UGS-ULTIMATE GPU Particle System

Document version v1.0
Code version v0.1b

You can find the latest documentation (with animated gifs) **online**.

Ultimate GPU Particle system is currently in Beta and will be fully released in January 2019. If you experience any bugs, you can check if it has been reported: **HERE**. If you want to report a problem or have suggestions, you can contact me via **support@maxproude.com**

Ideal use cases

How it works

Getting started

The editor
- General tab
- Emitter tab
- Emission tab
- Start values tab
- Lifetime values tab
- Forces tab
- Turbulence Tab
- Attractors Tab
- Render Tab
- Material Tab

Troubleshooting

Introduction

Ultimate GPU particle system is a GPU based particle system exclusive for Unity 3D. In contrast to most other GPU particle systems, it **does not** rely on **compute shaders**, which opens up most platforms. Create stunning visuals and breathtaking effects even on **Mobile** and **WebGL**. Ultimate GPU Particle System uses a technique called Pseudo-instanced particles. That means that the every particle in the system are always calculated and only shown if needed. That makes it very effective when you use many particles at once. It also has a stable impact on your framerate if used correctly.

Ideal use cases

Before you get started, it is very important to understand the advantages and disadvantages of GPU Particle Systems. There are two major performance costs involved:

1. Calculating Particles
2. Displaying Particles

The first case refers to the process of spawning and updating particles without actually displaying them. This part also uses a lot of memory, because all data is stored in half or floating point precision data, sometimes even double buffered. The good news is, that this is easy to test, because the load does not vary from particle system to particle system. The second case is a bit more complicated and it can have a varying impact on your framerate. The bottleneck here is the fillrate. In concrete terms, that means, that particles have to be small to reduce overdraw. This is especially important on mobile devices, that suffer dramatically from transparent overdraw.

You should be save if you consider the following steps:

- Particle count: As many as necessary, as few as possible
- Don’t stack GPU Particle systems (like Shuriken); Try to get the best out of as few emitters as possible
- Reduce overdraw: If you use many particles, make them small, because of transparent overdraw
- Turn off unnecessary features: By nature of this GPU particle system, if a feature is turned on, all calculations are done, no matter the value.
- Make sure the max particle count is set to an appropriate value; This controls the buffer size and it will use much memory!
- Don't instantiate and dont SetActive(true/false). Load the effect with a scene, pool it and put it somewhere out of sight. This is a common technique used to circumvent hiccups on instantiate/ setting active.

How it works

ULTIMATE GPU Particle System does not use compute shaders, but abuses the fact, that Render textures can store data in form of colors. In this case, render textures are used as a buffer that store information about particle life, size, velocity, position etc. Each pixel represents a single particle and can be read by the pre-created geometry that references the correct 'buffer address' in the UVs per vertex. That way it is possible to use triangles, quads or any other geometry. The Render textures are updated using classic HLSL shaders and using post process techneques: Graphics.Blit();

Meta buffer

This buffer contains information about a particle birth time (R), lifetime (G), start size (B) and start rotation (A).

Position buffer

The XYZ-Position information is stored in the RGB channel of the texture. Additionally, a random seed value is stored in the A channel.

Velocity buffer

The velocity buffer stores only information about a particle's velocity in XYZ direction in the RGB channel

All buffers are double buffered, because pre-DX12 hardware does not allow for random access memory.

Shaders

All steps are calculated by a shader, that changes by turning features on and off. It is therefore wise not to leave any unused features turned on! To further increase performance, conditions are linearly interpolated. This reduces unexpected framerate fluctuations.

Getting started

To create a new GPU Particle System, go to the **toolbar** and click **GameObject>Effects>GPU Particle System** This will add a new blank GPU Particle System to your scene.

The editor

The editor is the interface the artist has to use to create effects. It is heavily inspired by Unity Shuriken's particle system and replicates a lot of features and styles for ease of use.

General tab

As the name says, this is a tab that you usually set up once and keep for the rest of your time editing your effect.

Effect Length

This is the duration of the effect. If looping is turned on, this value is used to determine the progress of the emitter.

