Lux Lit Particles

Lux Lit Particles provide advanced particle shaders which add high quality yet performant real time lighting including normal mapping, wrapped around diffuse lighting, translucency and directional shadows in order to make your particles just fit your scene’s lighting.

Compatibility

Lux Lit Particles have been written with classic view aligned billboards in mind. Other billboard modes like *Horizontal Billboards* or non view aligned billboards are only partly supported. Find out more >

Lux Lit Particles only support alpha blended particles.

Lux Lit Particles have been successfully tested on DX11, OpenGLCore and Metal (desktop). Mobile platforms neither have been tested nor are recommended. DX9 is not supported.

Table of Content

Getting Started

1. Always add the **_LuxParticles_AmbientLighting prefab** to your scene.
2. In case you want to use directional shadows casted on the particles you have to assign the **LuxParticles_DirectionalLight** script to the directional light.
3. Lux Lit Particles rely on a combined normal, depth, alpha texture which you have to prepare upfront.

Please have a look at the included demo scenes to find out more.

Shader Properties

Color Color multiplier which lets you tint the particles and adjust transparency (alpha).

Normal (RG) Depth (B) Alpha (A) A combined texture which holds:

- the red and blue color channel of a regular normal map in RG,
- a depth or height map in B (only needed in case you use per pixel shadows)
- and alpha (transparency) in A.

By using a combined texture the shader usually gets away with just a single texture lookup. As the texture contains a normal map it has to be imported in linear color space. So please make sure that you have unchecked “sRGB Texture” in the import settings.
Storing a normal in RG unfortunately results in quite bad quality as the red color channel has pretty bad compression. So you should enhance compression quality by setting the format to “RGB(A) Compressed BC7”.

In case you use a *Texture Sheet Animation* make sure that the *Wrap Mode* is set to *Clamp* in the import settings.

Enable Albedo In case you want to use a dedicated albedo texture next to the base *Normal (RG) Depth (B) Alpha (A)* texture you have to check this. *Actually i think you do not need an albedo texture for particles such as smoke or fog. So most of the time you can simply save 1—2 texture lookups by leaving this unchecked and control albedo using the **Color** attribute and **Color Over Lifetime** in the particle modules only.*

Albedo (RGB) The dedicated albedo texture in RGB.
In case you use a Texture Sheet Animation make sure that the Wrap Mode is set to Clamp in the import settings.

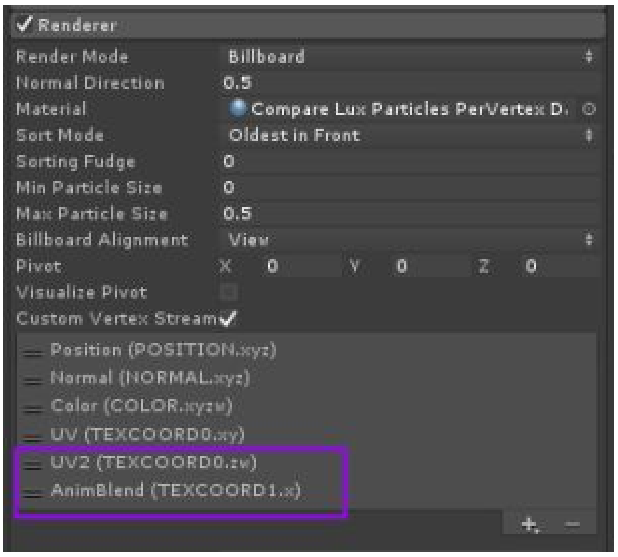
Enable Emission In case you want to use an emission texture next to the base *Normal (RG) Depth (B) Alpha (A)* texture you have to check this. *In order to find out what emission can do please check out the provided Emissive Smoke.prefab. Unlike the Albedo texture Emission really adds some unique features you can not mimic using the **Color** attribute and **Color Over Lifetime** in the particle modules. You can of course always add a 2nd particle system which renders the emissive parts instead like little sparks.*

Emission (RGB) Alpha (A) The emission texture in RGB. You should also provide an alpha for the emissive parts so that the shader can render these even if the base *Normal (RG) Depth (B) Alpha (A)* texture would define full transparency.
In case you use a Texture Sheet Animation make sure that the Wrap Mode is set to Clamp in the import settings.

Emission Scale lets you scale emission as sampled from the emission texture.

Enable Flipbook Blending If checked the shader will blend between two samples of your flipbook – smoothing the transition between the the frames of your *Texture Sheet Animation*. *This will effect all enabled textures (Normal (RG) Depth (B) Alpha (A), Albedo and Emission), so in worst case the shader will do up to 6 texture samples per pixel.*

Please note: In order to use this feature you have to declare a *Custom Vertex Stream* in the *Renderer* module of the particle system and add *UV2* and *AnimBlend*.



