Simple UGUI Particles

Table of contents

Requirements & Setup

Requirements

Unity 2021.3 or higher is required since that is the min version Unity allows for new assets in the store. However it may work just fine in older versions of Unity with minor changes.

Introduction

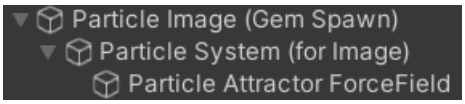
This section will help you understand what is going on in the hierarchy.

The particles are controlled by three objects:

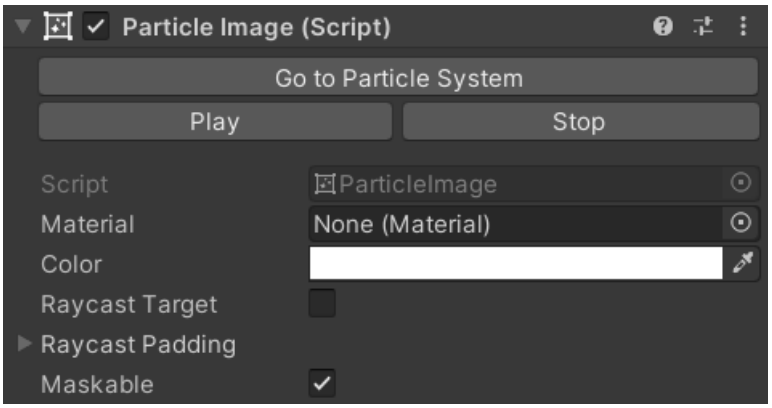
- The ParticleImage custom control in the UI
- The ParticleSystemForImage component in the scene
    - The ParticleSystem in the scene (tightly coupled with the ParticleSystemForImage)

The tool uses a normal Unity particle system and copies some (not all) of its properties into the UI Particle Image. To keep up with the current state of the particle system it has to do this every single frame. Don't worry, it's highly optimized.

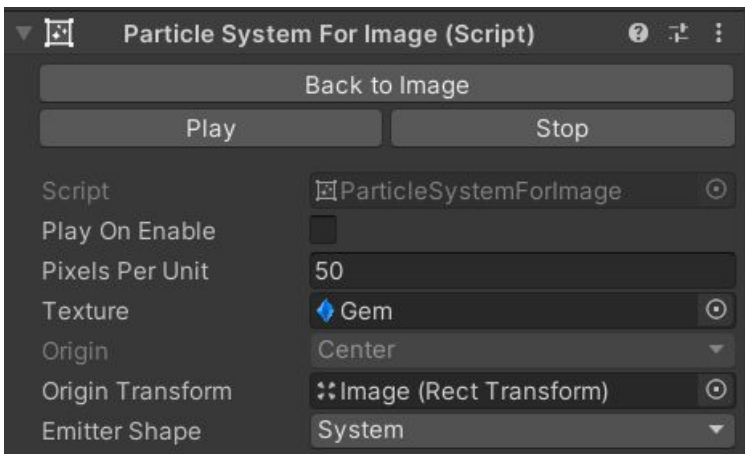
In the hierarchy a particle image looks like this (the attractor is optional and only added if you need it):



The ParticleImage itsel is just a normal Graphic component (like and Image or a Button):