Loop

Will the effect only play once?

Play on Awake

Will the effect start playing automatically?

Simulation space

Are particles relative to the world or to itself? This will determine, if particles move, rotate and scale with its gameobject or not (local).

Buffer size

This is crucial for the amount of particles but also for the amount of memory used. You can change the buffer size to a value of your liking, but is recommended to choose a power of two value. Additionally, you can choose the buffer precision to save memory. Float precision is a default value, but you can change it to Half precision to effectively half the memory consumption. This can lead to some precision errors like e.g. imprecise particle position and lifetime.

Fixed delta time

The delta time is used to make move particles per time and not per frame. IN some cases even the slightest change in delta time can lead to some unwanted hiccups in the particles animation (especially in the editor). To avoid this, you can choose to simulate with a fixed delta time.

Emitter tab

Shape

Different effects require different emitter shapes. Currently, Point, edge, circle, hemisphere, sphere, box and cone shapes are supported. Mesh emitters will follow in later releases.

Point

Does not have any options

Edge

Change the length of the emitter

Circle

Additionally to the radius, you can choose to emit particles only on the edge or on its surface

Box

This is a typical Box emitter that has a width, height and depth.

Hemisphere

A hemisphere with a changeable radius.

Sphere

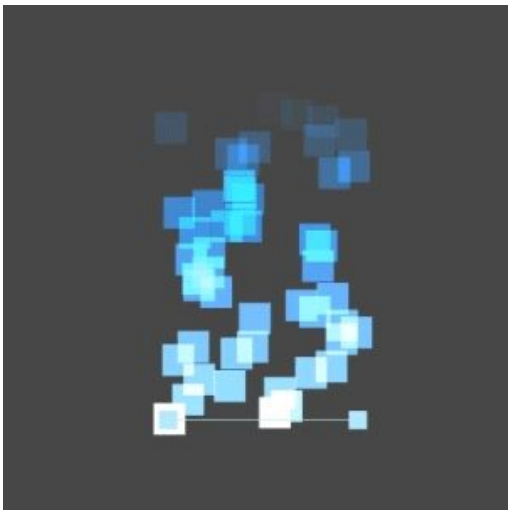
A sphere with a changeable radius.

Cone

The cone emitter is probably the most common and versatile emitter and can be used for many effects. Change the base size, angle and length to make the most of it. You can also choose to emit from its base and/or from its surface.

Emission tab

Currently only two types of emissions are available: static or curve. *Burst emission features will be available soon. There is also no sub frame interpolation at this time.*



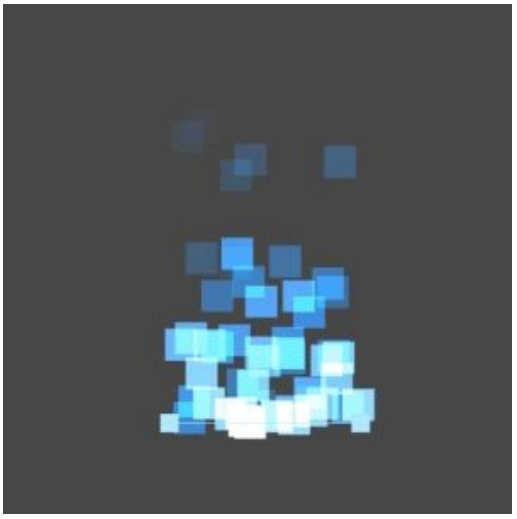
Fixed speed, lifetime, rotation and size. Only color over lifetime is set.

Start values tab

This tab lets you set particle properties at the time of birth. You can choose between a static value, random between two static values, a curve or random between two curves. These curves describe a value or values over emitter lifetime. That means that the initial value is set by the emitter.

Start speed

The gif demonstrates random speed between two values (Min 0.5 - Max 2.0) in seconds.



Random start speed: 1.0 - 2.0

Start lifetime

The gif demonstrates random length in lifetime by randomly choosing between two values.

