Simple UGUI Particles

Table of contents

Requirements & Setup **2**
Requirements 2
Introduction **3**
Tutorial: How to create a particle image **4**
Particle System (for Image) **5**
Play On Enable 5
Pixels per Unit 5
Texture 5
Origin 6
Origin Transform 6
Position X/Y 6
Attractor 7
Color 10
Material 10
Particle System **13**
Unsupported modules 13
How to modify source and target positions **15**
How to change the source (origin) position of particles 15
How to change the target position of particles by using an attractor 19

Tips & Tricks.....24
Debugging the particle system.....24
Previewing the particle system while changing the size.....24
Keep playing the particle system while changing the size.....25
Frequently Asked Questions.....26
The particle system stops playing in the scene view if not selected. How can I make it play always?.....26
There is some memory garbage created in the first couple frames.....26
I am having a hard time getting the attractor to look the way I want.....26
World Space particles move in an odd way?.....27

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

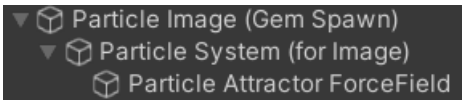
This section will help you understand what is going on in the hierarchy.

The particles are controlled by three objects:

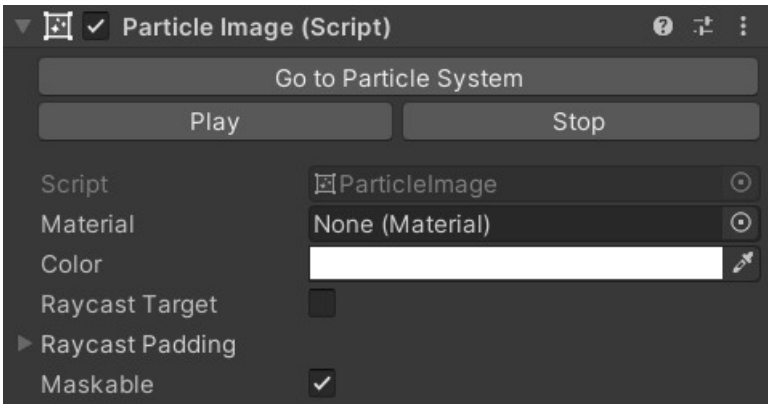
- The ParticleImage custom control in the UI
- The ParticleSystemForImage component in the scene
    - The ParticleSystem in the scene (tightly coupled with the ParticleSystemForImage)

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.

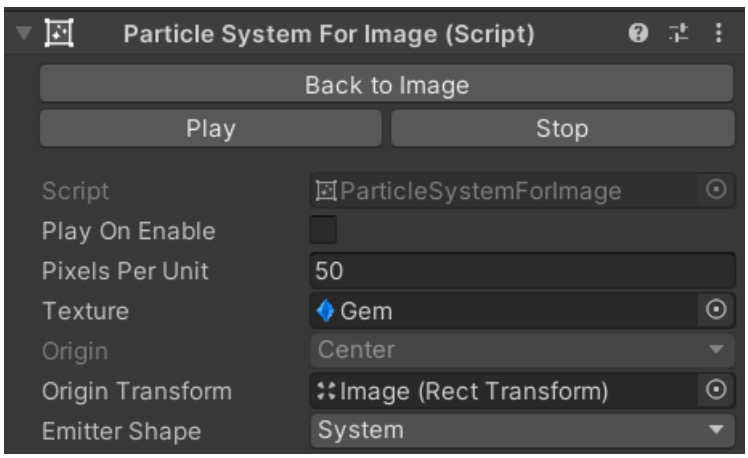
In the hierarchy a particle image looks like this (the attractor is optional and only added if you need it):



