UGS-ULTIMATE GPU Particle System

Document version v1.5
Code version v1.1

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

Ultimate GPU Particle system finally is out of Beta, with Beta/ Preview features still in place. Those features will gradually be fixed and improved, with more features coming in the future. 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 **m4xproud@gmail.com**

Patch notes

v 1.1

This release mainly fixes bugs and other problems with the Particle System.

- Added trails to the particle type
- Added proper normals to mesh emitters
- Fixed particle system layering
- Fixed creeping number issue of emitter parameters.
- Misc fixes

v 1.0

Version 1.0 ends the Beta phase of Ultimate GPU Particle System and introduces a few new features as well as bug fixes:

- Collision with depth (Preview)
- Refraction
- It is not possible do duplicate the particle system
- Mesh emitters now emit in normal direction instead of away from its pivot
- All properties are now initialized properly on start
- shader_feature has been replaced by shader_feature_local and don't add to the global shader keyword limit anymore

v 0.21b

Version 0.21b brings new Usability features to the Hierarchy as well as the Scene:

- Hierarchy buttons + Icon
- Scene Gizmos

v 0.2b

The latest patch comes with a variety of new features and fixes.

Features

- Collision with planes
- Mesh Particles
- Mesh Emitter
- Mesh Target
- Texture emitter
- Surface shader support
- Burst emission
- Flip book with motion vector blending
- Usability & Editor features

Fixes

- Attractors now work properly with strength and attenuation
- Time management has been updated. Start, Stop, Pause now works properly
- Hiding and showing the correct transforms in the Hierarchy

v 0.3b

The latest patch comes with a variety of new features and fixes.

- Skinned mesh emitter support
- Replaced shader_feature with multi_compile -> Massive compile time improvement
- Editable bounding boxes

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 techniques: 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. When using Trails, this changes to amount of trails and segments.

Axis aligned bounding box

Click 'Edit' to edit the axis aligned bounding box. The rotation will be set to 0,0,0 and gizmos will show until you hit 'Apply'. To check the bounds without editing, press "Show".

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.

Skinned mesh emitter

The skinned mesh emitter takes a skinned mesh and duplicates it. The mesh is modified by splitting every triangle and making the mesh renderable by a shader, that outputs world positions instead as a color. The result is a position buffer with evenly spaced triangles and particle density. It also doesn't require big buffers or creates any CPU overhead. It is possible however, to optimize the mesh by using a low poly version or by creating dedicated emitter geometry.

Emission tab

You can define a constant or a curve to define the rate at which particles are emitted. Bursts define a one-time emission of particles that have a range of particles that will be emitted at random (min & max) and a probability factor which is useful for looping particle systems (e.g sparks).

Start values tab

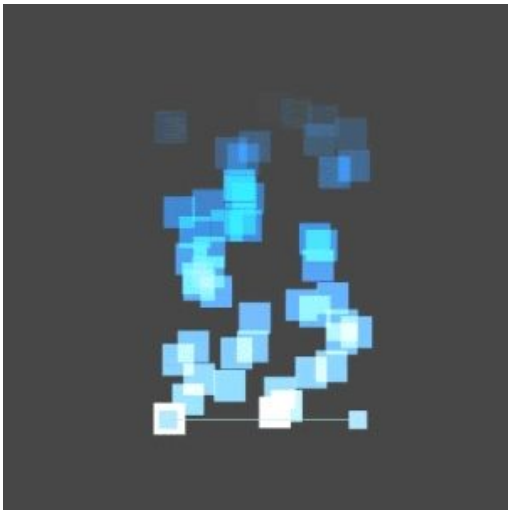
This tab lets you set particle properties at the time of birth. You can choose between a static value, random between two static values, a curve or random between two curves. These curves describe a value or values over emitter lifetime. That means that the initial value is set by the emitter.

