User Manual

1. Introduction

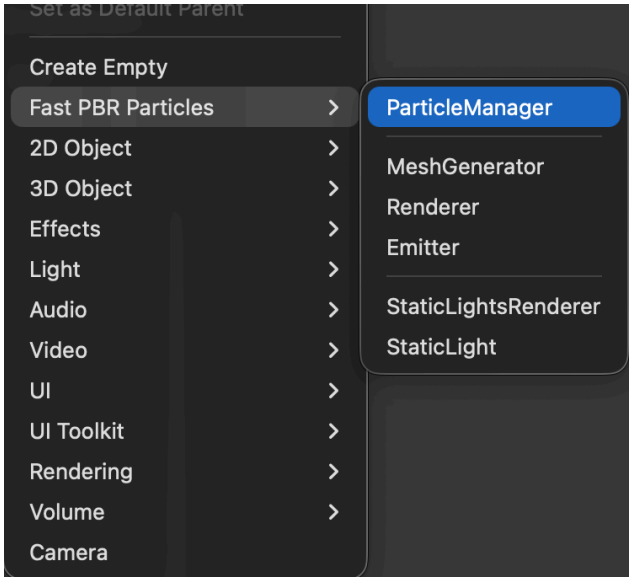
Fast PBR Particles is a mesh-based particle system that generates its geometry once and then evaluates all particle behaviour entirely in the shader, enabling cinematic effects with minimal CPU cost and excellent platform coverage.

Core principles:

- Minimal CPU workload: particle meshes are built once; there is no per-frame CPU simulation, all temporal evolution is evaluated on the GPU with single-draw-call rendering.
- Broad graphics API support: despite the simulation happening on GPU, it doesn't require any advanced features targeting nearly all mainstream graphics APIs including OpenGL(ES) 2.0, DirectX 9.0, and all modern API. while preserving the same feature set.
- Physically based imaging: motion blur, depth of field, inter-frame temporal control, and exposure are derived from physical optical phenomenons with strict energy conservation.

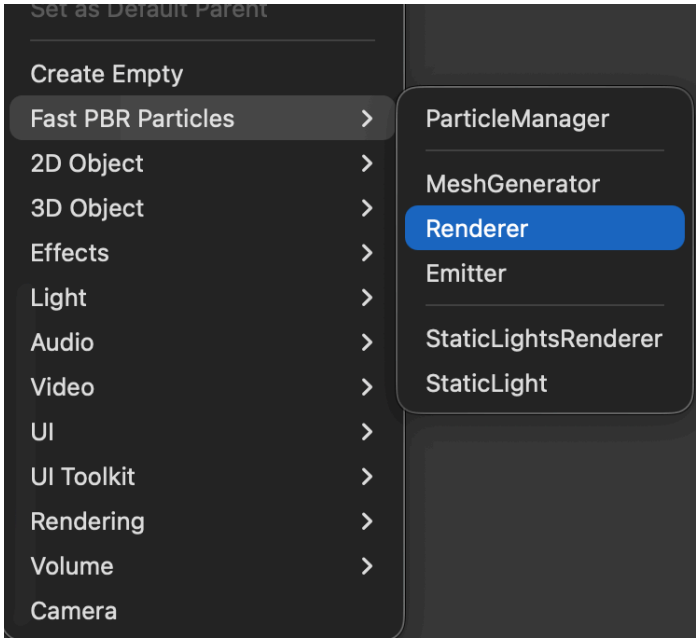
2. Creating a Particle System

Step 1 Create a Particle Manager on the Scene



Right click in **Hierarchy -> Fast PBR Particles -> ParticleManager**
Particle Manager is necessary object on the scene as it controls global properties of all particle systems

Step 2: Create a Particle Renderer



Right click in **Hierarchy -> Fast PBR Particles -> Renderer**
This Component will generate particle mesh on the fly and render it.

Step 3: Run and see it work

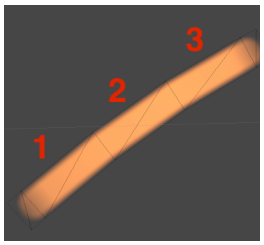
If it's hard to see adjust Exposure compensation of other parameters in Particle Manager depending on your scene.

