atm` onto `color`.

`float FogShadow (float3 worldPos)`
Calculates the fog light attenuation of a directional light. Uses the dominant OverCloudLight direction (sun or moon).

`float FogShadow (float3 worldPos, float3 lightDir)`
Calculates the fog light attenuation of a directional light. Uses the provided light direction.

`sampler2D _OC_ScatteringMask`
The screen-space scattering mask.

4.3.3. Defined in Atmosphere.cginc

`float4 _OC_CurrentSunColor`
The current color of the sun directional light.

`float4 _OC_CurrentMoonColor`
The current color of the moon directional light.

4.3.4. Defined in Skybox.cginc

`float4 OverCloudSky (float3 worldPos, float2 screenUV)`
The function used by the skybox to calculate the sky color. Also used by the AtmosphereCamera shader to blend the scene into the sky.

4.4. Floating Origin

A floating origin allows you to have a larger world than the single-precision floating point format in Unity can handle. This feature is only really useful if you have already implemented a floating origin in your game, in which case you probably know what you are doing. You can move the origin with *OverCloud.MoveOrigin* which takes the new offset (additive) as a parameter, and you can reset the origin with *OverCloud.ResetOrigin*. You can also retrieve (and modify) the current offset value, using *OverCloud.currentOriginOffset*.
Internally, OverCloud uses the origin offset for all effects which are world-space dependent. When calling *OverCloud.GetDensity*, you don't need to apply your origin offset to the position parameter beforehand (if you've used *OverCloud.MoveOrigin* up until that point*)*.

5. FAQ

Q: I want higher-quality clouds, what settings should I change?
A: All quality settings are found on the OverCloudCamera component.
Decreasing the *Downsample Factor* will make the result on-screen higher resolution.
Increasing the *Light Sample Count* will give the volumetric lighting inside the clouds higher fidelity.
Checking the *High Quality Clouds* will use the high-resolution 3D noise (normally only used for the cloud alpha) when ray-marching the lighting.

Besides this, you should also increase the compositor resolution.
Increasing the particle radius or decreasing the cloud plane radius will give you a more accurate representation of the 3D volume.

Q: I want better performance, what settings should I change?
A: Basically the opposite of above. Most importantly you should leave the downsample factor as high as it will go without the clouds looking terrible in your camera. Decreasing the compositor resolution will also greatly increase performance on some GPUs. If you have multiple cameras, consider not rendering the volumetric clouds in some of them, as this is the biggest performance cost. Also, turn off the scattering mask and rain mask.

Q: I want to change the shape of the volumetric clouds, what settings affect this?
A: The local shape of the clouds is controlled by Shape (3D Noise). The shape of the cloud coverage is controlled by the *Noise Texture, Noise Scale, Noise Macro Scale*, weather preset *Cloudiness*, *Macro Cloudiness*, *Sharpness*, *Macro Sharpness* and *Optical Density*.

Q: Does OverCloud support <insert other asset here>?
A: If the asset renders opaque geometry, it will most likely support it out of the box. If it renders transparent effects (such as most ocean renderers), it will need to be modified to interact properly with OverCloud, please see Adding OverCloud support to a transparent shader.
Please get in contact with me either through mail or the forum thread if you'd like help integrating OverCloud with other assets.

Q: How can I check the cloud visibility between two points?
A: Use the function OverCloud.CloudVisibility().

Q: Changing the weather preset parameters doesn't seem to have any effect, why?
A: This can happen if there are no OverCloudCameras active in the scene. If this happens in the scene view, make sure at least one camera with the OverCloudCamera component has the tag *MainCamera*.

Q: Can I add more than one volumetric cloud plane?
A: Currently, OverCloud only supports a single volumetric cloud plane. I would like to support multiple, so I will *probably* add this in the future but no guarantees.

Q: Do I have to use the atmospheric scattering/skybox? I just want the clouds!
A: No. Just uncheck *Render Atmosphere* on the OverCloudCamera component and/or *Override Skybox Material* on the OverCloud component. If you'd like for the clouds to use Unity's default fog, go into the *OverCloudMain* shader and uncomment the line:
`#define OVERCLOUD_ATMOSPHERE`

Q: How do I know if something is inside the cloud volume, or under a cloud?
A: Use an OverCloudProbe.

Q: Does OverCloud use temporal filtering/reprojection?
A: No. OverCloud renders all effects in a single frame.

Q: How many shader keywords does OverCloud use?
A: OverCloud uses the following shaders keywords:

OVERCLOUD_ENABLED	Set during the entire render pass if OverCloud is enabled for the camera.
OVERCLOUD_ATMOSPHERE_ENABLED	Set if "Atmosphere" is checked on the OverCloudCamera component.
OVERCLOUD_POINTLIGHT_ENABLED	Set if there is an active OverCloudLight in the scene.
OVERCLOUD_SKY_ENABLED	Set if OverCloud is rendering its own sky (Skybox material is set to OverCloud/Skybox)
DOWNSAMPLE_2D_CLOUDS	Set if "Downsample 2D Clouds" is checked on the OverCloudCamera component.
HQ_LIGHT_SAMPLING	Set if "High Quality Clouds" is checked on the OverCloudCamera component.
LOD_CLOUDS	Set for the volumetric cloud LOD pass.
RAIN_MASK_ENABLED	Set if the dynamic rain depth mask is enabled.
SAMPLE_COUNT_LOW	Controls clouds/scattering mask quality.
SAMPLE_COUNT_MEDIUM	Controls clouds/scattering mask quality.

Q: Why is there no atmosphere/fog/clouds in my reflections?
A: You need to add an OverCloudReflectionProbe component to your Reflection Probe. Please note that this has limited support for glossy reflections.

Q: Any chance for LWRP/HDRP support?
A: Maybe in the future, but currently there are no plans for it.

Q: Why is the sun shining through my scene?
A: What you are seeing is the Mie Scattering effect. Go to Atmosphere > Mie Scattering and play with the distance fade sliders until it looks right.

Q: Can I make custom changes to the weather preset parameters during runtime?
A: Yes. Subscribe to the `OverCloud.beforeShaderParametersUpdate` event, then make your changes to `OverCloud.current`.