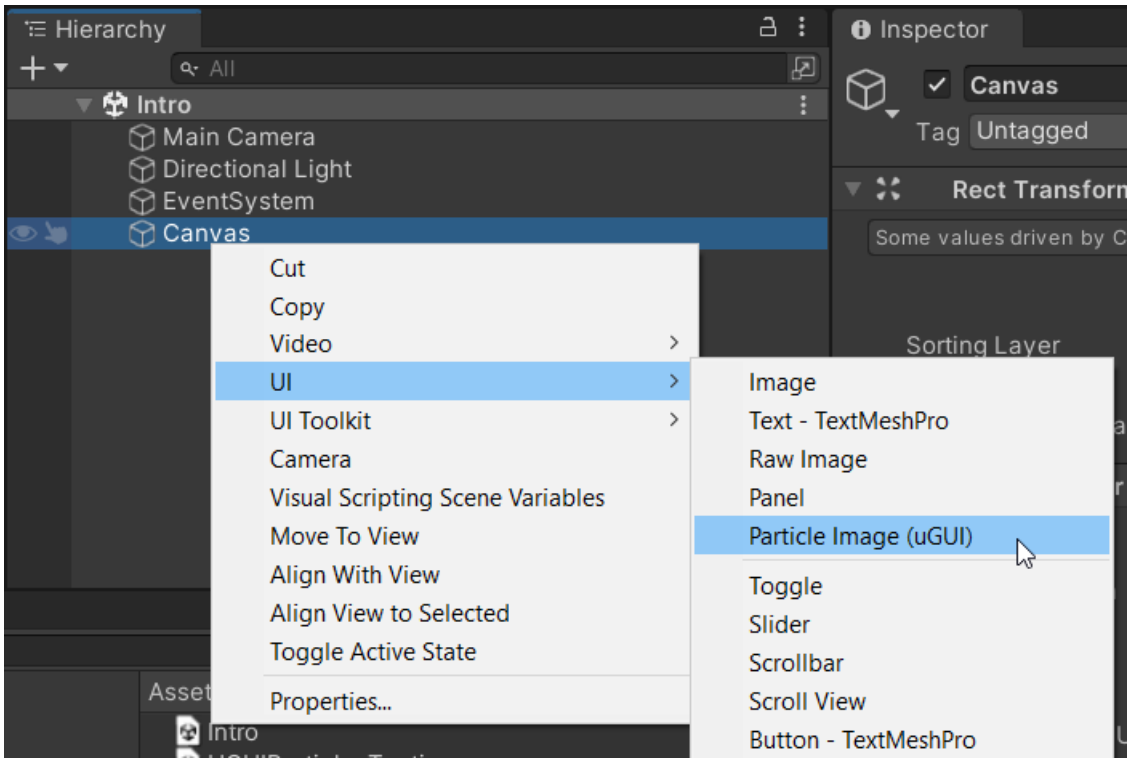
By itself it is rather boring. The interesting stuff if in the „Particle System (for Image)" which you can access via the hierarchy or the „Go to Particle System" button (more on that later):



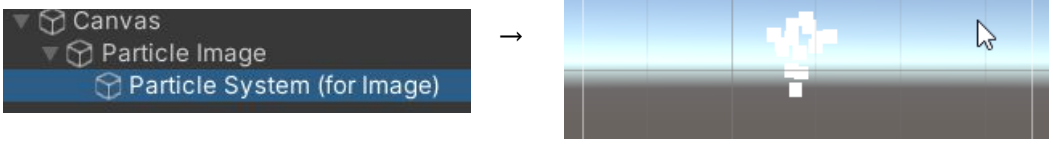
Tutorial: How to create a particle image

You can add a new particle image simpy by right-clicking on your canvas and select:

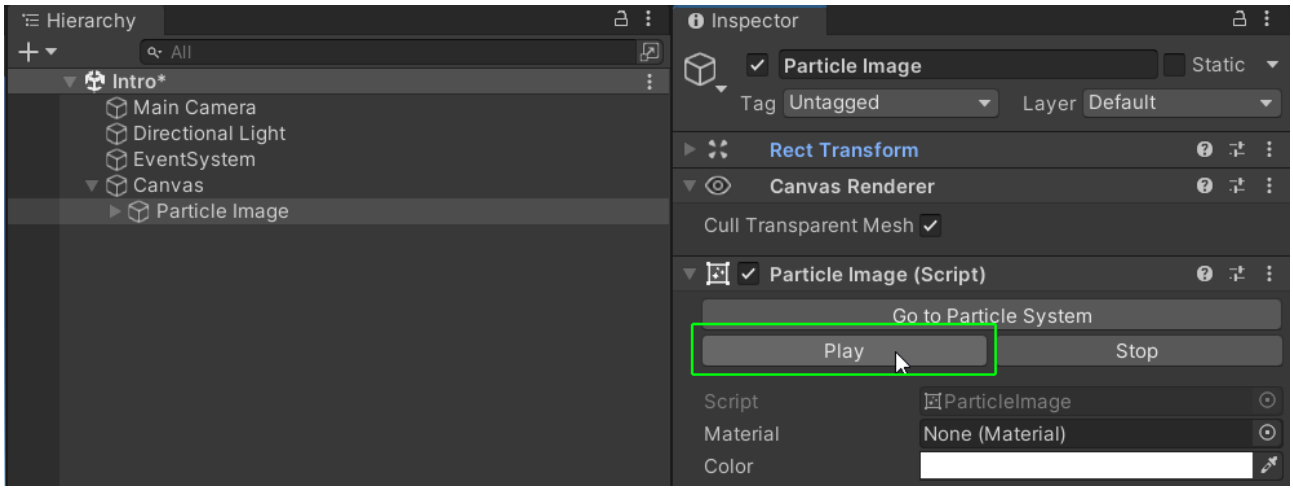
UI → Particle Image (uGUI)



It should automacitally add a Particle Image to your canvas and select the „Particle System (for Image)“ inside it for playback.