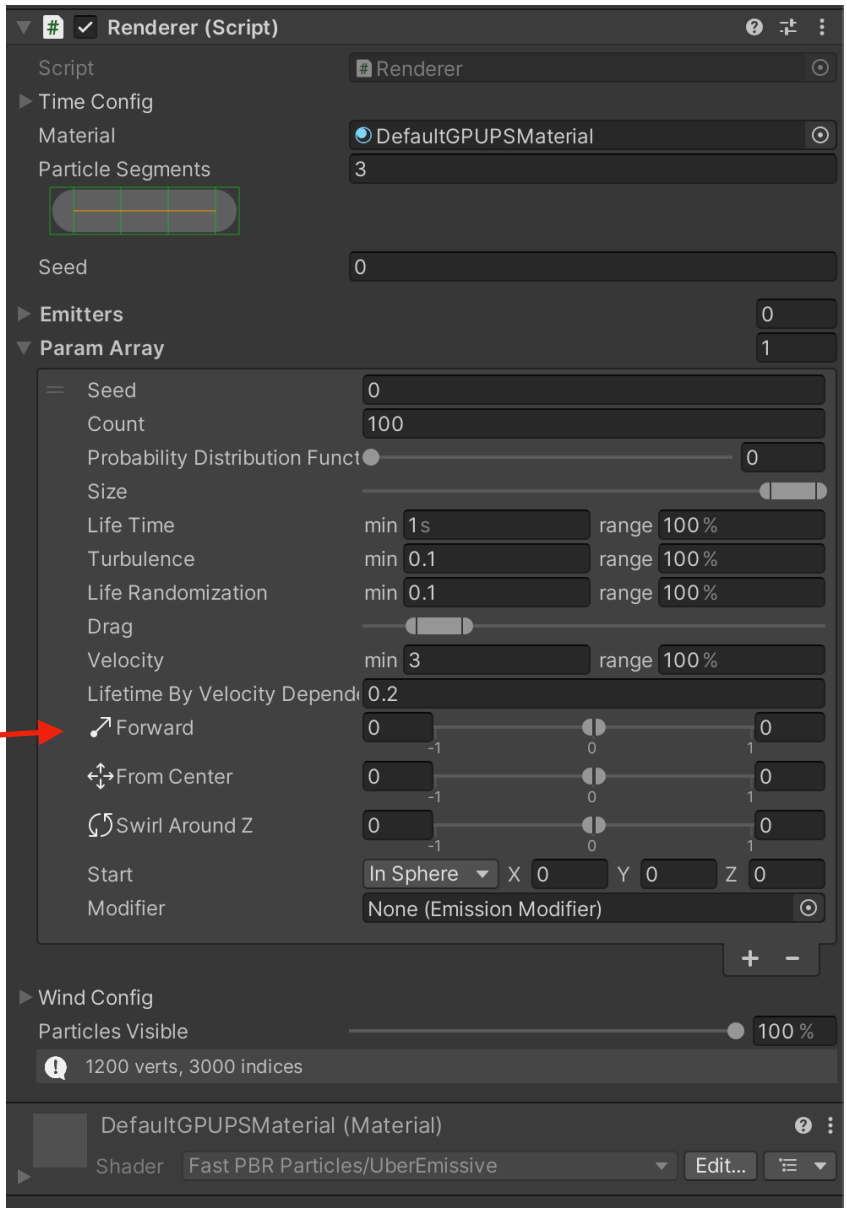
You should see something like this:

3. Component Overview

1. Renderer

Particle Segments defines how many segments each particle has, the more segments the smoother it will look between frames



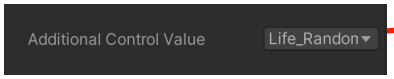


Param Array: contains sets of parameters for generating **Count** of particles.

Emitters: is the collection of objects with the same parameters from **Param Array.**

Size: is a range between 0 and 1 where relative particle sizes are chosen from and multiplied by shader **Max Radius**

Additional control Value (currently **Life Randomization**) depends on shader configuration



Lifetime By Velocity Dependence: controls dependency that makes lifetime shorter with higher speed