Start speed

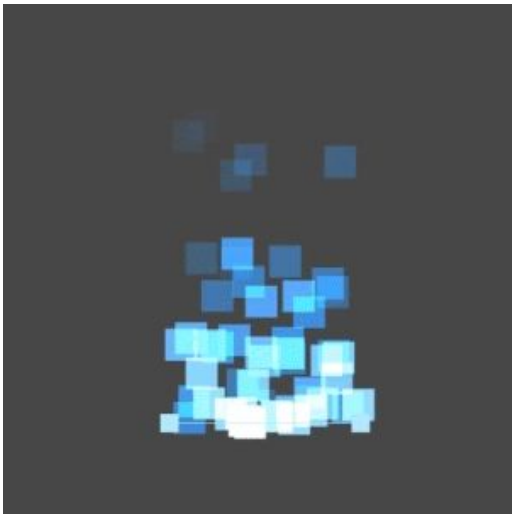
The gif demonstrates random speed between two values (Min 0.5 - Max 2.0) in seconds.

Start lifetime

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



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



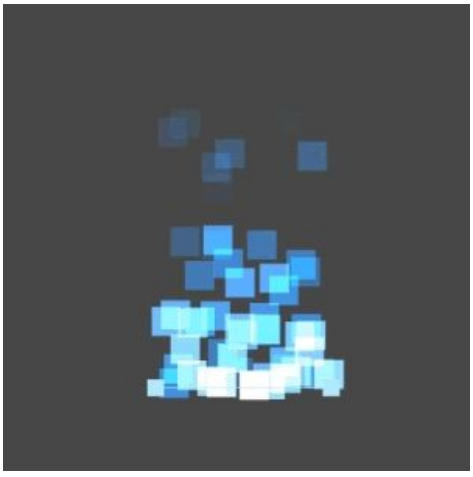
Random start speed: 1.0 - 2.0

Start size

The gif demonstrates random particle sizes between two values.

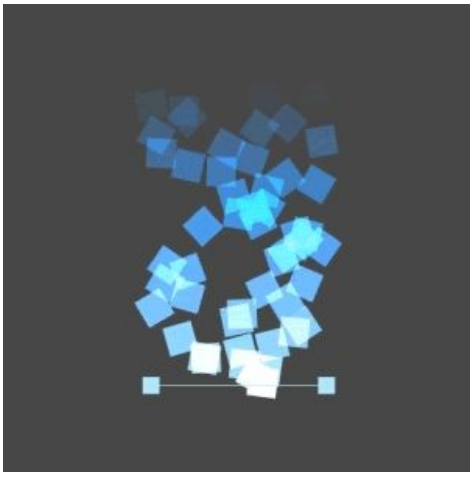
Start Rotation

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 speed: 1.0 - 2.0

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.

Rotation over lifetime

Define a rotation over its lifetime in degrees.

Color over lifetime

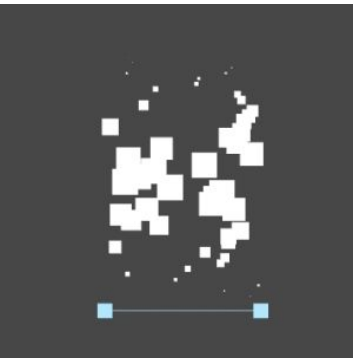
Choose a color, two colors, a gradient or two gradients to make particles change color (randomly) over their lifetime.

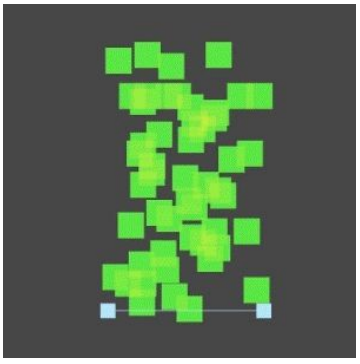
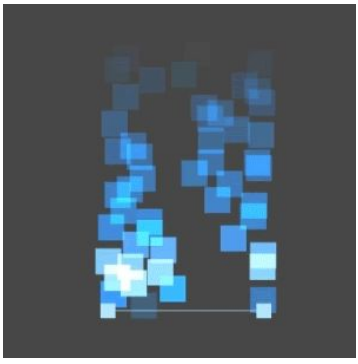
Color intensity over lifetime

This modifies 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.

Max velocity

This clamps the velocity to a given value.