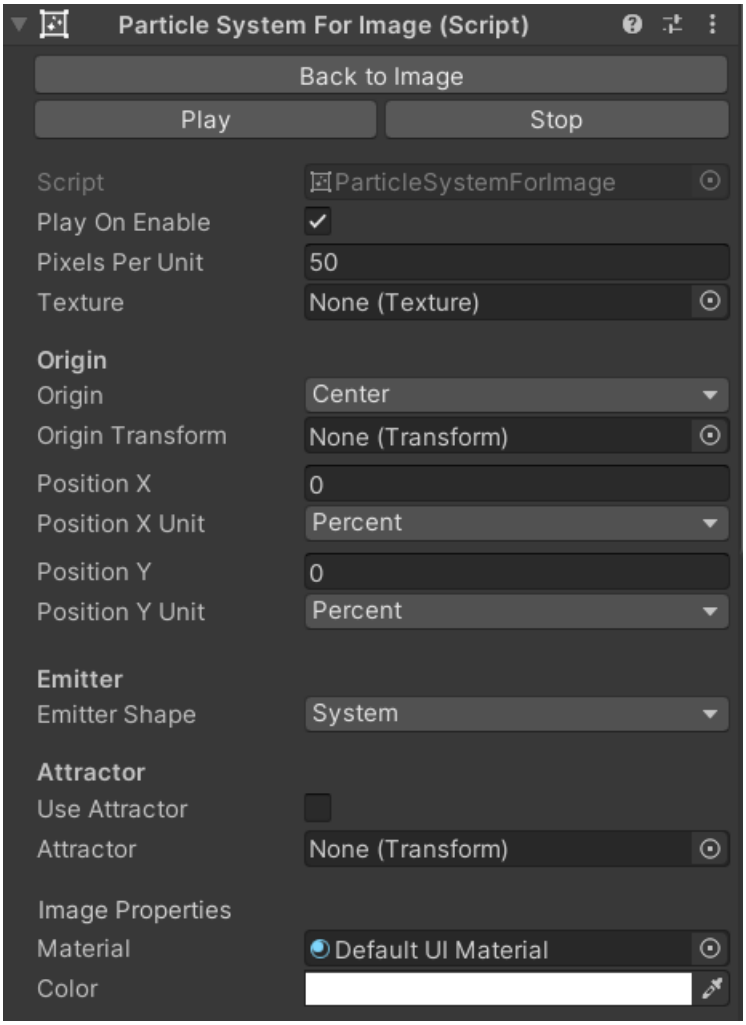


HINT: If it does not start playing automatically then you can start it manually via the „Play“ button on the particle image:



Particle System (for Image)

The „Particle System (for Image)“ is where you can configure all the particle system attributes.



Play On Enable

This one is similar to the „Play On Awake“ you know from the normal particle system. Except it reacts to OnEnable instead of Awake. If enabled then the particles will start to play every time the particle game object is enabled.

Pixels per Unit

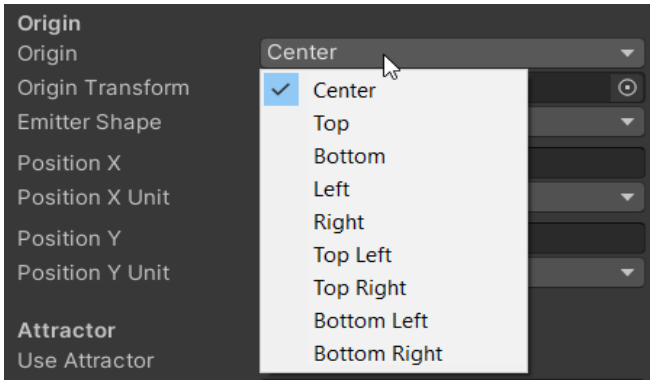
Since the actual particle system runs in the world space and not UI space we need to convert the size from the 3D world space to the UI space. It is a simple scale factor. By default one 3D world unit will become 50 pixels.

Texture

The texture used for each particle. You can only use one texture per particle system. It’s a limitation of the current implementation. Usually adding multiple particle images is a nice solution for cases where multiple images are needed.

Origin

The origin refers to where the particles will spawn in relation to the ParticleImage rectangle. There are some presets to the most commonly used anchor points:



Center for example will make the particles spawn in the center of the image. NOTICE: These are relative to the RECT of the ParticleImage. The pivot is ignored

Origin Transform

Origin Transform can be used as an alternative origin for particles. Here you can drop in a UI Element (RectTransform) or a regular Transfrom from world space.