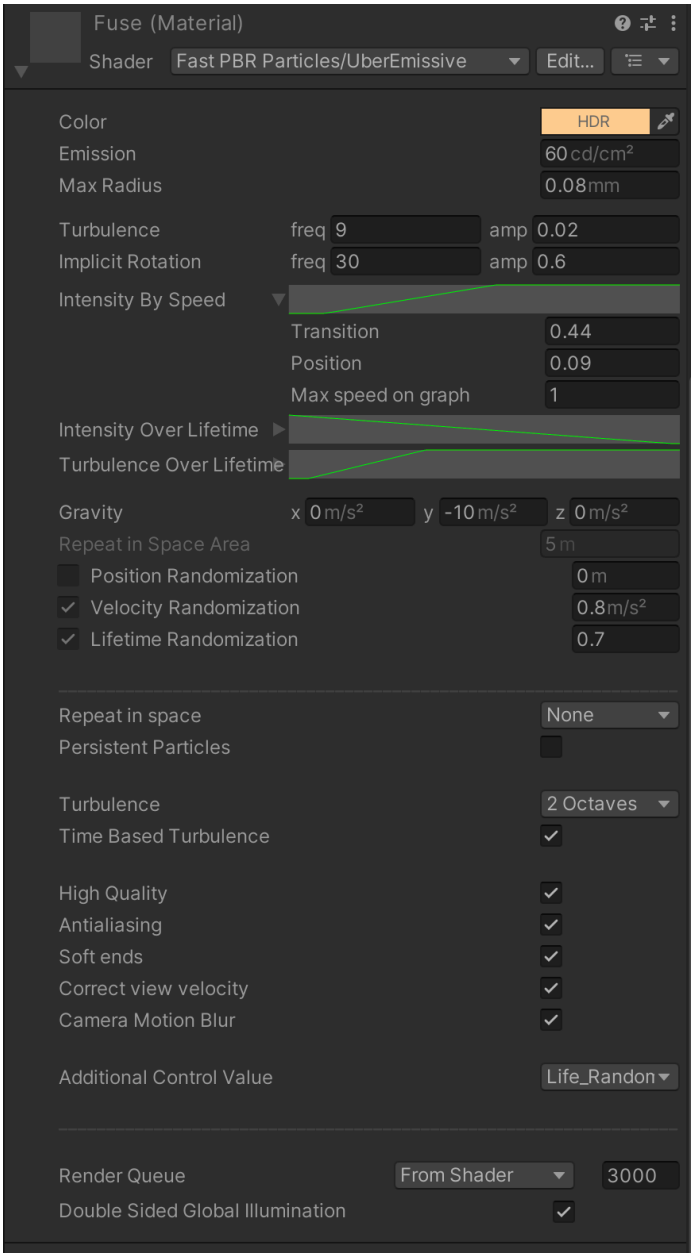
Forward, **From Center**, **Swirl Around Z**: Defines starting velocities of generated particles

Start: Defines starting position and its parameters

Modifier: Plug in custom code to modify or redefine generation of particles using burst jobs.

Particles Visible: Turns off Particles with a slider without regenerating mesh.

2. Emissive shader



Max Radius: will be multiplied by **Size** coming from Particle system.

Implicit Rotation: if a particle had an emissive size and cold dark side and was rotating, the emission would oscillate.

Intensity By Speed, Intensity Over Lifetime, Turbulence Over Lifetime: A set of very simple linear curves that control variables during a particle lifetime. Intensity By speed depend on the speed of a particle at a moment.

Gravity: should be a normal gravity unless you simulate a different planet, or microgravity. Don't try to use it to simulate wind, there's a dedicated set of Wind parameters for that.

Position/Velocity/Lifetime Randomization: will randomize these parameters every particle cycle without regeneration of a mesh. Toggle to the left indicates weather a certain feature is enabled in the shader(it's disabled when the parameter is 0)

Repeat in Space: will infinitely repeat particles in space, as camera moves. Meaning the particles will teleport in front of the camera from behind the camera, it's like a cube of repeating in space particles following the camera. (Doesn't regenerate the mesh)

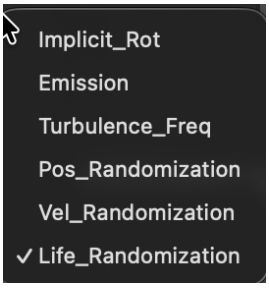
Persistent particles: particles will have no lifetime, and will never die.

Turbulence: Choose between: None, 1 and 2 octaves.