✓ Renderer

Render Mode	Billboard
Normal Direction	0.5
Material	Compare Lux Particles PerVertex D.
Sort Mode	Oldest in Front
Sorting Fudge	0
Min Particle Size	0
Max Particle Size	0.5
Billboard Alignment	View
Pivot	X 0 Y 0 Z 0
Visualize Pivot	
Custom Vertex Stream	✓

Position (POSITION.xyz)
Normal (NORMAL.xyz)
Color (COLOR.xyzw)
UV (TEXCOORD0.xy)
UV2 (TEXCOORD0.zw)
AnimBlend (TEXCOORD1.x)

Soft Particle Factor controls how particles fade out when they get close to the surface of objects written into the depth buffer.

Camera Fade Distances Lets you specify at which distances the particles fade in and out:

- **Near: X (Start)** The closest distance particles can get to the camera.
- **Near: Y (Range)** Range over which particles will fade in (near only).
- **Far: X (End)** The farthest distance particles can get away from the camera before they are fully faded from the camera's view.
- **Far: Y (Range)** Range over which particles will fade out (far only).

Tessellation (only available if you select the tessellation shader variant)

- **Subdivisions** Number of subdivisions created by the tessellator. Keep this number as low as possible for performance reasons. Make sure you always use an odd value (like 3, 5, 7, …) as even values will create odd tessellation.
- **MinDist (X)** distance to camera where the specified number of subdivisions will be reached.
- **MaxDist (Y)** distance to camera where no tessellation will be applied anymore.

Lighting find out more >

- **Full Per Pixel Lighting** if checked the shader will render 4 lights per pixel. If unchecked only the first and most dominant light (usually the directional light) will get per pixel lighting and normal mapping and 3 further lights will be shaded per vertex.
- **Wrapped Diffuse** lets you smooth the lighting on the particles. 0 gives you full NdotL lighting while 1 would make your particles just flat.
- **Translucency** controls the amount of light scattering through the particle.
- **Depth Influence** lets you occlude translucent lighting according to the depth texture.

Shadows find out more >

- **Enable directional Shadows** if checked the particles will receive real time shadows from the directional light (per vertex by default).
    - **Per Pixel Shadows** if checked the cascaded shadow map will be sampled per pixel instead of per vertex.
        - **Extrude** will shift the worldspace position at which the cascaded shadow map is sampled according to the depth texture.
- **Casted Shadow Density** in case your particles cast real time shadows use this parameter to adjust the density of the casted shadows.

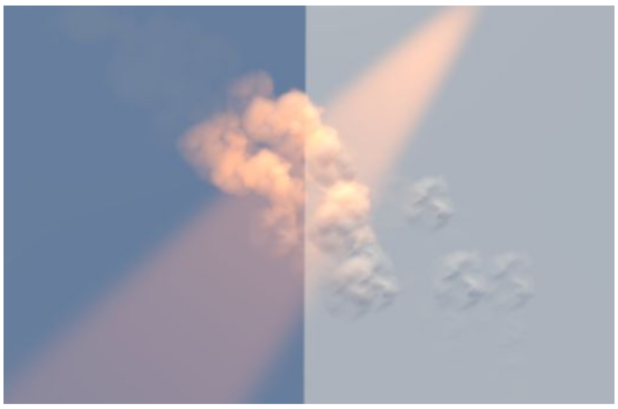
Ambient Lighting

- **Per Pixel Ambient Lighting** If checked the shader will sample ambient lighting per pixel instead of per vertex (expensive…). **Please note:** Do not check this if *Lighting → Environment Lighting → Source* is set to *Color* as the shaders do not handle this (as it does not make any sense anyway).
- **Enable Light Probes** Check this in case you want to the shader to sample ambient lighting from Light Probes. **Please note:** you also have to add the LuxParticles_LocalAmbientLighting script to all particle systems using the material to make this work. Find out more >

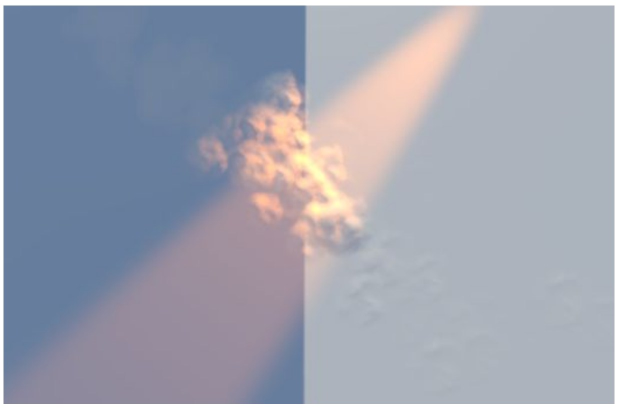
Lighting

Vertex and Pixel Lighting

Lux Lit Particles support up to 4 per pixel lights which of course are more expensive than per vertex lights. Usually you can get away with just a single per pixel light which will reveal the normals and do all other lighting just per vertex:



1 Per Pixel Light Only the directional light will show normal mapping. The spot light is calculated per Vertex. This is the default lighting if *Full Per Pixel Lighting* is unchecked.