HINT: With this you can easily make particles emmit from a world space object in the ui. Very useful for tutorials ;-)

Position X/Y

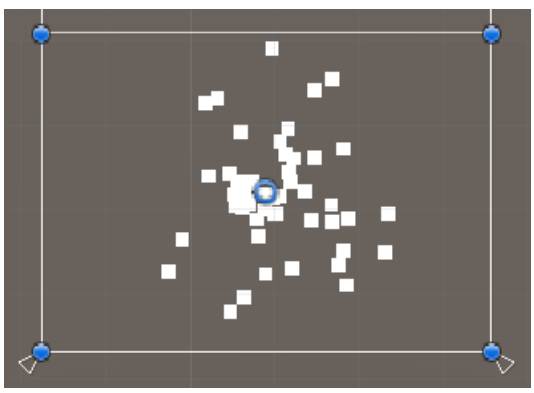
Position X / Position X Unit can be used to offset the particle spawn origin either by pixels or by a percentage of the particle image rect transform size.

HINT: Check out the progress bar demo. It uses PositionX to move the particle spawn point along with the progress bar.

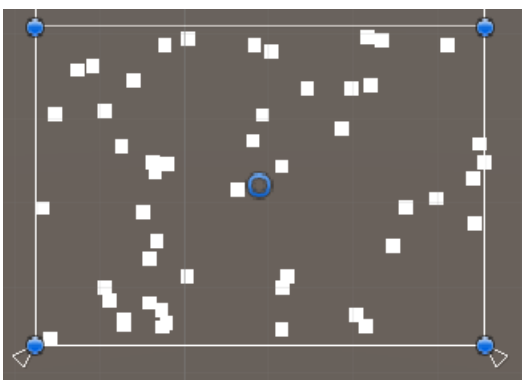
Position Y / Position Y Unit can be used to offset the particle spawn origin either by pixels or by a percentage of the particle image rect transform size.

Emitter Shape Whether the shape of the particle emitter should be based on the ParticleImage rect transform or the particle system shape module.

Emitter Shape „System“:



Emitter Shape „Box Fill“



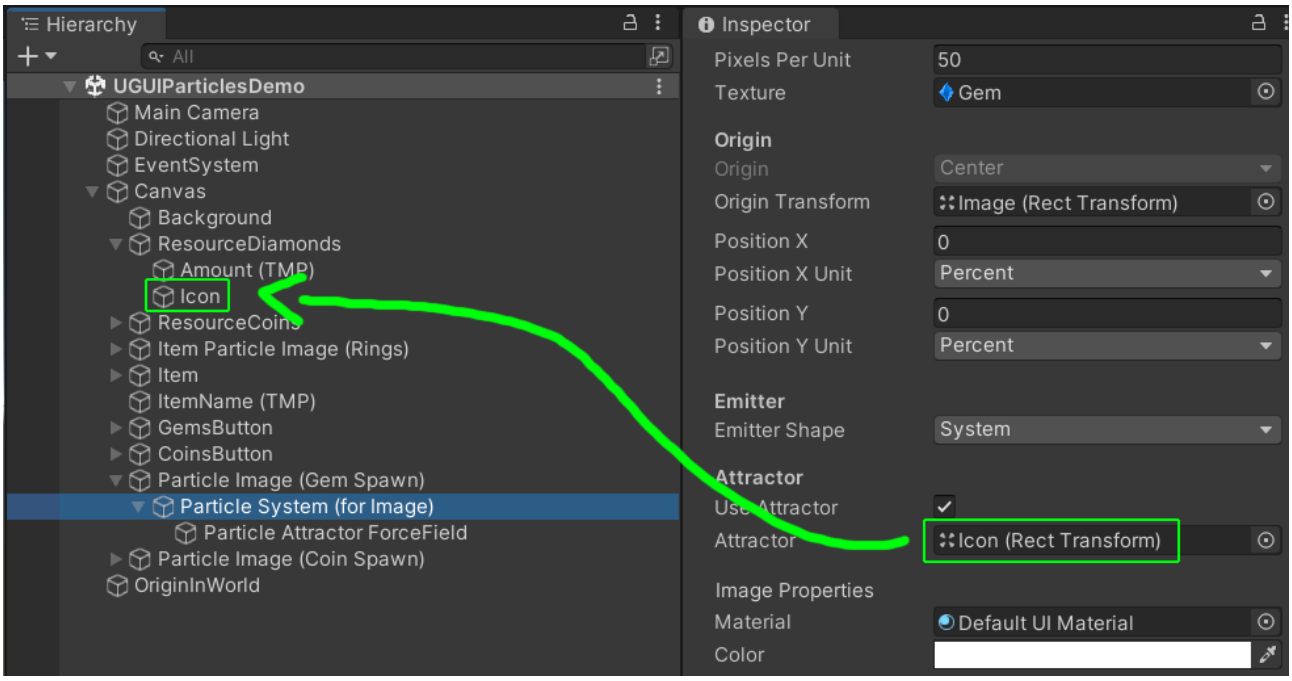
Attractor

Use Attractor can be used to enable/disable the attractor.