Size over lifetime: Bell shape; Rotation over lifetime: Bell shape; Random between two gradients

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)

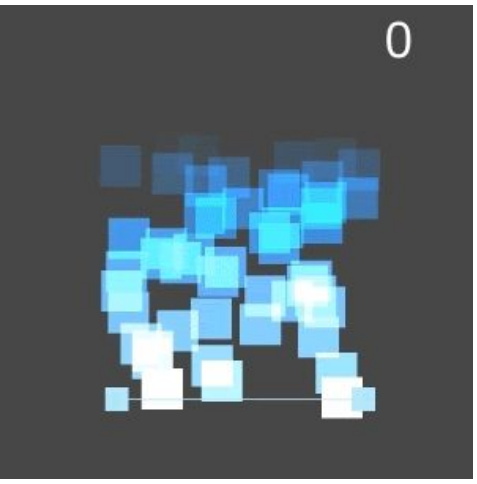
Air resistance makes particles constantly slower. This is particularly useful in conjunction with effects that accelerate particles constantly (e.g. Turbulence)

Inherit velocity

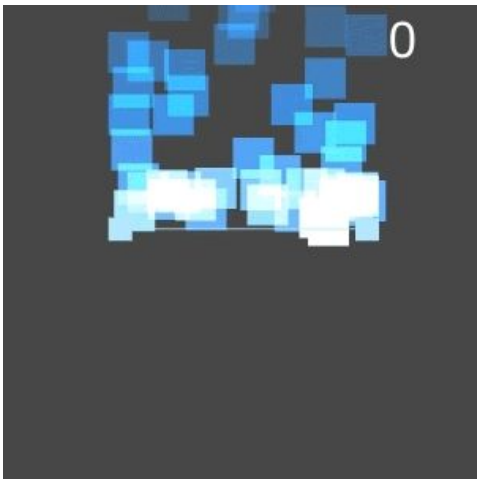
Makes particles move in the same direction as tits emitter. A multiplier defines how fast particles will go after emission. Positive values will follow the emitter, while negative values move particles in the opposite direction. Please note, that Ultimate GPU Particle System does not support sub frame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

Circular force

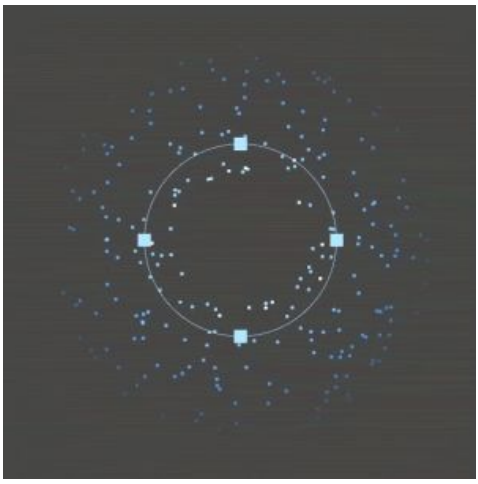
Rotate particles around a centerpoint transform. The transform can be replaced and moved to offset the center point. All 3 Axis are supported.



Air resistance



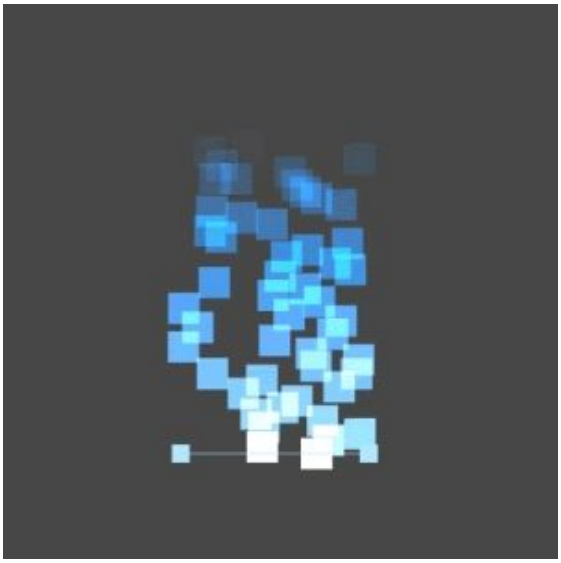
Gravity



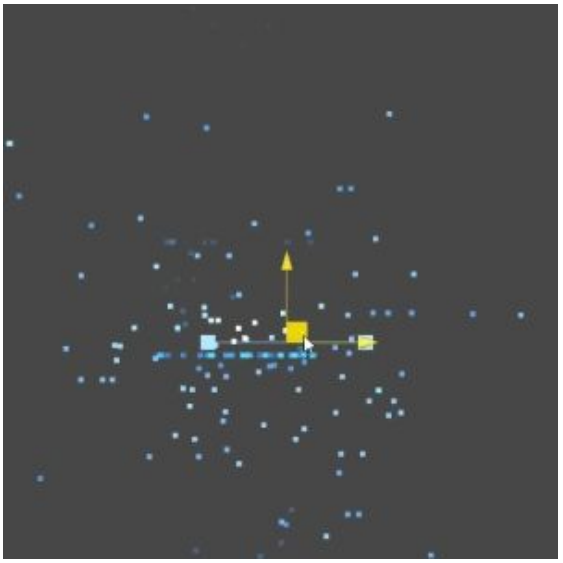
Circular force

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.

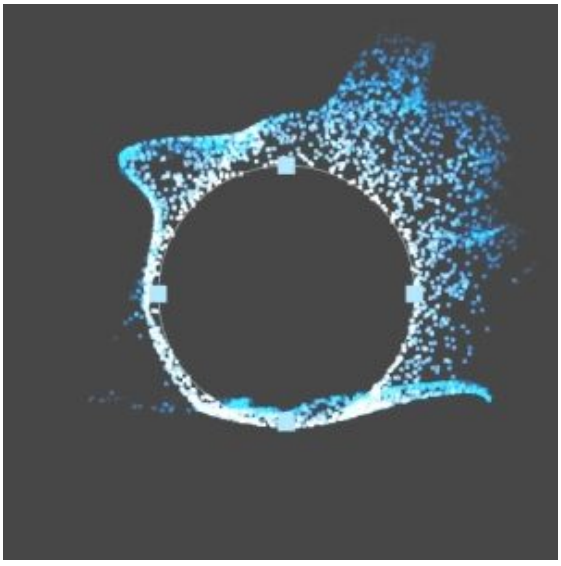
Frequency

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.

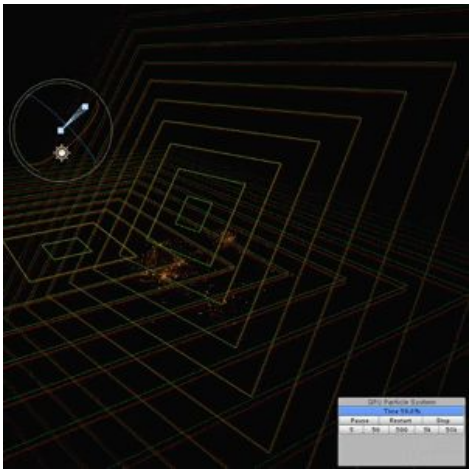
Mesh Target Tap

In this Tab you can define a mesh or a mesh filter that is used to create a mesh target. If you are using a mesh filter, that mesh filter's transform is used instead of the particle system's transform.