4 Per Pixel Lights If *Full Per Pixel Lighting* is checked also the spot light will show the normal mapping..

Normal Direction

When it comes to particle lighting the *Normal Direction* of the particles as set in the *Renderer* module of the particle system plays an important role. Unity Docs >

A *Normal Direction* of 1 will give you quite accurate ambient lighting from the Skybox or a Gradient as the particle normals will more or less will fit the world normal of the particle's billboard. Per vertex lighting however might look a bit harsh.

You may try to work against this by setting up *Wrapped Diffuse* and *Translucency* or lowering the *Normal Direction* to 0.5 - 1.0.

Ambient Lighting

Lux Lit Particles support ambient or *Environment Lighting* driven by *Skybox*, *Gradient* or *Color*. However as the particles use the vertex lightmode ambient lighting is not set by Unity automatically and we have to set it up our own.

Do so by adding the "**_LuxParticles_AmbientLighting**" prefab to your scene. The attached script will sample the ambient lighting and pass the result as global shader variables to the particle shaders.

Please note: In case your ambient lighting does not change over time and your particle systems do not move you may uncheck "Update Per Frame".

Shadows

Lux Lit Particles may receive shadows from the main directional light. In order to make this work you have to:

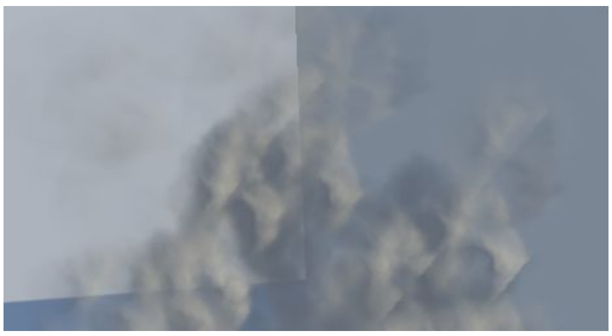
1. assign the **LuxParticles_DirectionalLight** script to the directional light. This script will copy the cascaded shadow map into a rendertexture which then can be sampled by the particle shaders.
2. check **Enable directional Shadows** in the particle material inspector.

Unfortunately shadows are calculated based on the world position of the given vertex or pixel. So flat billboards will sample "flat" shadows. Furthermore we can't really afford to sample soft shadows, so per pixel sampled shadows will create some harsh and noticeable diagonal edges between shaded and lit parts of the particles.

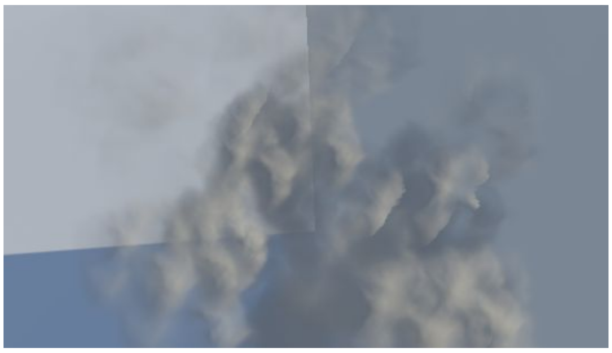
Using the *Extrude* parameter lets you break up the straight edge but does not solve the harsh border.

Sampling shadows per vertex instead will soften the edge simply because of the vertex to fragment interpolation but may introduce popping from shaded to lit particles.

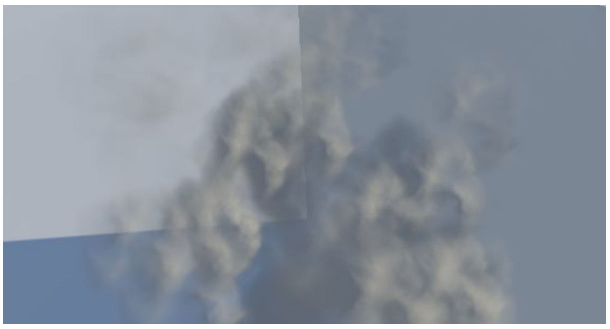
Here *tessellation* comes to rescue: Using a low tessellation value of 3 or 5 according to the particles' size will already improve shadows (and per vertex lighting as well) while keeping the extra costs low.



Per Pixel Shadows will reveal the simple billboard planes and create some noticeable diagonal edges between shaded and lit parts of the particles.



Using the extrude parameter you may break up these diagonal edges. Per pixel sampled shadow however still do not have soft edges.



Per Vertex Shadows create a rather smooth transition between shaded and lit parts of the particles (because of the vertex to fragment interpolation) but may pop. Using the tessellated version lets you create more vertices and smoother transitions.

Light Probe Lighting

Particles may sample ambient lighting from Light Probes. In order to make this work:

1. make sure there is a **Light Probe Group** in your scene and Lighting is baked.
2. make sure the **_LuxParticles_AmbientLighting prefab** is in your scene.
3. check **Enable Light Probes** in the inspector of all materials which shall sample probe lighting.
4. add the **LuxParticles_LocalAmbientLighting script** to all particle systems which shall sample probe lighting.

   If **Enable Light Probes** in the material inspector is checked but you have not assigned the **LuxParticles_LocalAmbientLighting** script or there is no **Light Probe Group** ambient lighting will be corrupted.

As soon as there is one single active LuxParticles_LocalAmbientLighting script in the scene, all Lux Particle shaders will sample ambient lighting as if it was set to *Skybox* (even if it is set to Color or Gradient). Default ambient lighting is sampled at the position of the game object which holds the *LuxParticles_AmbientLighting* script (most likely the _LuxParticles_AmbientLighting prefab).

If there is no instance of the LuxParticles_LocalAmbientLighting script in the scene but you may spawn one at runtime please make sure that you have checked **Always use SH** in the LuxParticles_AmbientLighting script on the _LuxParticles_AmbientLighting prefab to prevent light popping.

Mixing Light Probe Lighting and simple Ambient Lighting

You may mix particle systems using Light Probes and systems not using light probes, but in order to make sure that you will not get any popping in lighting when a probed particle systems gets active you should check **Always use SH** in the LuxParticles_AmbientLighting script on the _LuxParticles_AmbientLighting prefab.

Not using Light Probes on all particle systems has the advantage that updating the ambient lighting most likely will be cheaper.

Light Probe support is enabled per material, so you have to duplicate the material if you want to use the “same” material with and without Light Probe support.

Script Components

LuxParticles_AmbientLighting

This script (which you will find on the _LuxParticles_AmbientLighting prefab and must be added to all scenes) calculates and writes ambient lighting as global shader variables for all particles systems as Unity does not provide it due to the fact that the particle shaders use *“Vertex Lighting”*.

Inputs

Update per Frame If checked ambient lighting will be updated each frame. Leave it unchecked if your scene does not change ambient lighting over time. However in case you move particle systems which shall receive ambient lighting from Light Probes or you have animated emissive materials or animated lights you have to check *Update per Frame*.
The script will update probed lighting only for visible particle systems for performance reasons.

Always Use SH If checked the Lux Particle shaders will always sample ambient lighting from Spherical Harmonics as if *Environment Lighting* → *Source* was set to *Sky* — even if it is set to e.g. *Gradient*. Default ambient lighting is sampled at the position of the game object which holds the *LuxParticles_AmbientLighting* script (most likely the _LuxParticles_AmbientLighting prefab).

Please note: You have to add a Light Probe Group before you enable *Always Use SH*.

LuxParticles_LocalAmbientLighting

This script must be attached to all particle systems which shall sample ambient lighting from Light Probes. Find out more >

Inputs

Sample Offset Lets you offset the position at which interpolated ambient lighting from Light Probes will be sampled based on the transform of the particle system.

LuxParticles_DirectionalLight

This script will copy the cascaded shadow map into a rendertexture which then can be sampled by the particle shaders. You have to assign it to your primary directional light in case you checked *“Enable directional Shadows”* on any of the particles’ materials.

Limitations

Particle Render Modes

Lux Lit Particles have been written with classic camera aligned billboards in mind. Other billboard modes like *Horizontal Billboards* are only partly supported as ambient lighting most likely will be off if the particles' *Renderer → Normal Direction* is set to 1.0. Setting it to 0.0 however may give you better results.

Lighting

Lux Lit Particles use screen space normals which are pretty cheap to render and do not need additional vertex streams such as tangents but at the same time are not as accurate as tangent space normals. Negligible when it comes to particles i think.

The shaders expect the directional light being the most dominant light thus being passed as first by Unity. This however might not always be the case. So lighting may change if the directional light is rather weak and a strong spot or point light hits the particles.