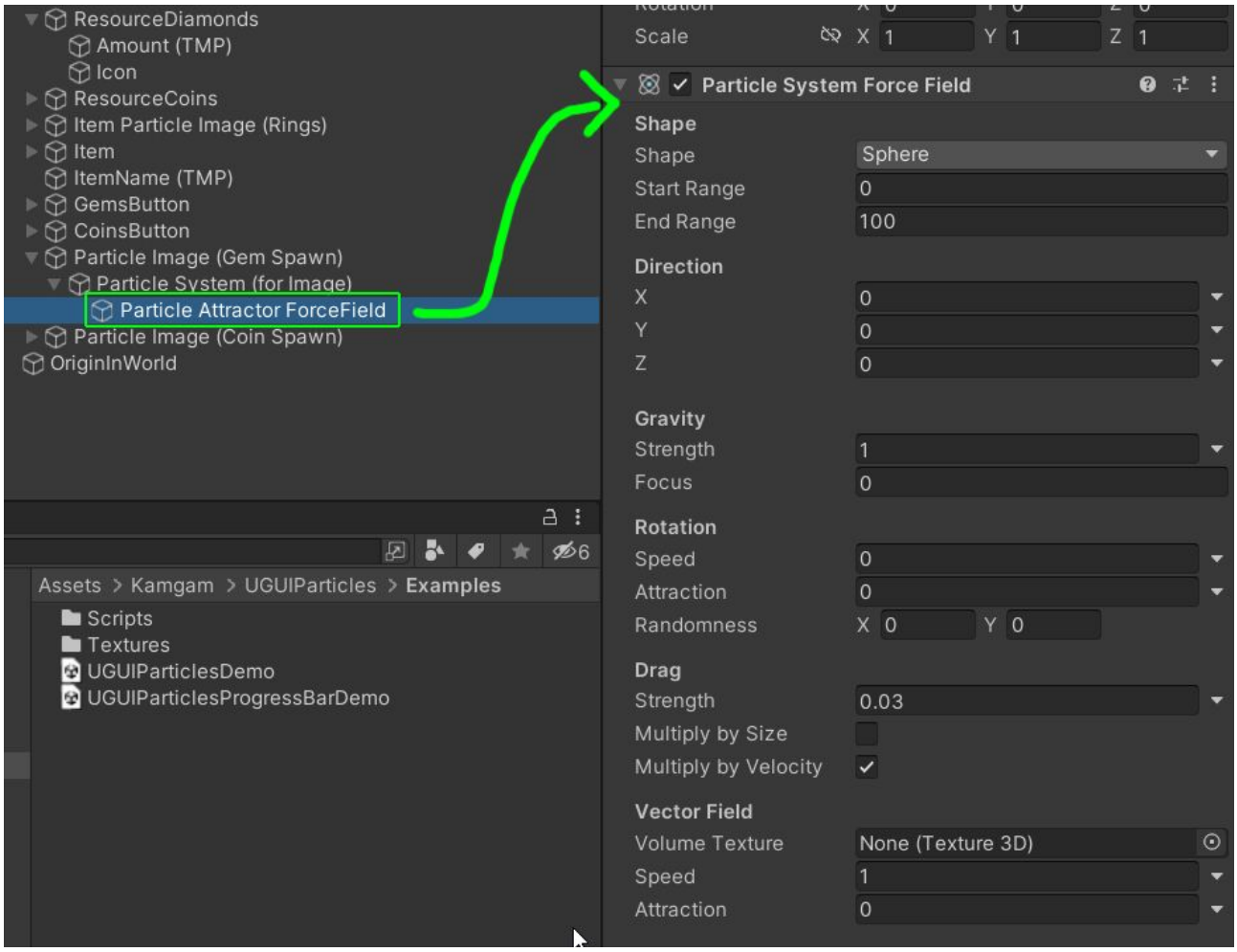
Attractor Attractors are useful to make particles go to a certain position or follow an object.

HINT: The gems and coins spawning from the buttons in the demo are using attractors.