Start size

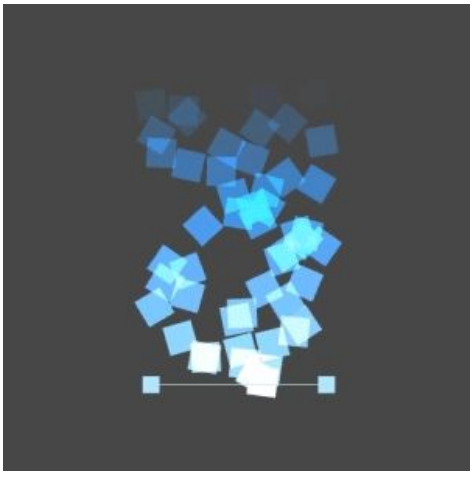
The gif demonstrates random particle sizes between two values.

Random start speed: 1.0 - 2.0

Start Rotation

This gif demonstrates the effect of random start rotation. At this point, only one rotational axis is supported. To use rotation over lifetime, you also have to enable start rotation!

Random start size: 0.5 - 1.0



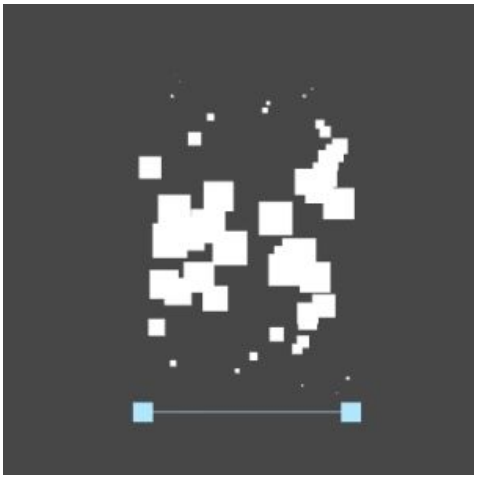
Random start rotation: 0.0 - 360.0

Lifetime values tab

Over lifetime values refer to parameters that are calculated over particle lifetime (except max velocity and intensity). In order to make this possible, linear, smooth and curve values can be chosen. The linear and smooth curves are simplified and are also supported on mobile. The current implementation of curves require arrays that are not supported on some mobile devices (e.g. Samsung).

Size over lifetime

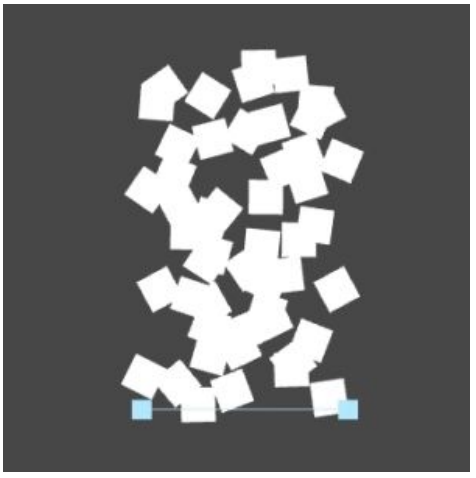
Define a curve that represents the size of a particle over its lifetime.



Size over lifetime: Bell shape

Rotation over lifetime

Define a rotation over its lifetime in degrees.



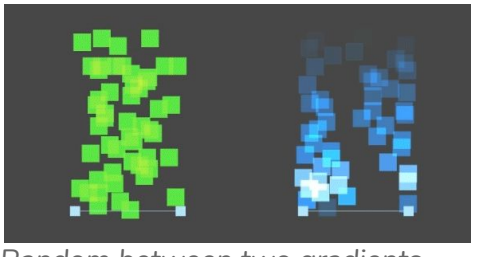
Rotation over lifetime: Bell shape

Color over lifetime

Choose a color, two colors, a gradient or two gradients to make particles change color (randomly) over their lifetime.

Color intensity over lifetime

This modies the intensity of the color in HDR. Values bigger than 1 make colors brighter. Values between 0 and 1 make the color darker. This is particularly useful with bloom post effects.



Random between two gradients

Max velocity

This clamps the velocity to a given value.