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



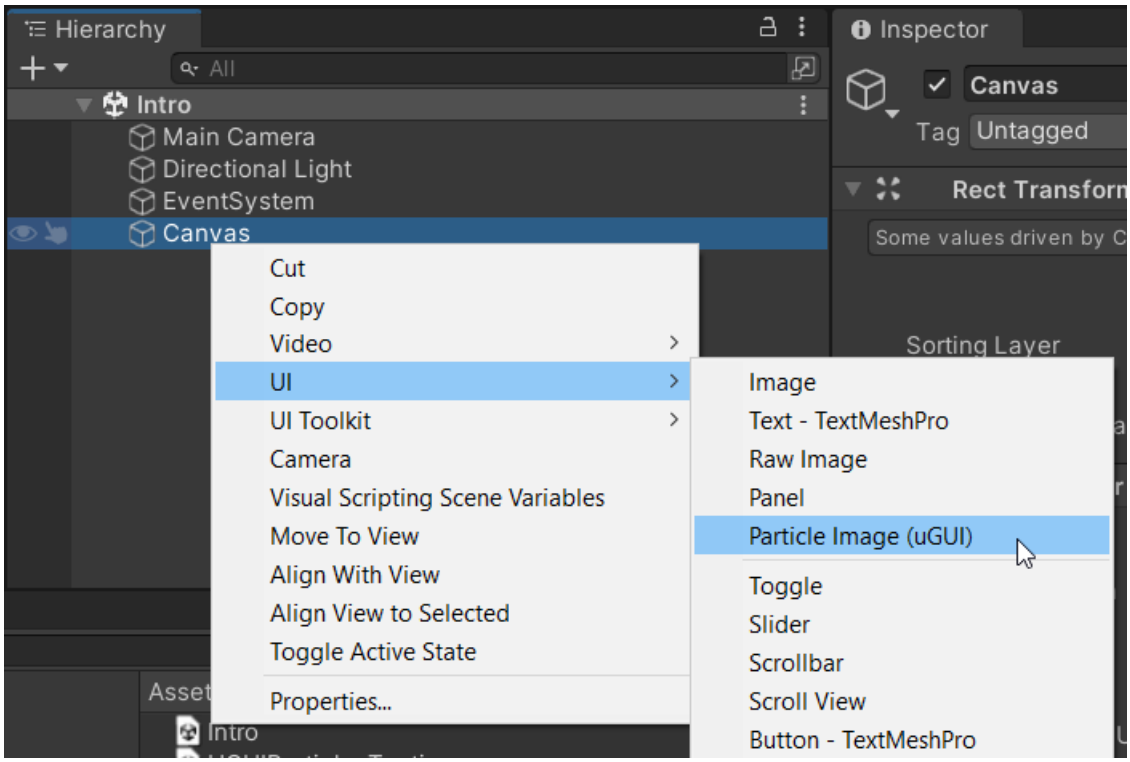
By itself it is rather boring. The interesting stuff if in the „Particle System (for Image)" which you can access via the hierarchy or the „Go to Particle System" button (more on that later):



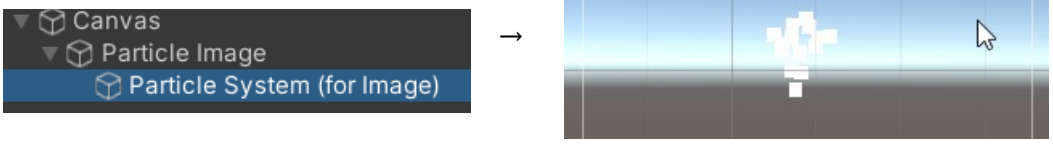
Tutorial: How to create a particle image