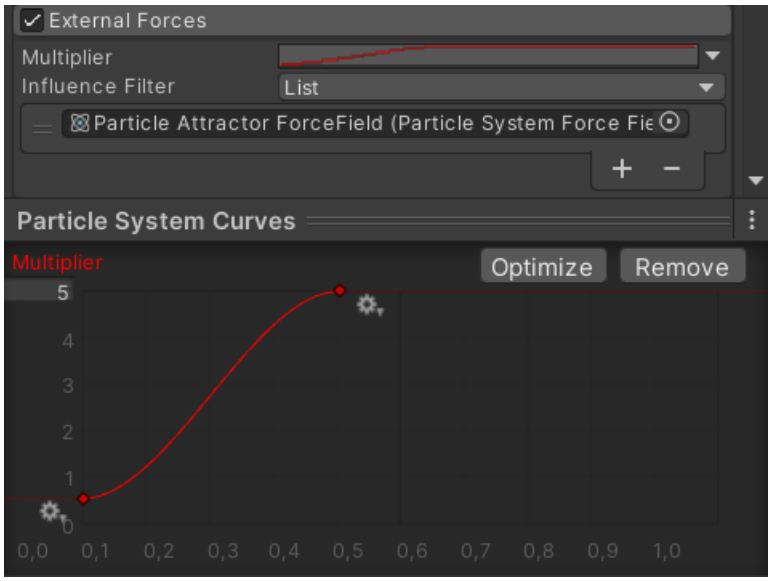
← There is an attractor at the top that makes the gems move towards it.



To edit the attractor itself you have to select it in the hierarchy:



HINT: You can also influence the attactor via the „External Forces“ module in the particle system:



Color

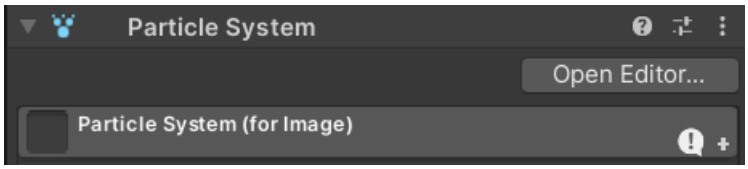
You can use this to change the color tint of your particles, even after you have set a color in the particle system. It supports opacity too. You will notice that this simply is a shortcut to the color property of the Particle Image component.

Material

You can use this to change the used material of the particles. NOTICE: The material should be uGUI compatible. You will notice that this simply is a shortcut to the color property of the Particle Image component.

Particle System

The particles are controlled by three objects the standard Unity ParticleSystem component.