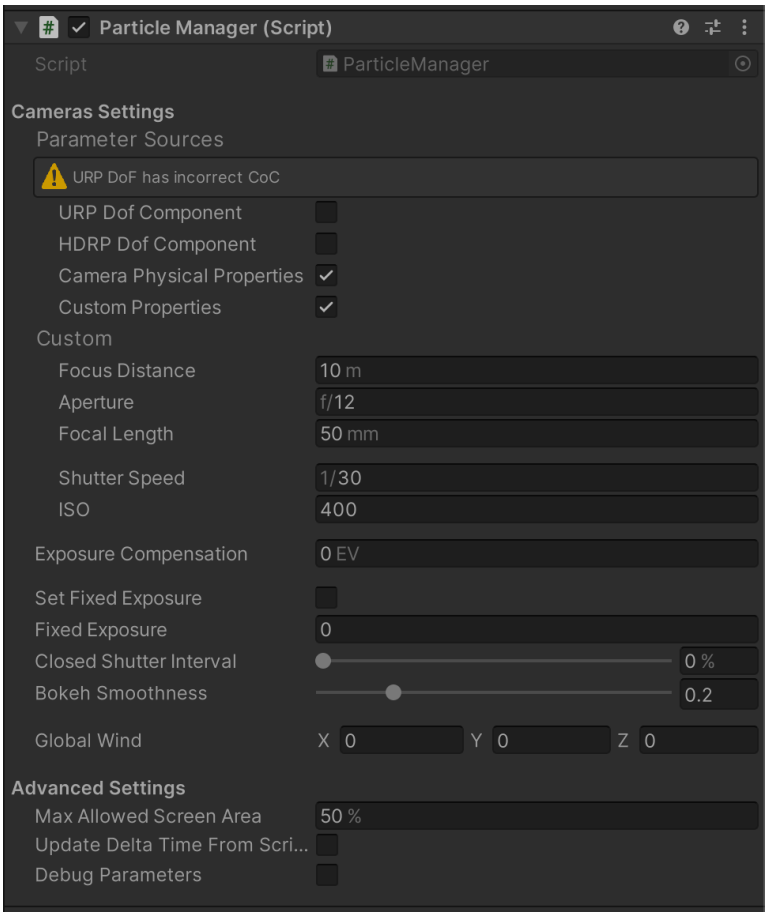
Time based turbulence: turbulent displacement will be calculated from time, *Otherwise* from space.

High Quality: more accurate computations for fragment shader
Antialiasing: more accurate when particles become at the size of 3 or less pixels.
Soft ends: for better temporal blending between frames.
Correct View velocity: motion blur calculations are slightly more accurate.

Additional Control Value: there's an additional value in particle data reserved for the features that are rarely used. Saves space by not having all of them at once.



3. Particle Manager



Parameter Sources: choose sources the system will gather parameters from in order from top to bottom. URP/HDRP components are on volumes, Custom Properties are below.

Custom: will be used when **Custom Properties** are on.

Sometimes you need **Exposure Compensation**

$$\text{Exposure} = C \cdot 2^{(EC - \text{EV}_{100})}$$

Where:

C = CameraProperties.CalibrationConstant

EC = Exposure Compensation

Fixed Exposure: will override exposure so it will no longer depend on properties such as Aperture

Closed Shutter Interval: sets the percentage of time of the frame the shutter is closed.

Bokeh Smoothness: add smoothness to bokeh it usually depends on the camera optics.

Global Wind: just as it suggests, you can also choose how much global wind each particle system receives.

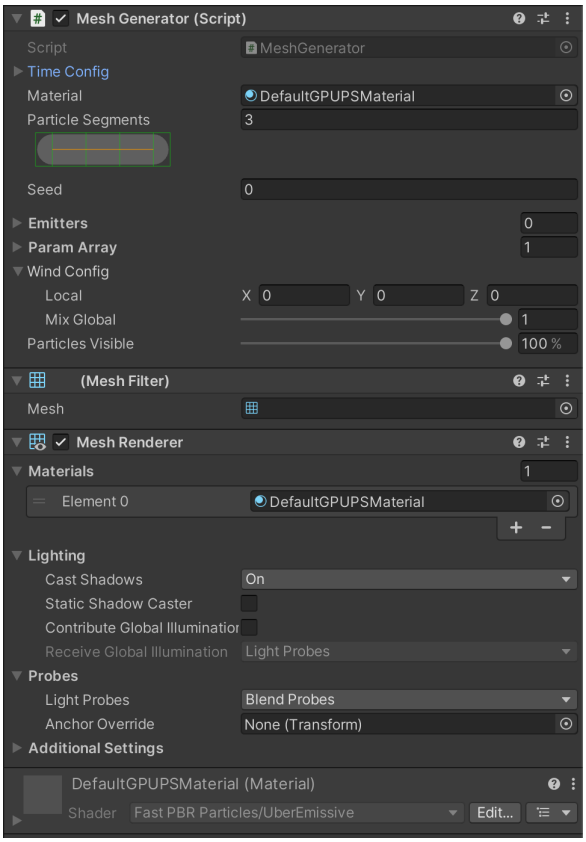
Max Allowed Area: this is a very important setting to reduce the overdraw in case if you have very large CoC(Bokeh) it will gently fade the largest ones based on this parameter.

Update Delta Time From Script: This will allow you to set your global delta time from script for example if you want deterministic behaviour, or turn time backwards.

Debug Parameters: will print final parameters gathered from **Parameter Sources** per camera. This is useful if you want to debug the sources of parameters.

4. Mesh Generator

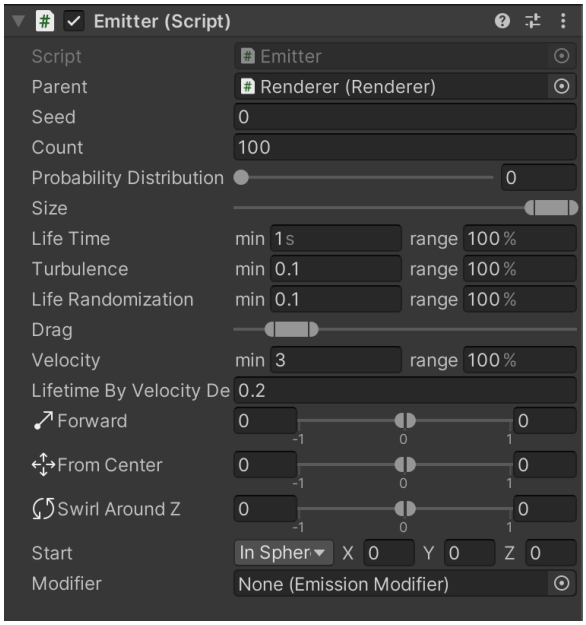
Works the same as Renderer, but instead of rendering particle mesh this one sets the mesh to a Mesh Filter for a better and more explicit control.
Everything else is the same.



5. Emitter

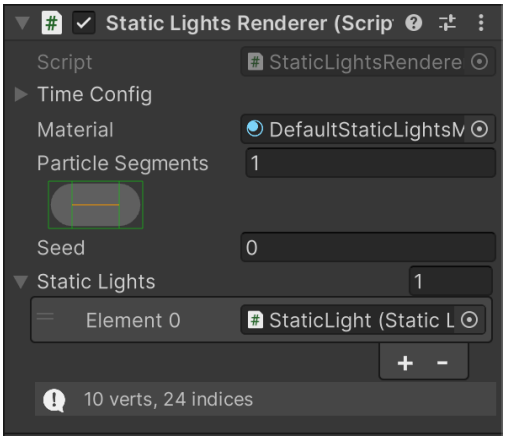
Emitter contains parameters for generation, the same as the ones from Param Array. The only difference is it also has transform from the object it attached to.

Put **Emitter** under **Renderer** or **Mesh Generator** to make it work.



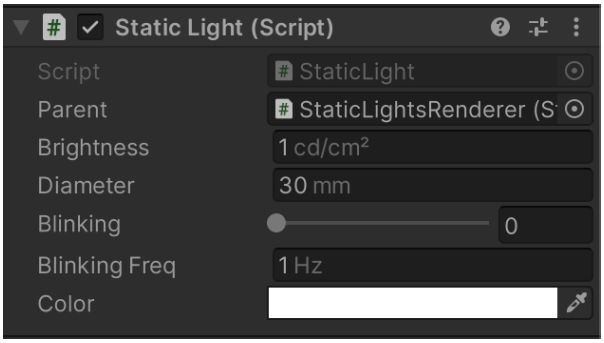
6.Static Lights

Static Lights Renderer is a similar component to **Renderer** but with less functionality and works with **Static Light** components.
I recommend to use only one segment as it's only needed for camera motion blur.