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

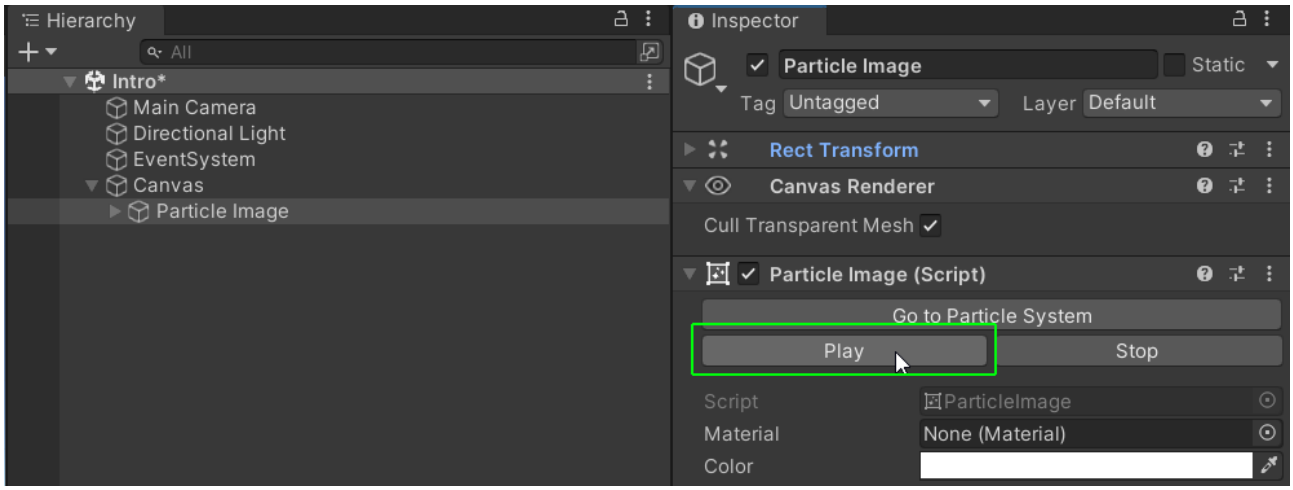
UI → Particle Image (uGUI)



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

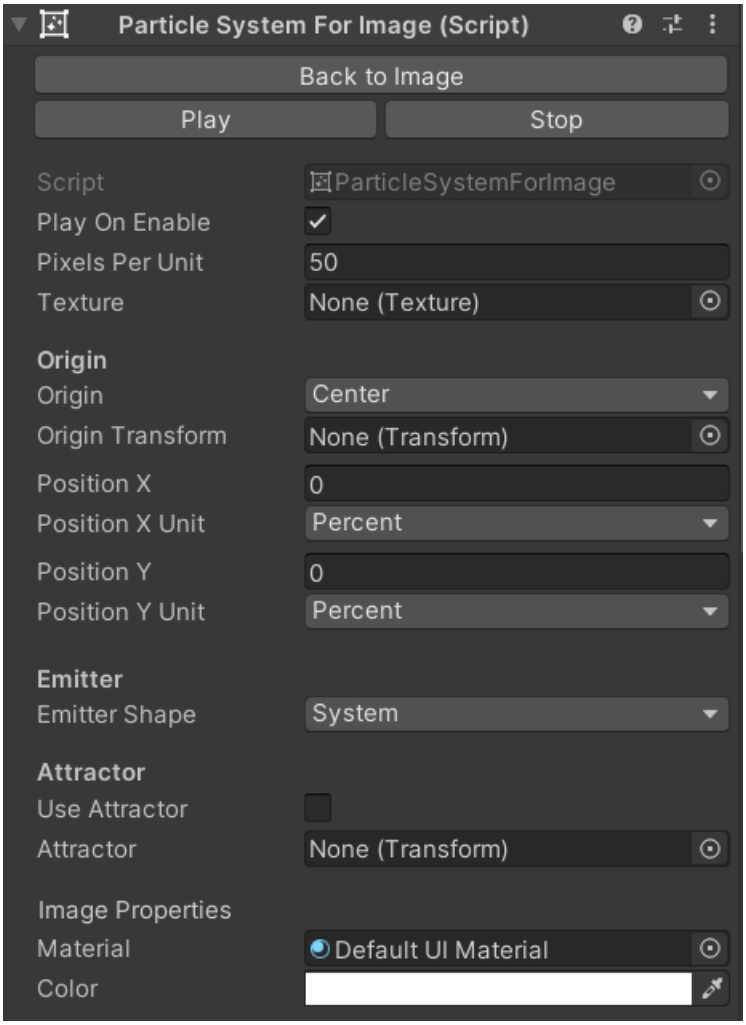


HINT: If it does not start playing automatically then you can start it manually via the „Play“ button on the particle image:



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

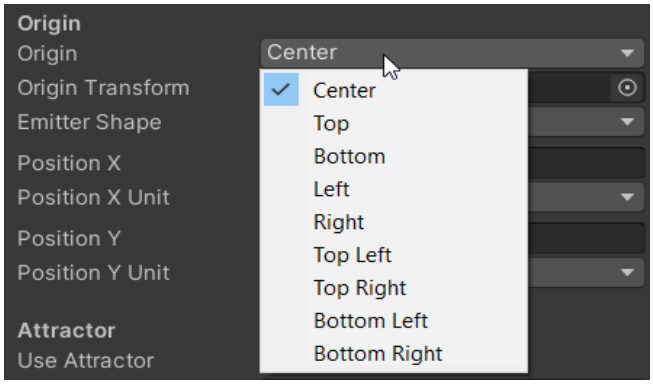
Since the actual particle system runs in the world space and not UI space we need to convert the size from the 3D world space to the UI space. It is a simple scale factor. By default one 3D world unit will become 50 pixels.

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

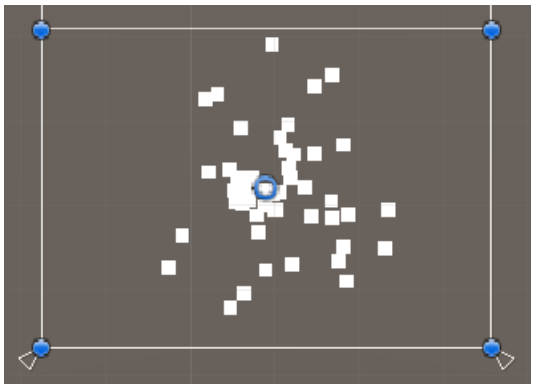
Position X / Position X Unit can be used to offset the particle spawn origin either by pixels or by a percentage of the particle image rect transform size.

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

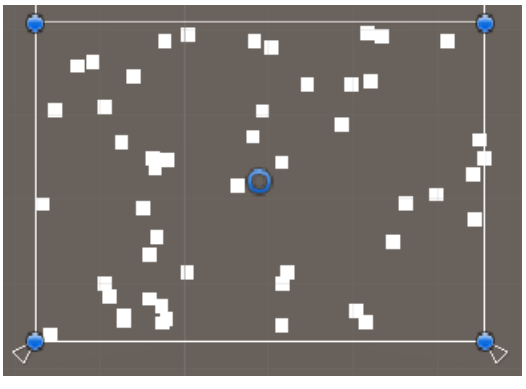
Position Y / Position Y Unit can be used to offset the particle spawn origin either by pixels or by a percentage of the particle image rect transform size.