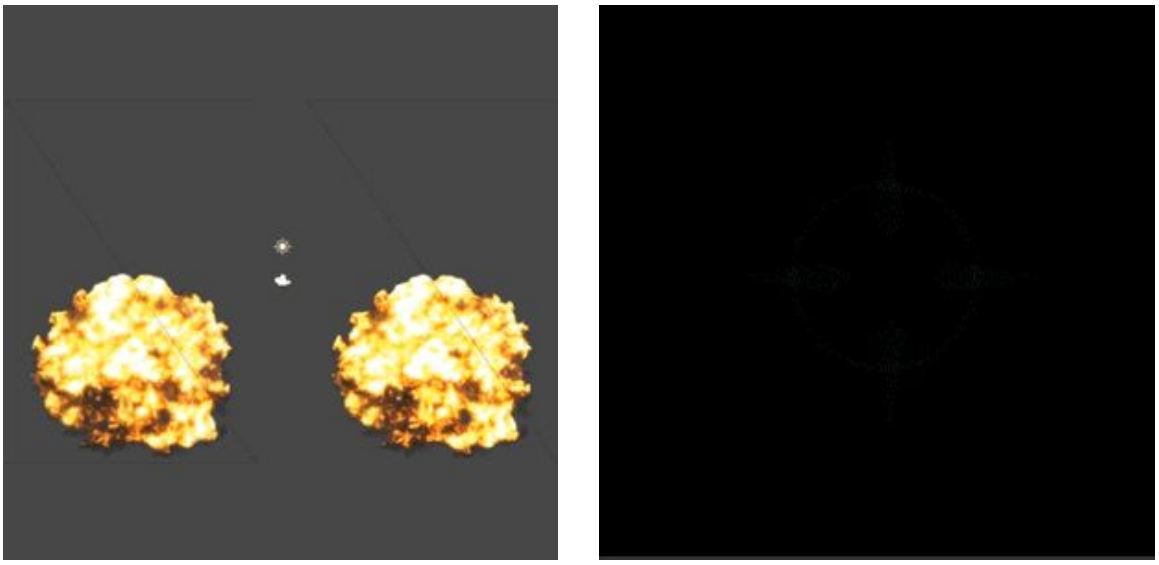
Collision Tab

Ultimate GPU Particle System supports collision with up to 6 planes. Every plane has a green and a red side, which represents the space where particles can and cannot be. Please make sure, that particles are emitted on the correct side, otherwise they will constantly collide. After creating a collider, you can create or assign a transform that represents the collision plane. To enable Depth Collision choose Collision Type: Depth in the dropdown list. Make sure, that you have a Camera assigned to render the depth Texture. Only objects in view of that Camera can collide. Damping controls the amount of momentum lost on collision. Collision distance defines the thickness of the depth buffer, to prevent fast moving particles from going through an object without colliding. Please note, that the depth collision feature currently is in preview only.

Portal with collision



Collision editor



Flipbook sprite sheet with advanced frame blending (motion vectors) & Mesh emitter and target

Rendering Tab

In this tab, you can define the shape of particles: Point, Triangle, Billboard, Horizontal Billboard, Vertical Billboard, Stretched Billboard, Tail-Stretched Billboard, Mesh and Trails.
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 multiplies the stretch length.

Material Options Tab

This tab controls the Material and the Shader used to display the particles. You can choose to use an unlit, a surface shader or a refraction shader, change the blending, Z-write and add textures. Additionally, you can also set up a flip-book for sprite sheet animations with motion vectors.

Light mode: Choose lit refraction or unlit mode in this tab. If you choose Standard, you will be using a surface shader, with PBS properties: Metallic, smoothness, normal map and emission map. Each features is being turned on and off based on assigned textures.

Standard light mode does not support blending and point lights at the moment.

Refraction is a variant of the unlit shader, but is better suited to refract the scene background. The Refraction map is packed like this:

R: Left Right distortion; **G**: Up Down distortion; **B**: Alpha

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.

Linear color space setting

At the moment linear color space in Unity is not supported, because Unity treats float precision render targets as colors. That has an influence on the particle's position and velocity (amongst other values), which are no longer calculated accurately. This results in strange particle behaviour, such as turbulence only moving particles in a single direction.

Trails

The latest update added trails as particle shapes. The amount of trails can be defined in the generals tab. When switching to trails, the buffer width and height will display trail count and segment count values. segment count has to be predefined and is the same for every trail. The length of a trail is defined, by follow speed (Render settings) and the speed of the first trail segment. Every segment follows its preceding segment. The first element behaves just like a particle and can be animated accordingly.

Changing the length of a trail:
The length of a trail can be changed, by increasing or decreasing the amount of segments in a trail and the follow speed value. If the follow speed value is very high, but the trail does not shorten, try reducing the amount of segments.

Post effects

Version 2.1.7 of Unity's post processing has been used to create the effects in UGS. These effects are not automatically added to the project and you have to do so manually in order to see the demo scenes with the correct vfx:

Window/Package manager/Post Processing

Make sure you can see all packages!