Forces tab

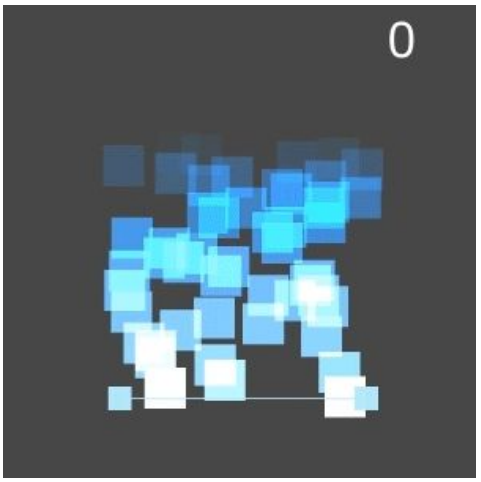
Forces are a great way to modify particle movement and make their movement more dynamic.

Gravity

Moves particles always in -Y direction. Note that -Y changes when switching from world to local simulation space!

Air resistance (drag)

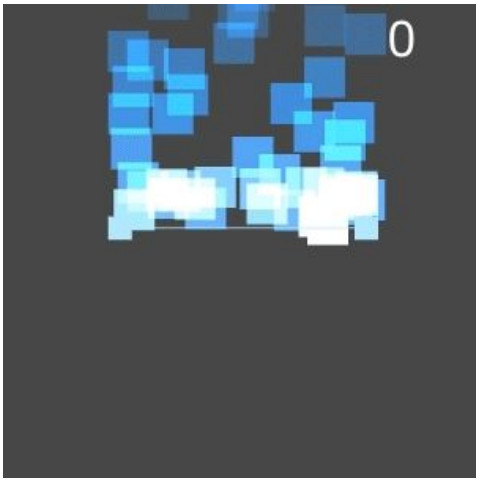
Air resistance makes particles constantly slower. This is particularly useful in conjunction with effects that accelerate particles constantly (e.g. Turbulence)



Air resistance

Inherit velocity

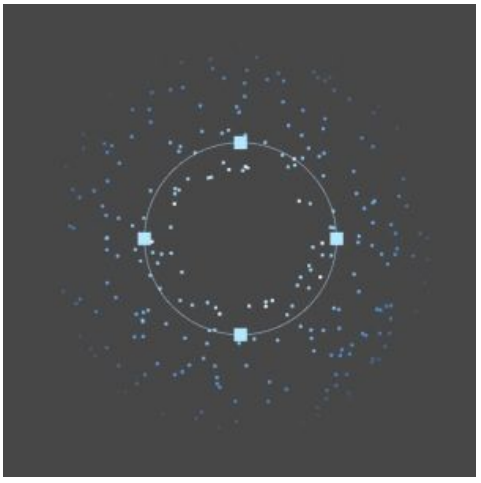
Makes particles move in the same direction as tits emitter. A multiplier defines how fast particles will go after emission. Positive values will follow the emitter, while negative values move particles in the opposite direction. Please note, that Ultimate GPU Particle System does not support sub frame interpolation for particle emission, which may result in gaps when the emitter is moving very fast.



Gravity

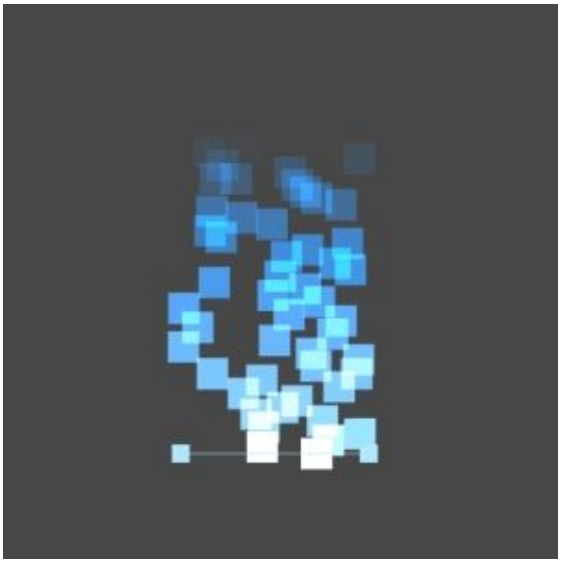
Circular force

Rotate particles around a centerpoint transform. The transform can be replaced and moved to offset the center point. All 3 Axis are supported.