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“



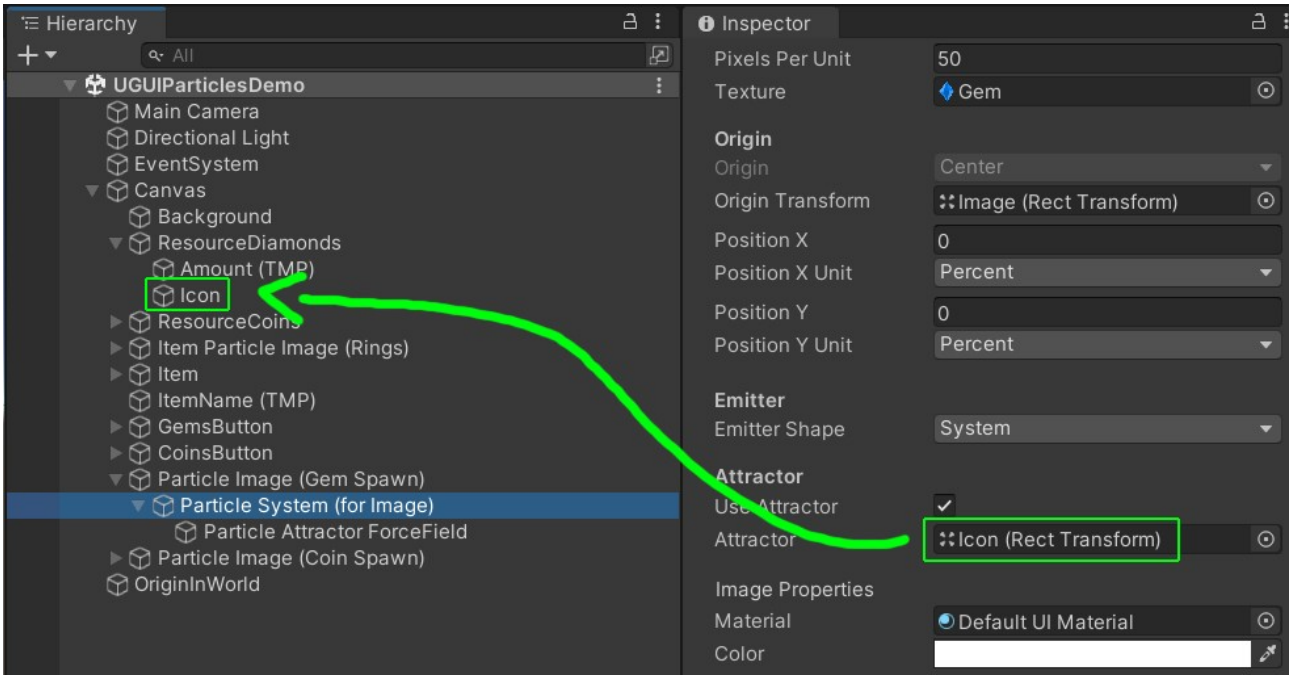
Attractor

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

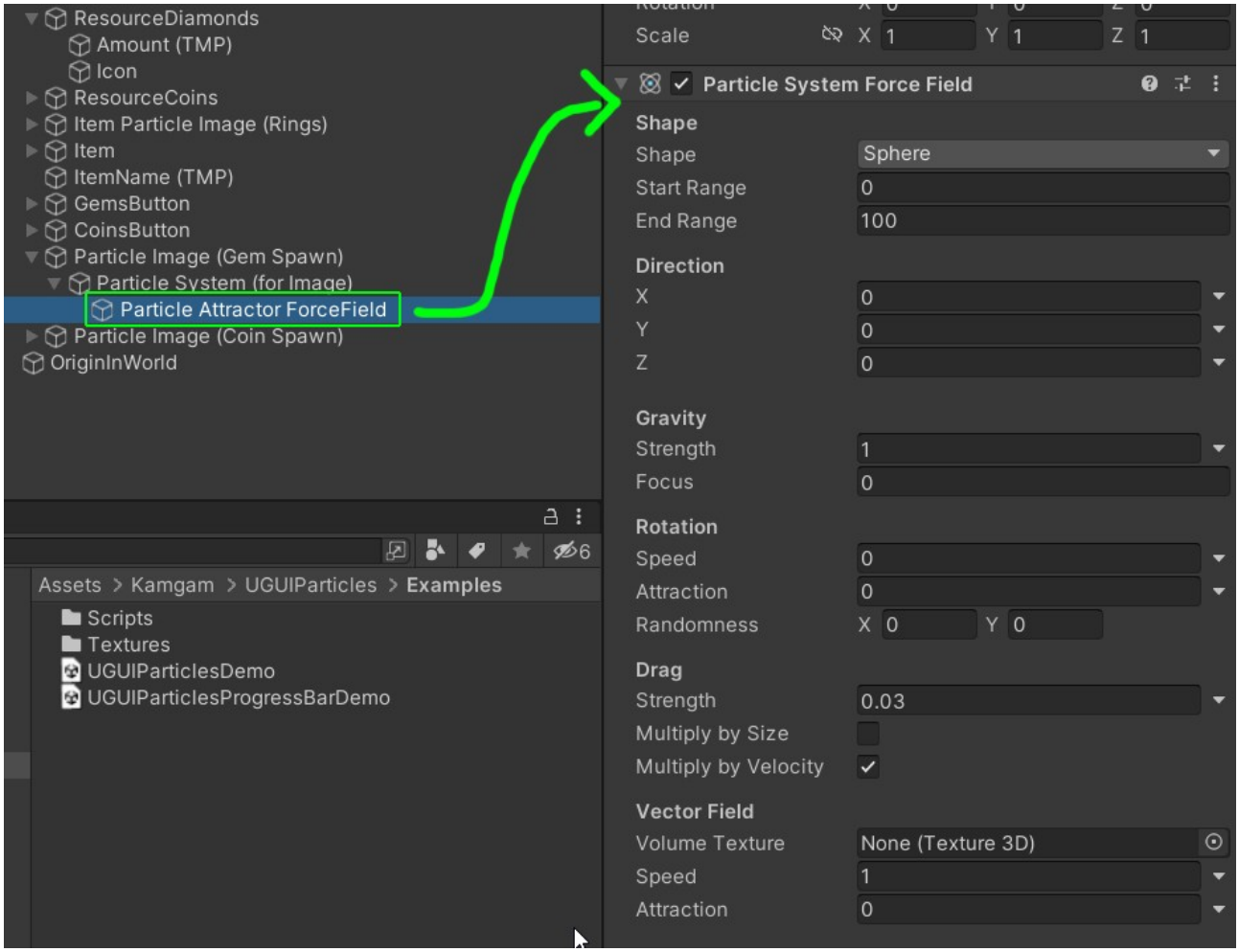
Attractor Attractors are useful to make particles go to a certain position or follow an object.