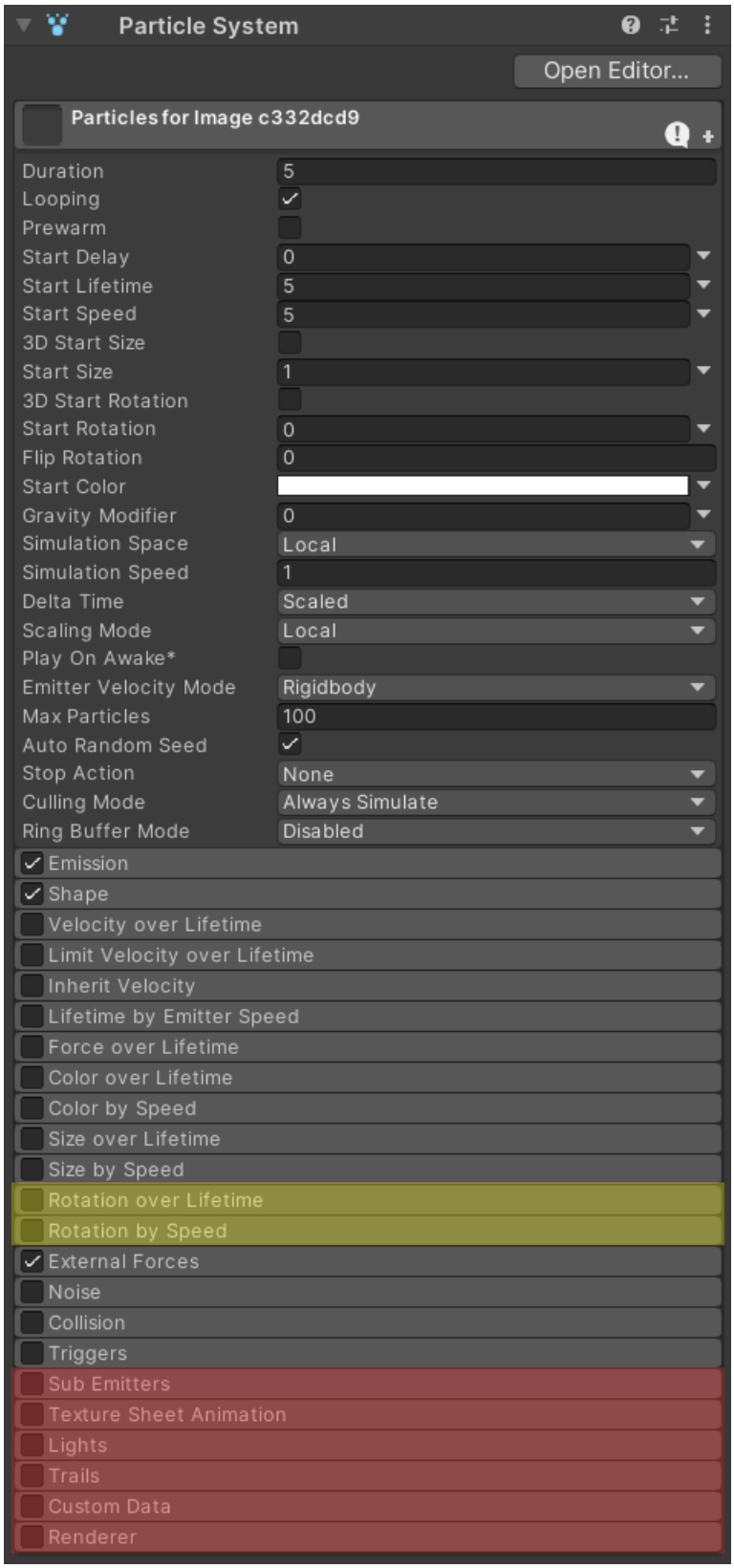
Unsupported modules

The particle system is the Unity default component. However, not all its features are supported. The most notable limitations are:

- **Z-axis rotation only**: Only rotation around the z-axis is supported. The reason is that UI may get clipped or intersect other UI if rotated on the x or y axis.
- **Texture Sheet Animation**: If you need multiple textures please use multiple particle images.
- **Custom Materials**: This uses the default UI shader which is unlit.
- **Sub Emitters**: These are not supported at the moment.
- **Trails**: These are not supported at the moment.
- **Custom Data**: The ParticleSystemForImage does not have an API to access custom particle data. Yet, if needed you can code this yourself. The particle system is fully accessible via ParticleImage.ParticleSystem.
- **Renderer**: The particle renderer is not used at all and thus none of it is supported

To give an overview the modules that are definitely not working are marked red in the image below. Those working partially are marked yellow.

That does not mean all the other modules and all their options do work. For example the „Simulation Space" setting supports „Local" and „World" but not „Custom". The best way to find out what's working is to just try it out.


Particle System
Open Editor...
Particles for Image c332dcd9
Duration 5
Looping
Prewarm
Start Delay 0
Start Lifetime 5
Start Speed 5
3D Start Size
Start Size 1
3D Start Rotation
Start Rotation 0
Flip Rotation 0
Start Color
Gravity Modifier 0
Simulation Space Local
Simulation Speed 1
Delta Time Scaled
Scaling Mode Local
Play On Awake*
Emitter Velocity Mode Rigidbody
Max Particles 100
Auto Random Seed
Stop Action None
Culling Mode Always Simulate
Ring Buffer Mode Disabled
Emission
Shape
Velocity over Lifetime
Limit Velocity over Lifetime
Inherit Velocity
Lifetime by Emitter Speed
Force over Lifetime
Color over Lifetime
Color by Speed
Size over Lifetime
Size by Speed
Rotation over Lifetime
Rotation by Speed
External Forces
Noise
Collision
Triggers
Sub Emitters
Texture Sheet Animation
Lights
Trails
Custom Data
Renderer

Tips & Tricks

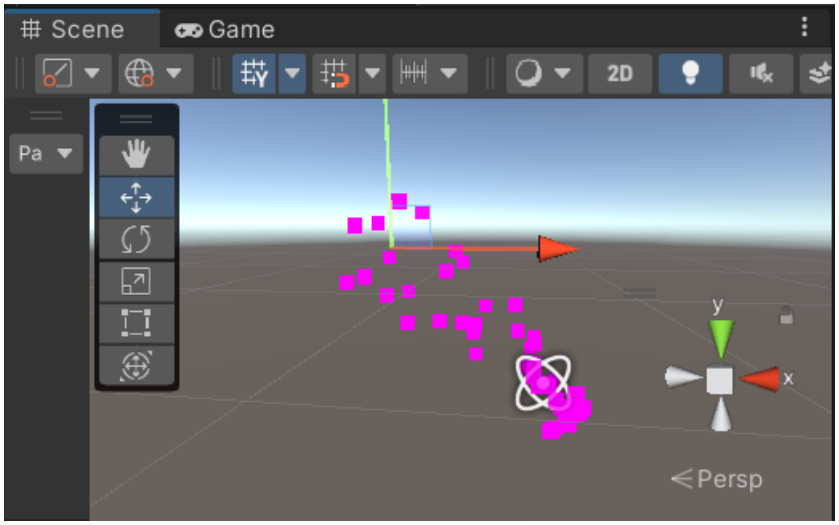
Debugging the particle system

It sometimes helps to see what the particle system is doing. Most of the emitter shapes for example are 3D, not 2D. Observing what they actually do can help understand the behaviour of your particles.