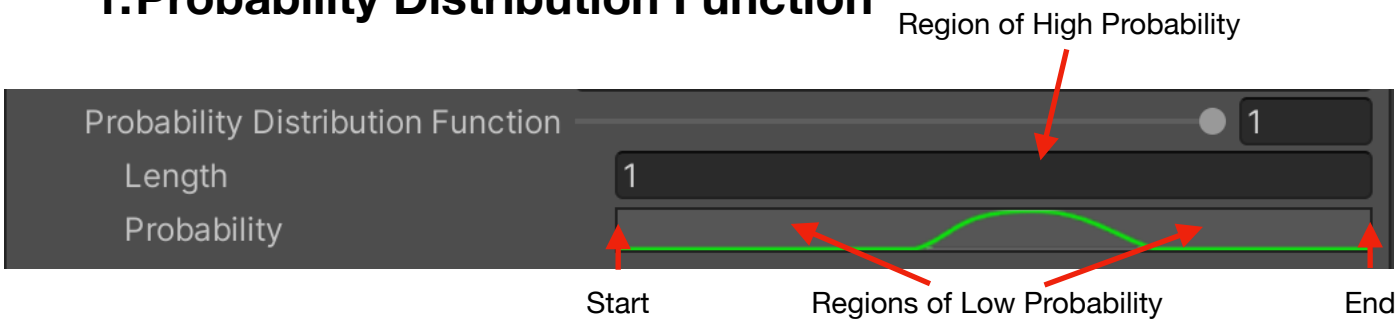
Static Light should be under **Static Lights Renderer** object in hierarchy.

Blinking: emulates shimmer effect due to heat haze.



4. Concepts

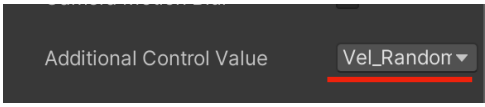
1. Probability Distribution Function



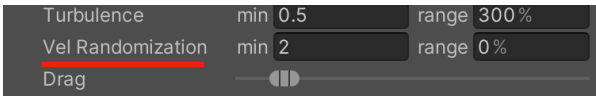
Probability Distribution Function controls when to emit particles. It locks lifetime properties such as **Lifetime** and **Lifetime by Velocity Dependence** because each particle has it's own loop based on its lifetime.
Length sets the lifetimes of each particle. And mix slider on top mixes between PDF defined probability and random probability.

2. Additional Control Value

I added an extra control value you can link to one of several behaviour available in shader, or even write your own shader to used it.



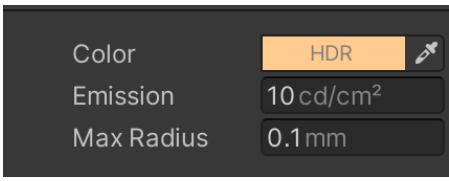
When you choose one in the shader, a label between Turbulence and Drag will change to reflect selected behaviour.



There are 6 predefined behaviours that this value can control.
- **Implicit Rotation**
- **Emission**
- **Turbulence Frequency**
- **Position Randomization**
- **Velocity Randomization**
- **Lifetime Randomization**

You can find each of those properties in the shader.
This value controls selected behaviour per particle.
Note that if for example Velocity Randomization in the shader is 0
the feature will be turned off and the control value will do nothing.
Also note that control values are multiplied by the corresponding values in the shader.

3. Size and Emission



You can find Emission and Max Radius properties in the shader,
But you will find only Size range(0;1) in the renderer.

This is because the Emission is $\mathrm{cd}/\mathrm{cm}^2$ and the total emission of the particle depends on square of it size πr^2 => $\text{Emission} = \pi \cdot \mathrm{cm}^2 \cdot \mathrm{cd}/\mathrm{cm}^2$
So the bigger the particle the brighter it is.

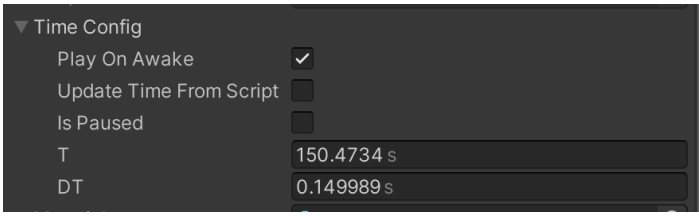
4. Repeat in space

Use this feature if you need particles to fill all the space around camera, such as rain, snow, etc.

If you select **3D** it will fill all the space around the camera, and when you move particles will be disappear from behind and reappear in front of the camera, all with a single draw call, no cpu processing or fancy graphics features, it will simply *frac(position - cameraPos)*.

If you select **Horizontally**, it will stay around the same position along Y axis.
Enable **Persistent Particles** to make them never disappear, useful for rain or snow.

5. Time Config

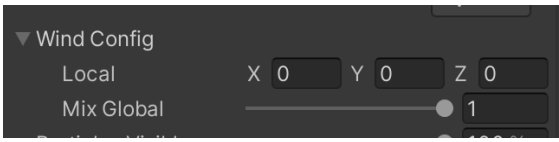


Time can be controlled from script, all fields are exposed.

Time and **Closed Shatter interval** are taken from **Particle Manager**, which also could be fully controlled from script.

Time could be stopped and rewinded per particle system or individually see **Determinism** example.

6. Wind



Wind could be set locally or globally from **Particle Manager**, it could be manipulated from script.
Mix Global sets how much wind is taken from **Particle Manager**, it's useful if you have some particle systems indoor.

7. Inter-Frame Temporal Control

1) Hard ends, no blending between frames

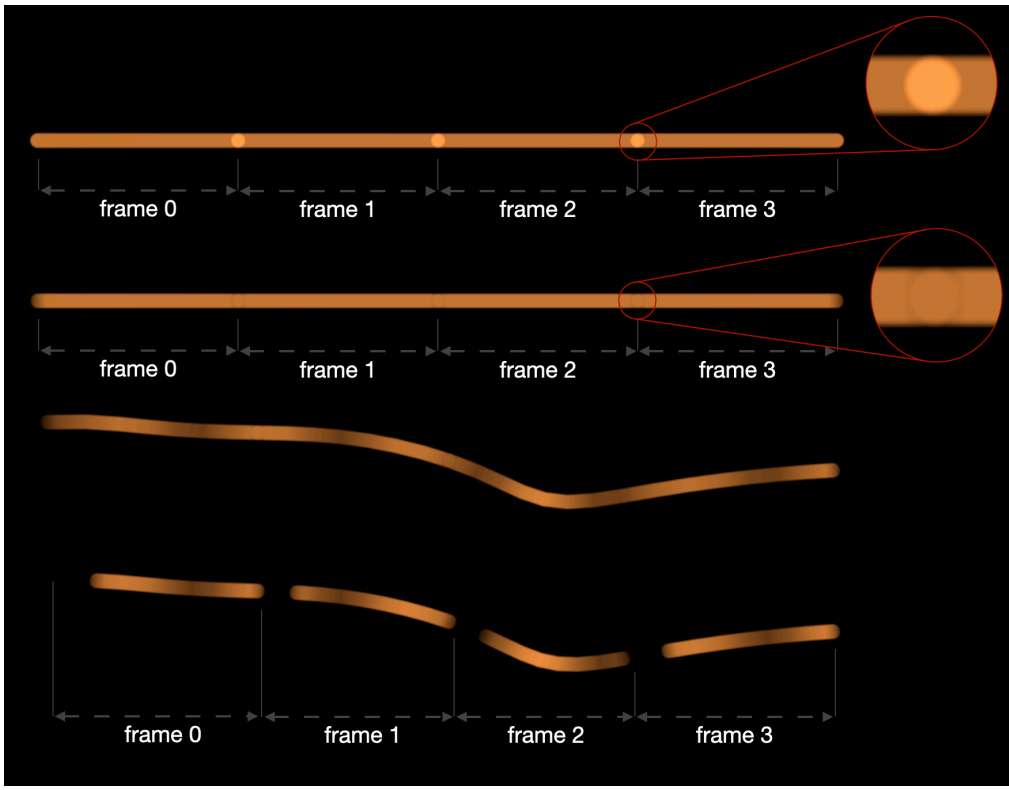
2) Soft ends, blending between frames is on

correct shape of the particle along the duration of the frame results in almost perfect result

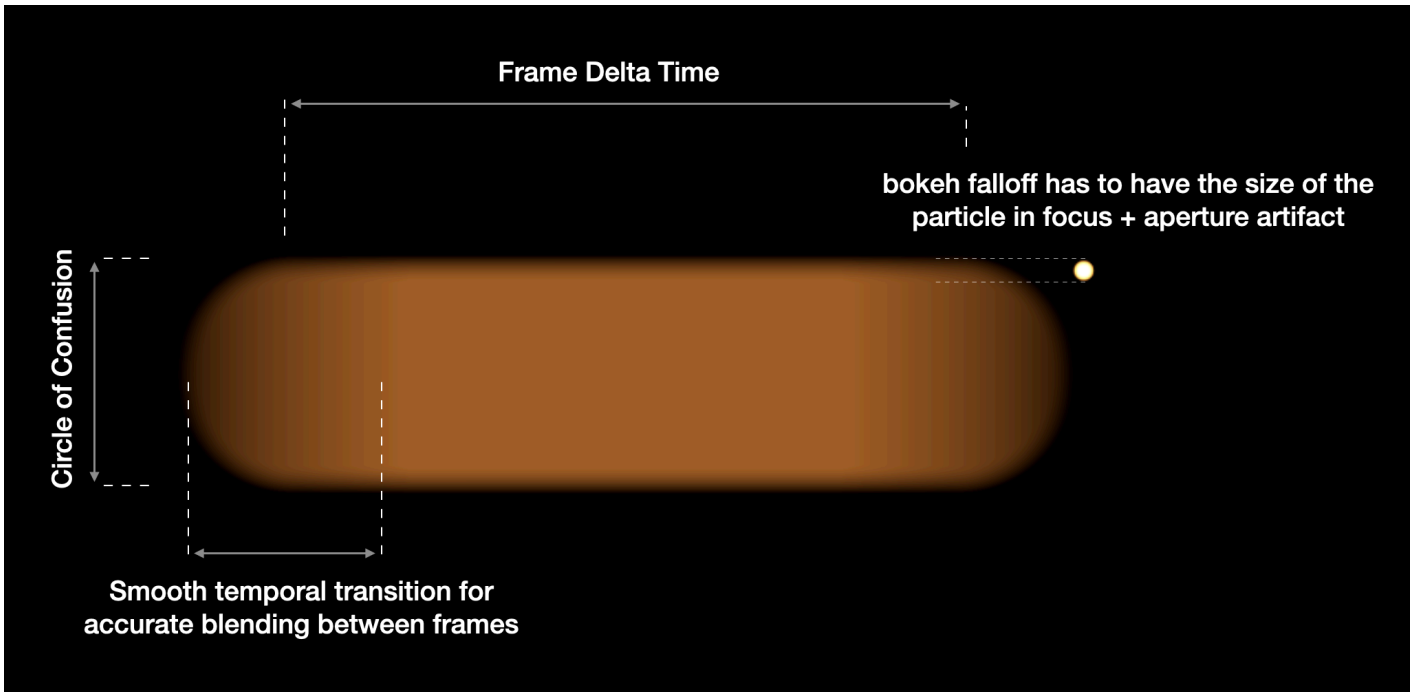
2) In between frames turbulence and implicit rotation of a particle added

Implicit rotation simulates a non-uniformly emissive particle rotating, resulting in temporarily variable emission

4) Shutter is closed for 20% of the frame



8. Temporal particle anatomy

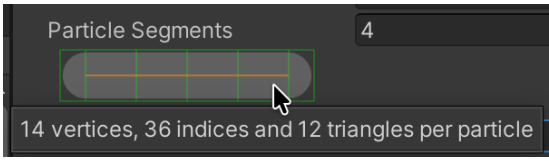


Size of the particle controls the size of the falloff of Bokeh. If you want to smooth it out further, check **Bokeh Smoothness** parameter in **Particle Manager**.

Also **Closed Shatter Interval** will make the particle shorter emulating the time the chatter was closed.

Smooth temporal transition comes from **Soft ends** option in the material

You can make a particle path curvier (it will look better during bending) by adding segments in the middle of the particle, be careful it will add more work for vertex shader



5. Extension

There are several ways you can extend functionality of this particle system Write EmissionModifier ScriptableObject and plug it into parameters. Or inherit from one of the classes and redefine any method.

1. EmissionModifier

Emission modifier has a method:

```
public abstract JobHandle Schedule(in Particles slice, in EmissionParams parameters, in RigidTransform tr, uint seed, JobHandle inputDeps );
```

You can use it to modify or redefine parameters using Burst Jobs.

See RandomBurstsModifier and ModifierChain from Examples/Scripts folder to see how it's implemented in practice.

2. Inherit from one of the classes

Classes you can inherit from usually very small, such as

ParticleManager < 200 lines,
Renderer, MeshGenerator < 100 lines,
StaticLightsRenderer < 150 lines.

Just read those classes they are very easy to read, they are by themselves examples of how to extend functionality. I tried not to make any field private and made everything virtual just in case.