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

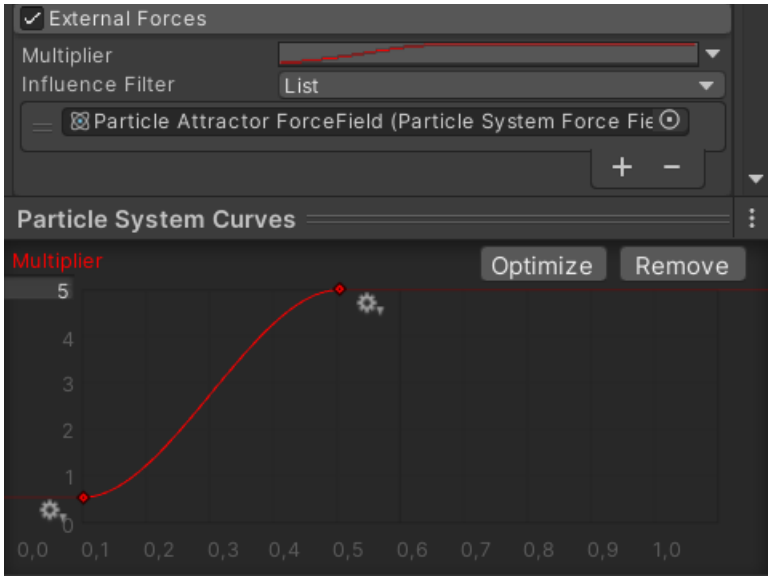
← There is an attractor at the top that makes the gems move towards it.


Hierarchy
All
UGUIParticlesDemo
Main Camera
Directional Light
EventSystem
Canvas
Background
ResourceDiamonds
Amount (TMP)
Icon
ResourceCoins
Item Particle Image (Rings)
Item
ItemName (TMP)
GemsButton
CoinsButton
Particle Image (Gem Spawn)
Particle System (for Image)
Particle Attractor ForceField
Particle Image (Coin Spawn)
OriginInWorld
Inspector
Pixels Per Unit
50
Texture
Gem
Origin
Origin
Center
Origin Transform
Image (Rect Transform)
Position X
0
Position X Unit
Percent
Position Y
0
Position Y Unit
Percent
Emitter
Emitter Shape
System
Attractor
Use Attractor
Attractor
Icon (Rect Transform)
Image Properties
Material
Default UI Material
Color