By default the „renderer" module of the particle systems is disabled. Yet for debugging you can turn it on.

If you then double click your particle system in the scene it should take you there.

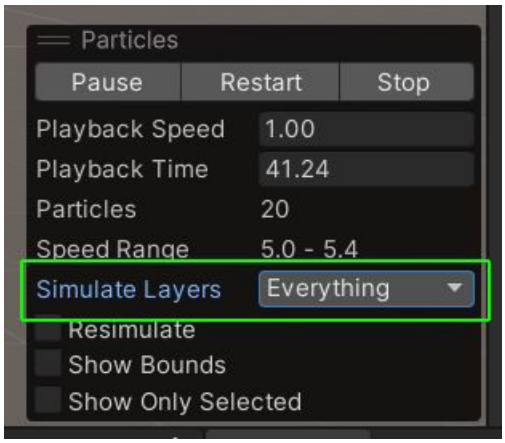
All the particles will be pink because not material is assigned, but you can still see their movement. Especially in combination with forcefields this is a handy trick.



Frequently Asked Questions

The particle system stops playing in the scene view if not selected. How can I make it play always?

Choose „Everything" from the „Simulate Layers" drop-down in the Particles window that pops up in the scene view after selecting a particle system. This will keep the particles from stopping.



HINT: Don't forget to reset the layers or else the particles will re-start playing whenevery you select a new game object.

There is some memory garbage created in the first couple frames.

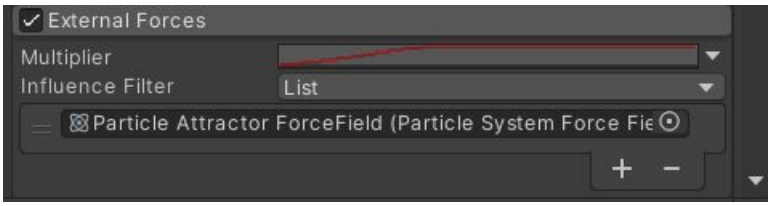
This is due to the Unity internal allocation of the geometry for the particles. Sadly this can not be avoided, but after about 5 frames the internal pools are warmed up and then every new frame has zero GC allocations.

Increasing the max particles property of the system will also cause some new allocations as there is more memory required if the number of max particles increases.

I am having a hard time getting the attractor to look the way I want.

Attractors are a bit tricky to use. Go check out the „UGUIParticlesDemo" scene. It uses some attractors for the gems and coins. Maybe you can copy some properties from there.

One thing I find particularly useful is to use the „Multiplier" in the „External Forces" module of the particles system. It will allow you to fade-in the attractor forces.



World Space particles move in an odd way?

If you use ScreenSpaceOverlay canvases and have set the simulation space to WORLD then sadly the underlying Unity particle system spawns particles in an odd position (depending on the movement speed of the transform). Sadly I do not (yet) know why this is happening. The only thing that is known for sure is that it is caused by the particle system being a child of the canvas RectTransform.

The fix for this is moving the particle system out of this transform hierarchy at RUNTIME. You can do this via the „ParticleSystemForImage.MoveToRoot()" method. This will move the particle system to the root which fixes the issue. HOWEVER: Please notice that by doing this you will be responsible for cleaning up particle systems at runtime as they are no longer part of the particle image hierarchy (just a headsup if you destroy the image).

Here is a code sample:

```
// Fix particles
var particleImage =
yourObject.GetComponentInChildren<Kamgam.UGUIParticles.ParticleImage>(includeIna
ctive: true);
if (particleImage && particleImage.ParticleSystemForImage)
{
    particleImage.ParticleSystemForImage.MoveToRoot();
}
```

HINT: Another common cause for such problems is using World simulation space on a moving particle image (which is a common use case) but also setting the OriginTransform.

If the image is moved in world space settings the origin transform will be counter productive as the position will already match the image automatically. Setting it will „confuse" the system. Make sure (if you use world space and you move your particle image) that the origin is cleared, like this: