Starting Guide

Raphael Ernaelsten

@RaphErnaelsten

Table Of Contents

Table Of Contents	**1**
About Aura	**5**
Glossary	**7**
Getting Started	**8**
Aura Main Component	**11**
Aura Lights	**12**
Aura Volumes	**16**
Illuminate/Fog Particles	**23**
Using Aura in your Shaders	**24**
Requirements	**24**
Acknowledgement	**26**
Special Thanks	**26**
License	**26**

About Aura

TL;DR;

What Aura is :

- a local (within a defined distance from the camera) simulation of the light scattering into the surrounding medium

What Aura is not:

- an atmospheric scattering simulation
- a cloud simulator

Aura simulates the scattering of the light in the environmental medium and the illumination of micro-particles that are present in this environment but invisible to the eye/camera.
This effect is also called “volumetric fog”.

The directional light is scattered into the air and illuminates the invisible micro-particles.

Aura uses a globalist approach that makes every lights and injection volumes interconnected. This means that when you locally modify the environment somewhere, it will automatically affect all the lights and injection volumes. No boring and redundant per-instance setup ...

Also, all lights and injection volumes are dynamic and then, can be created, modified or destroyed at runtime.

Aura packs a bunch of features such as full lighting support, injection volumes and particles illumination.

Click here to view the recap of all the features of Aura (on YouTube).

Glossary

This document will often refer to specific words that worth a bit of contextualization.

Medium
The environmental matter that fills the environment and which the lights will scatter in. Typically it will be air, water or glass; but can be more exotic gases/liquids/solids in your convenience.

Density
This is the amount of invisible micro-particles that fill the medium.
It can also be understood as the “fog factor”.

Color (or Light Color)
The tint and intensity of volumetric light that is in a particular point.

Anisotropy
This represent the scattering factor the light will undergo from its emission through the camera.
0 means that there is no scattering and that the light will travel straight to the camera without loss. 1 means that the light will be totally scattered in all directions before reaching the camera.
In other words, it controls how much the volumetric lighting concentrate around the light source.
Tips : This can practically be useful to simulate the amount of humidity of the medium (0 = dry medium, no scattering; 1 = humid medium, maximum scattering). Note that this is not a general rule, just an interpretation tip. The best way is always to try.

Getting Started

Set the Aura Main Component

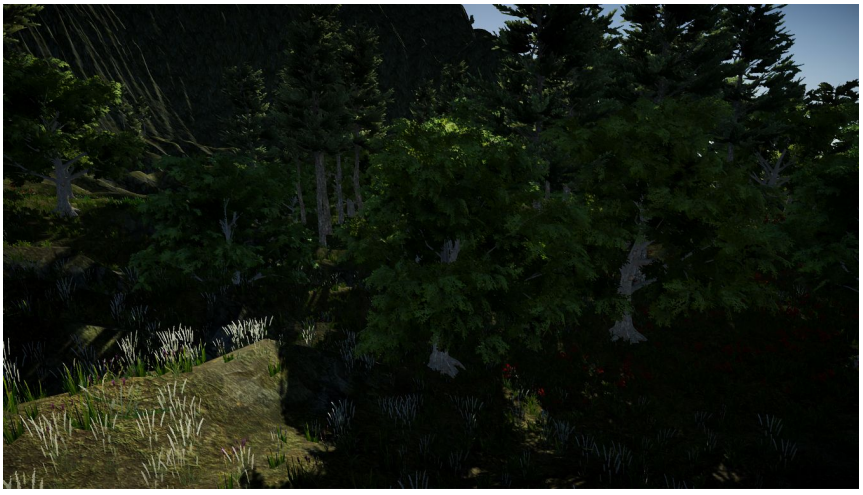
Always set the Aura Main Component on the Camera GameObject (see Aura Main Component section).

The default values are purely for visual feedback.
The process should be thought as iterative, adding one element after another.
Following that guideline, the ambient data should be set as low as the medium would be.

For example, if we want to simulate air, starting with a very low Ambient Density factor is a good starting point as, often, surrounding air is not much foggy.
Let's set it to 0.025.

Then, we will evaluate the wetness of the air and express it through the Ambient Anisotropy factor. This can range from (nearly) 0 very dry air like in deserts, to (nearly) 1 very humid like in a dampy cave.
Let's set it to 0.8.

Finally, we will have to assess the Ambient Color and its Strength, in other terms, the minimum lighting value in the medium.
Pretty much like the usual ambient lighting, this can be thought as the lighting that will be in the shadows.
For our example, being in a forest-ish environment, we will set the color to a low saturated green and the strength to a small value as it's supposed to be in the shadows, 0.45 will do.



Add Aura Lights

We can now add/assign Aura Lights in the scene (see Aura Lights section).

In our example, we will just modify the out-of-phase color in order to tint the light to a bluish color when the light is scattered (or when the light ray is not around the light source).

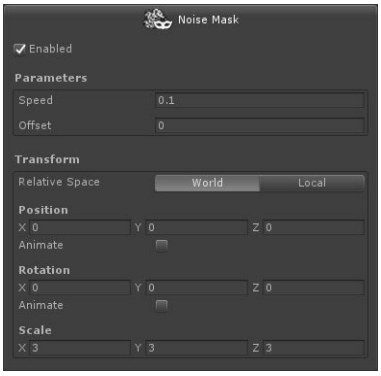
Add Aura Volumes

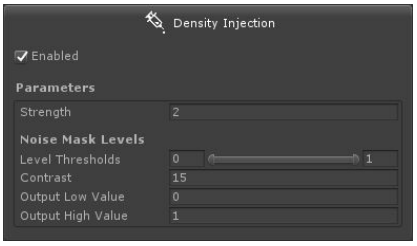
We can also add Aura Volumes in the scene (see Aura Volumes section).

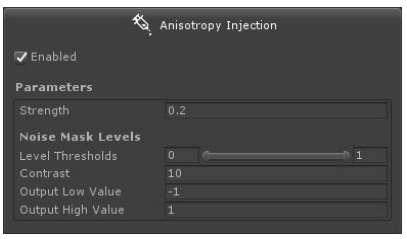
In our example, we will put one Aura Planar Volume to add more fog down the hill, in the woods.

To do so, we will add Density (the amount of fog) and modify the Anisotropy (to simulate variations of humidity and catch the eye).

On top of that we will use a Noise Mask so it doesn't look like a solid block but more like a smoky/cloudy place.



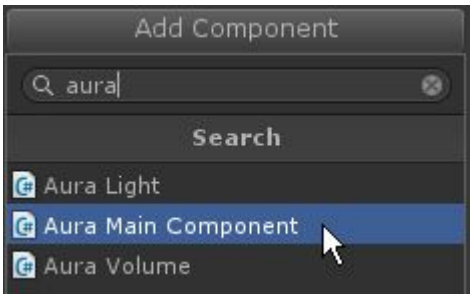




Click here to see this example in the On/Off Comparison video (on YouTube).

Aura Main Component

First thing to do to get started is always to set the Aura Main Component on the Camera GameObject.



This will start the whole volumetric computation process and set the ambient data for the volumetric lighting.

After having assigned the Aura Main Component, any amount of Aura Light or Aura Volume can be added in no particular order.

The Aura Main Component is composed of two tabs of parameters.

Base Injection Tab

Ambient Injection

The "Ambient Injection" parameters set the starting Density, Color and Anisotropy of the environment.

Ambient Density

The starting amount of fog in the medium.

Ambient Anisotropy

The starting light scattering factor of the medium.

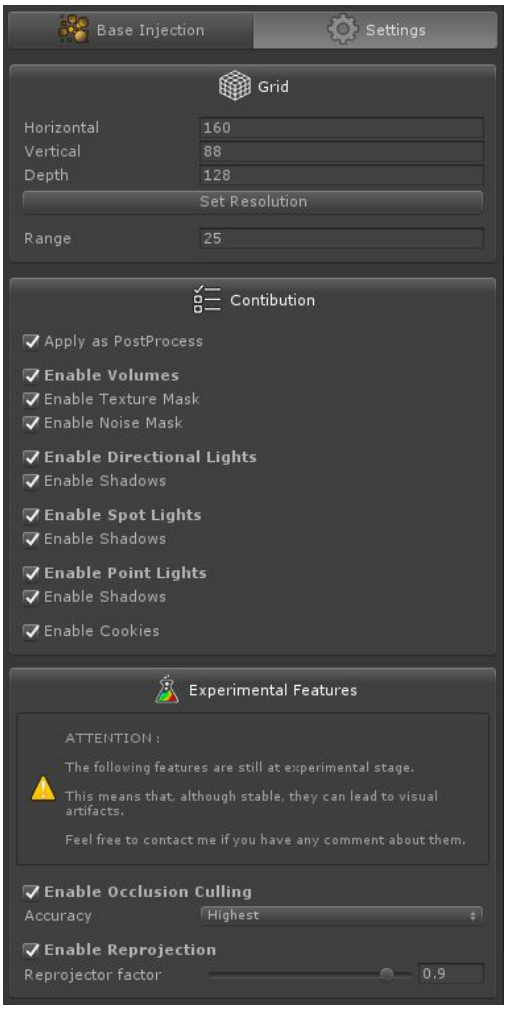
Ambient Light

The tint of the starting volumetric lighting of the medium.

Ambient Light Strength

The intensity of the Ambient Light.

Settings Tab



Grid

Allows to determine the density of cells used to compute the volumetric lighting.

Horizontal/Vertical/Depth

The resolution of the grid.

Range

The maximum distance within which the volumetric lighting will be computed.

Contribution

Allows to enable/disable what type of contribution will be computed.
Note that the existence of the different contributions are handled by the system at runtime.

Experimental Features

These features are still at experimental stage.
This means that, although stable, they can lead to visual artifacts.

Enable Occlusion Culling

Allows to compute the maximum visible depth of the frustum grid.
This leads to avoid computing cells that are invisible to the camera because hidden behind objects.

Occlusion Culling Accuracy

The precision used to determine the maximum visible depth.

Enable Reprojection

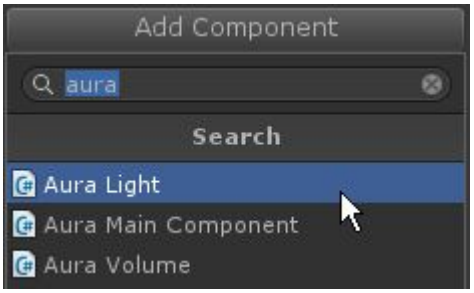
Allows to blend the current (jittered) computed frame with the previous one.
This leads to a smoother volumetric lighting, especially with a low resolution grid.

Reprojector factor

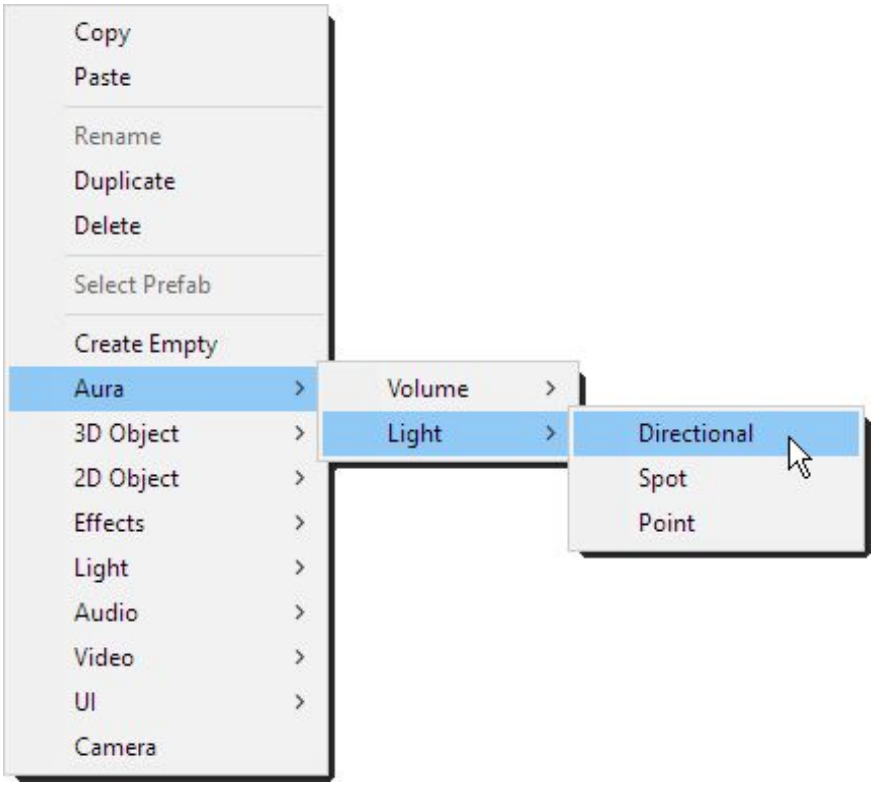
Controls the scale and amount of reprojection.

Aura Lights

The Aura Light components can be assigned on the Lights you want to be taken into account into the volumetric lighting computation.



Aura Lights can also be directly created using the GameObjects menu.



The default value of the Aura Light components are normalized to fit Unity's lighting system. Therefore, leaving a strength factor of 1 for the volumetric lighting will closely fit the strength of the surface lighting of Unity's lighting system.

The Aura Light component is composed of two tabs. One for the common parameters, whatever the type of light; one with additional parameters that will change regarding the type of the light.

Base Parameters Tab

Strength

The intensity of the volumetric lighting computed from the light.

Override Color

Allows to ignore and replace the color of the light for the volumetric lighting.

Enable Shadows

Enables the computation of the shadows of the light in the volumetric lighting (if the light casts shadows).

Enable Cookie

Enables the computation of the cookie of the light in the volumetric lighting (if the light has a cookie).

Additional Parameters Tab (Directional Light)

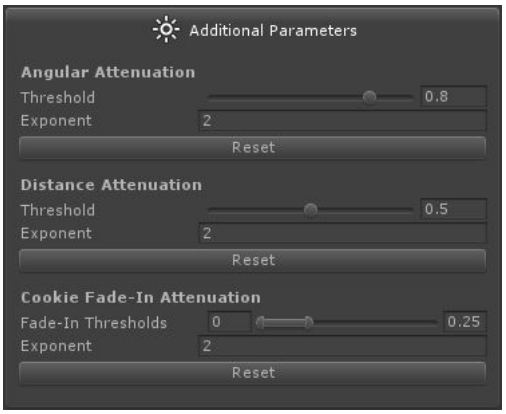
Enable Out-Of-Phase Color

Allows to cancel the extinction of the light when it goes scattered and replace it with a color.

Strength

The intensity of the Out-Of-Phase Color.

Additional Parameters Tab (Spot Light)



Angular Attenuation

Allows to tweak the fade-out of the light from the center to the outer border of the cone.

Threshold

Normalized angle when the fade will start, until 1.

Exponent

Curve of the fading.

Distance Attenuation

Allows to tweak the fade-out of the light within its range.

Threshold

Normalized distance when the fade will start, until 1.

Exponent

Curve of the fading.

Distance Attenuation

Allows to tweak the fade-out of the light within its range.

Threshold

Normalized distance when the fade will start, until 1.

Exponent

Curve of the fading.

Cookie Fade-In Attenuation

Allows to tweak the fade-in of the light's cookie within its range.

Fade-In Thresholds

Normalized range where the cookie will fade in.

Exponent

Curve of the fading.

Additional Parameters Tab (Point Light)

Distance Attenuation

Allows to tweak the fade-out of the light within its range.

Threshold

Normalized distance when the fade will start, until 1.

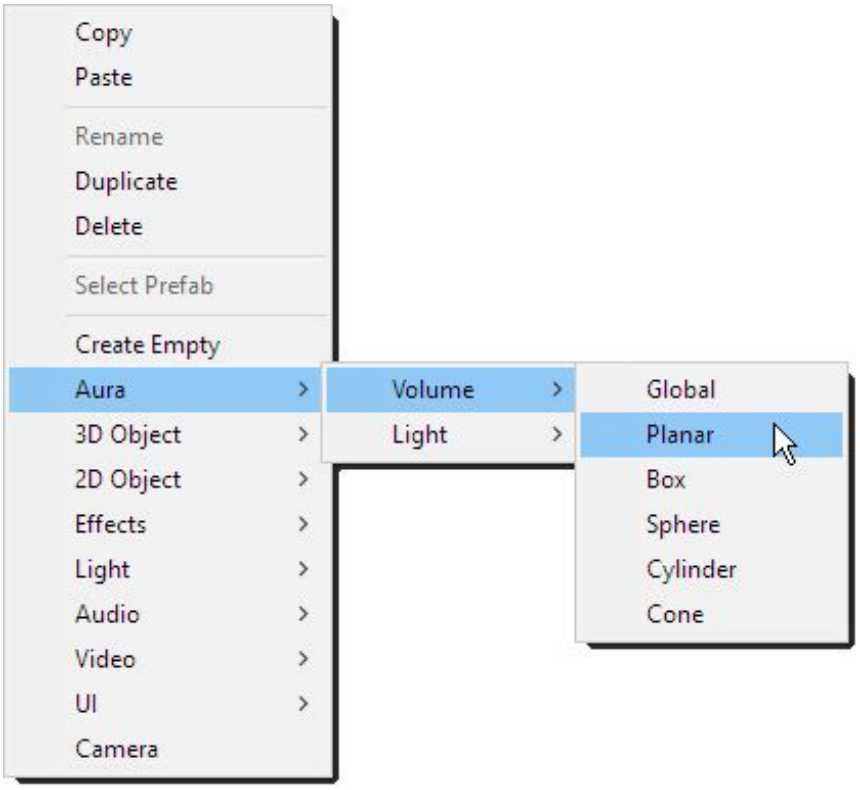
Exponent

Curve of the fading.

Aura Volumes

Aura Volumes can be added into the scenes to locally modify one or several of the base ingredients of Aura's system (Density, Color and Anisotropy).

Aura Volumes can be directly created using the GameObjects menu.



The Aura Volume component is composed of four tabs.

Settings Tab

The settings tab will allow to tweak the shape and the border attenuations of the Volume.
Noise and Texture masks can also be enabled to modulate the injections parameters.

Volume Shape

Shape of the Volume

Select the volumetric shape of the Volume between :

- Global
- Planar
- Box
- Sphere
- Cylinder
- Cone

Parameters

Allow to tweak the fading on the borders of the shape, allowing a smooth transition between the inside and the outside of the volume.

Falloff Exponent

Curve of the fade-out attenuations.

Noise Mask

Enable

Activates the dynamic noise mask and applies it on the Density, Color and Anisotropy injections.

Speed

Speed of the morphing effect.

Offset

Offset of the morphing effect, in order to desynchronize noise that have the same speed.

Transform

Allows to set (and animate) the position, rotation and scale of the noise.

Texture Mask

Enable

Activates the volumetric texture mask and applies the RGB channels data on the Color injection and the Alpha channel on the Density and Anisotropy injections.

Texture

The source Texture3D to use.

Transform

Allows to set (and animate) the position, rotation and scale of the volumetric texture.

Warp Mode

Sets how the texture should warp outside the 0-1 coordinates range.

Filter Mode

Set how the volumetric texture should be filtered when displayed.

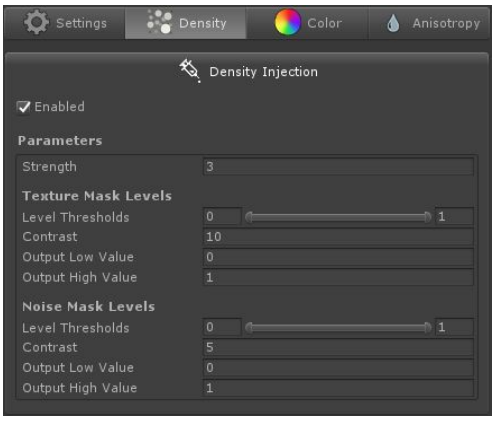
Clip on Alpha

Allows to bypass computation of the Volume cell based on the Alpha channel.

Clipping Threshold

Threshold under which the Volume cell will be bypassed.

Density Tab



Enable

Allows to add/remove fog inside the Volume.

Strength

The amount of fog to be added.
This value can be negative, meaning that fog can be removed.

Texture Mask Levels (if Texture Mask)

Remaps and contrasts the texture mask value.

Levels Thresholds

Clamps the lower and upper range of the value.

Contrast

Contrasts the clamped value.

Output Low Value

Remap the upper threshold of the value to a new value.

Output High Value

Remap the lower threshold of the value to a new value.

Noise Mask Levels (if Noise Mask)

Remaps and contrasts the noise mask value.

Levels Thresholds

Clamps the lower and upper range of the value.

Contrast

Contrasts the clamped value.

Output Low Value

Remap the upper threshold of the value to a new value.

Output High Value

Remap the lower threshold of the value to a new value.

Color Tab

Enable

Allows to add/remove light inside the Volume.

Strength

The intensity of the light to be added.
This value can be negative, meaning that light can be removed.

Texture Mask Levels (if Texture Mask)

Remaps and contrasts the texture mask value.

Levels Thresholds

Clamps the lower and upper range of the value.

Contrast

Contrasts the clamped value.

Output Low Value

Remap the upper threshold of the value to a new value.

Output High Value

Remap the lower threshold of the value to a new value.

Noise Mask Levels (if Noise Mask)

Remaps and contrasts the noise mask value.

Levels Thresholds

Clamps the lower and upper range of the value.

Contrast

Contrasts the clamped value.

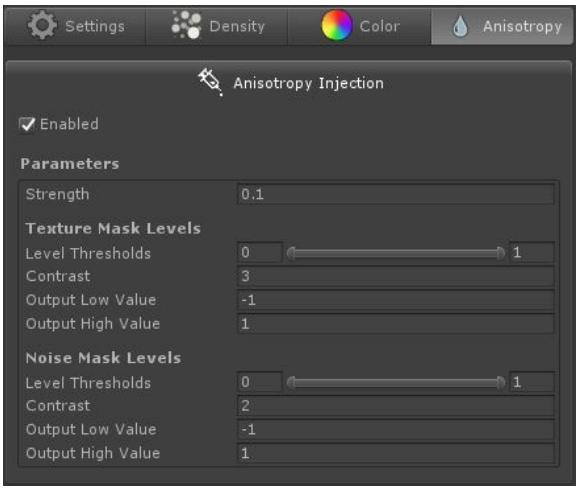
Output Low Value

Remap the upper threshold of the value to a new value.

Output High Value

Remap the lower threshold of the value to a new value.

Anisotropy Tab



Enable

Allows to add/remove anisotropy inside the Volume.

Strength

The amount of anisotropy to be added. This value can be negative, meaning that anisotropy can be removed.

Texture Mask Levels (if Texture Mask)

Remaps and contrasts the texture mask value.

Levels Thresholds

Clamps the lower and upper range of the value.

Contrast

Contrasts the clamped value.

Output Low Value

Remap the upper threshold of the value to a new value.

Output High Value

Remap the lower threshold of the value to a new value.

Noise Mask Levels (if Noise Mask)

Remaps and contrasts the noise mask value.

Levels Thresholds

Clamps the lower and upper range of the value.

Contrast

Contrasts the clamped value.

Output Low Value

Remap the upper threshold of the value to a new value.

Output High Value

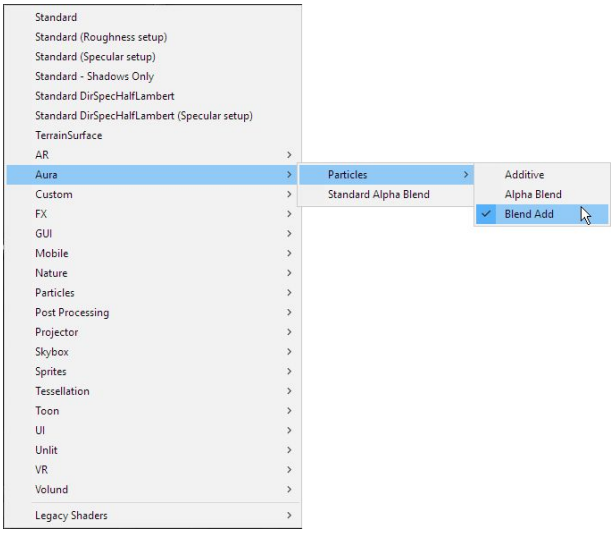
Remap the lower threshold of the value to a new value.

Illuminate/Fog Particles

Since Aura stores and sends the punctual and accumulated volumetric lighting as global buffers, it is easy to retrieve them if we know the element's position inside the frustum.

That is how the particles illumination and fog work with Aura.

If you want to illuminate/fog your particles with Aura, you can choose between the provided shaders or make your own, following the provided ones as examples.



Using Aura in your Shaders

In the same idea as the particles illumination/fogging, Aura can be implemented in your own custom shaders (transparent or not) in 3 very piece-of-cake steps.

1. Include “Aura.cginc” located in the “Aura/Shaders/” folder.

```
#include "../../../Aura/Shaders/Aura.cginc"
```

2. In the Vertex Shader, compute the position inside the Aura frustum with
   void Aura_ApplyLighting(inout float3 colorToApply, float3 screenSpacePosition, float lightingFactor)

```
o.frustumSpacePosition = Aura_GetFrustumSpaceCoordinates(v.vertex);
```

3. You can now apply the volumetric lighting with the following method
   void Aura_ApplyLighting(inout float3 colorToApply, float3 screenSpacePosition, float lightingFactor)

```
Aura_ApplyLighting(color, i.frustumSpacePosition, 1.0f);
```

   and/or apply the fog with the following overloads
   void Aura_ApplyFog(inout float3 colorToApply, float3 screenSpacePosition)
   void Aura_ApplyFog(inout float4 colorToApply, float3 screenSpacePosition)

```
Aura_ApplyFog(color, i.frustumSpacePosition);
```

Requirements

Aura strictly requires full support of the following elements to work :

- RenderTextures (3D as well)
- Texture2DArrays
- ComputeShaders

Please verify that the support of these elements is not limited especially on lower platforms.

Aura release was targeted for Unity 2017.2 :
- older version will not be supported (although I can help making it work ;-))
- newer version will be supported with updates if necessary

Historically, Texture2DArrays were introduced in Unity 5.4 which makes it the lowest version compatible with Aura.

However, multi-threaded ComputeShaders compilation was introduced in Unity 2017.2 which makes it the advised minimum version (as the main ComputeShader in looong to compile with its many variants).

Acknowledgement

Aura uses the following works :

- Bartlomiej Wronski (@BartWronsk)'s presentation : "Volumetric Fog : Unified compute shader based solution to atmospheric scattering"
- Sebastien Hillaire (@SebHillaire)'s integration formula : "Physically Based and Unified Volumetric Rendering in Frostbite"
- Ashima Arts's 4D Simplex Noise : https://github.com/ashima/webgl-noise

About the provided demo scene :

- Marko Dabrovic's Sponza scene : http://hdri.cgtechniques.com/~sponza/files/
- Unity Lab's smoke spritesheet : "VFX Image Sequences & Flipbooks"

Special Thanks

For their time and help, I would like to cheerfully thank :

- Bartlomiej Wronski (@BartWronsk)
- Florent Guinier (@FlorentGuinier)
- Sebastien Lagarde (@SebLagarde)
- Gil Damoiseaux (@Gaxil)
- All the people that helped me by testing Aura, and kept me motivated with their constructive feedbacks and their kind words.

License

Copyright (c) Raphaël Ernaelsten (@RaphErnaelsten)
All Rights Reserved.

Although Aura (or Aura 1) is still a free project, it is not open-source nor in the public domain anymore.
Aura is now governed by the End Used License Agreement of the Asset Store of Unity Technologies.

All information contained herein is, and remains the property of Raphaël Ernaelsten.
The intellectual and technical concepts contained herein are proprietary to Raphaël Ernaelsten and are protected by copyright laws.
Dissemination of this information or reproduction of this material is strictly forbidden.