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

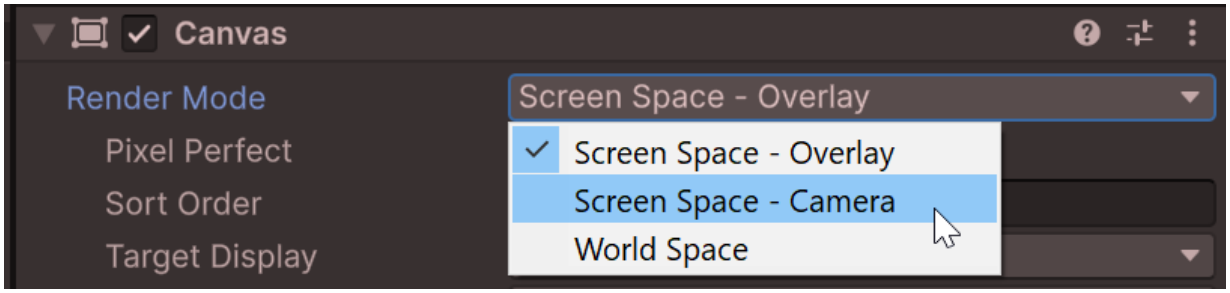
You can use this to change the color tint of your particles, even after you have set a color in the particle system. It supports opacity too. You will notice that this simply is a shortcut to the color property of the Particle Image component.

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.

INFO / CAVEAT: Using custom Materials is always a bit hit or miss, especially if used on ScreenSpaceOverlay canvases.

Solution: Try making the canvas a screen space camera canvas instead. That should give you rendering like in the scene view.



The reason is that the rendering for overlays works differently from the rest (direct write to screen instead of color and depth buffer) and most blend shaders do not work properly there (not even Unity's own shaders). Sadly that is a Unity limitation and not specific to the particles asset (the asset renders the particles just like any other UI image). The reason why it works in the scene view is that in the scene view the UI is always rendered in a camera context (like a ScreenSpaceCamera UI in the game view).

Here is a simple custom alpha blend shader for UI:

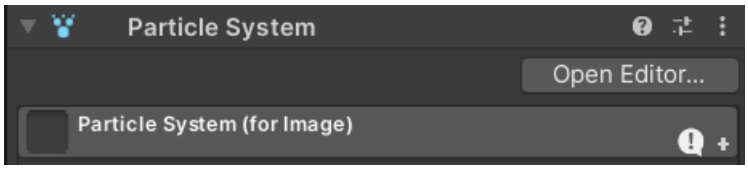
```
Shader "UI/UIAlphaBlendShader"
{
    Properties
    {
        [PerRendererData] _MainTex ("Sprite Texture", 2D) = "white" {}
        _Color ("Tint", Color) = (1,1,1,1)
        _Alpha ("Alpha", Range(0, 1)) = 1
        _BlendMode ("Blend Mode", Int) = 0 // 0: Alpha, 1: Additive, 2: Subtractive, 3:
Multiplicative
    }

    SubShader
    {
        Tags
        {
```

```
        "Queue"="Transparent"
        "IgnoreProjector"="True"
        "RenderType"="Transparent"
        "PreviewType"="Plane"
        "CanUseSpriteAtlas"="True"
    }

    Cull Off
    Lighting Off
    ZWrite Off

    Blend SrcAlpha OneMinusSrcAlpha // Default blending mode

    Pass
    {
        CGPROGRAM
        #pragma vertex vert
        #pragma fragment frag
        #include "UnityCG.cginc"

        struct appdata_t
        {
            float4 vertex   : POSITION;
            float4 color    : COLOR;
            float2 texcoord : TEXCOORD0;
        };

        struct v2f
        {
            float4 vertex   : SV_POSITION;
            fixed4 color    : COLOR;
            half2 texcoord  : TEXCOORD0;
        };

        sampler2D _MainTex;
        float4 _Color;
        float _Alpha;
        int _BlendMode;

        v2f vert(appdata_t IN)
        {
            v2f OUT;
            OUT.vertex = UnityObjectToClipPos(IN.vertex);
            OUT.texcoord = IN.texcoord;
            OUT.color = _Color;
            OUT.color.a *= _Alpha;
            return OUT;
        }

        fixed4 frag(v2f IN) : SV_Target
        {
            fixed4 c = tex2D(_MainTex, IN.texcoord) * IN.color;

            // Apply blending mode
            switch (_BlendMode)
            {
                case 1: // Additive
                    c.rgb += c.a;
                    break;
                case 2: // Subtractive
                    c.rgb = -c.rgb;
                    break;
```

```
                    case 3: // Multiplicative
                        c.rgb *= c.rgb;
                        break;
                    default: // Alpha (default)
                        c.rgb *= c.a;
                        break;
                }

                return c;
            }
            ENDCG
        }
    }
}
```