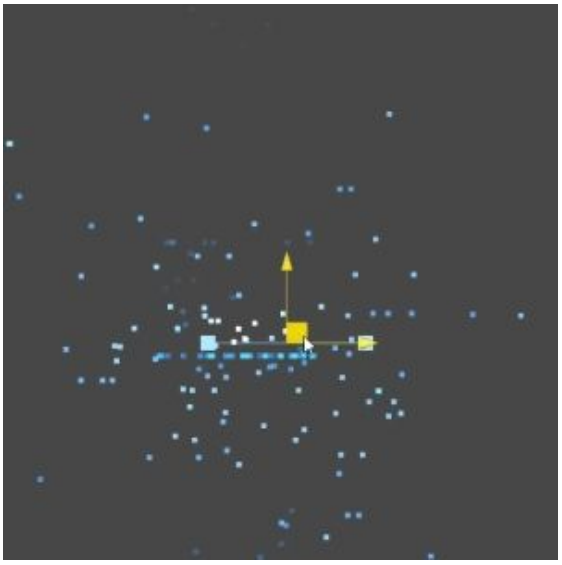

Circular force

Directional force

This is a constant force that is applied to particle velocity on every frame. Changes with simulation space.



Directional force



Inherit velocity

Turbulence Tab

Ultimate GPU Particle System supports two types of turbulence: 2D turbulence and vector fields (3D Texture). The 2D turbulence uses a noise texture while vector fields only can be imported from .fga files. Both types have very simple but powerful controls:

Amplitude

The amplitude controls the strength of the turbulence.

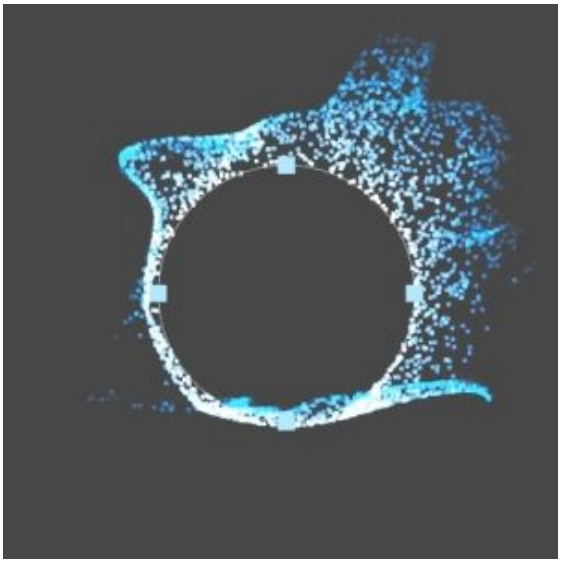
Frequency

Small values make small waves while higher values make bigger waves.

Offset (Speed)

To make the turbulence evolve over time, change this value. it effectively offsets the noise (2D) or rotates it (3D).

Vector fields are not supported on mobile and WebGL



2D, high frequency + Offset

3D, low frequency + Offset

Attractors Tab

In this tab you can create up to 4 attractors. Attractors attract or repel particles towards the center of the emitter if no other transform has been defined.

Rendering Tab

In this tab, you can define the shape of particles: Point, Triangle, Billboard, Horizontal Billboard, Vertical Billboard, Stretched Billboard, Tail-Stretched Billboard and Mesh (soon).
When using Point particles, please test on your target devices, because not all platforms support point mesh topology.
Stretched and Tail-Stretched Billboards have a modifier, that multiply the stretch length.

Material Options Tab

In this tab, you can change the particle texture, the blend mode, Z-buffer usage and render queue.

Troubleshooting

UGS is still under development and new features will be added while bugs are being fixed. If you find a bug, you can check this **document** if the same or a similar bug has been reported. If you want to report a new bug, please contact me via **support@maxproude.com**

I will do my best to fix all bugs as soon as possible.