Particle System

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



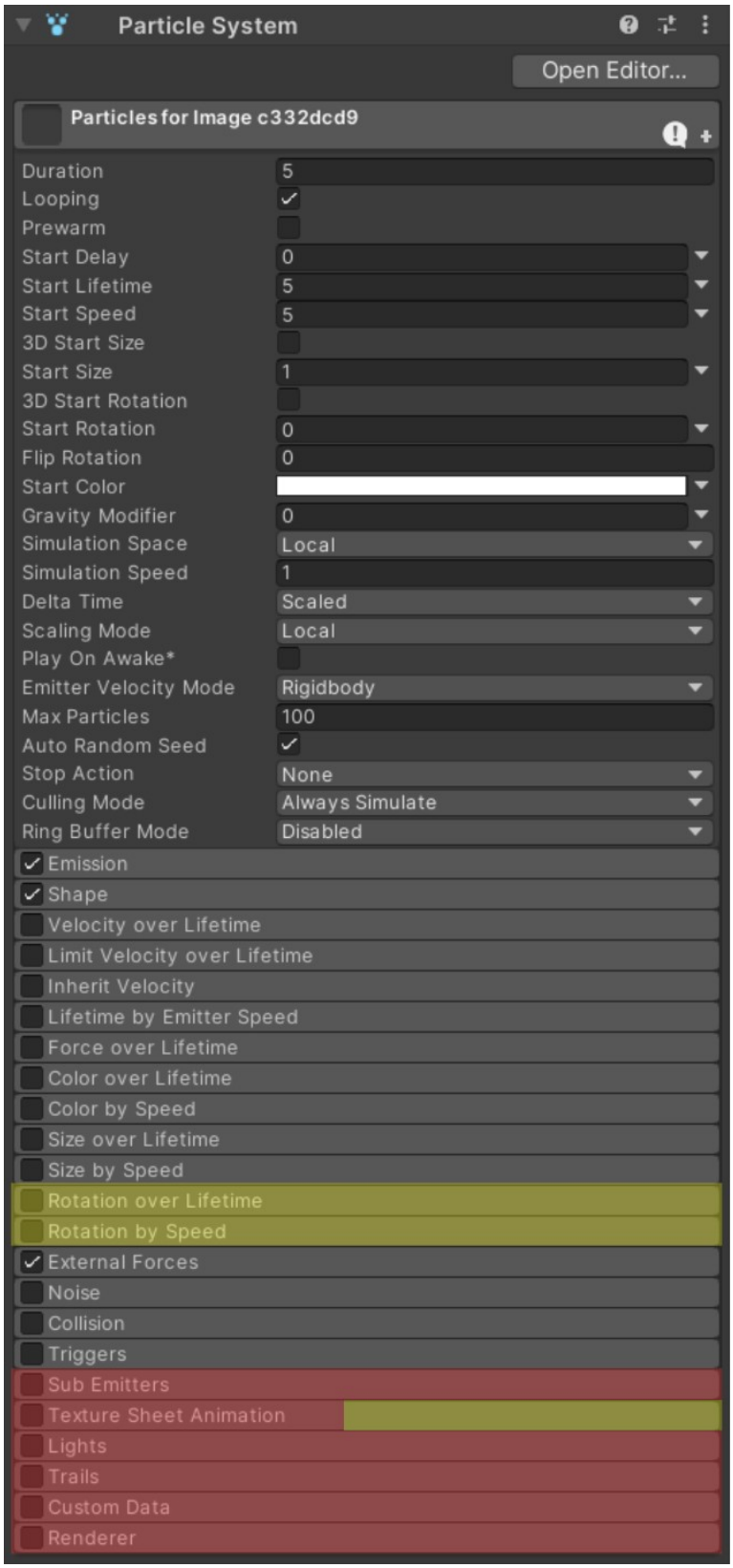
Unsupported modules

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**: Partial Support, lifetime and fps are supported, speed mode is not.
- **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.

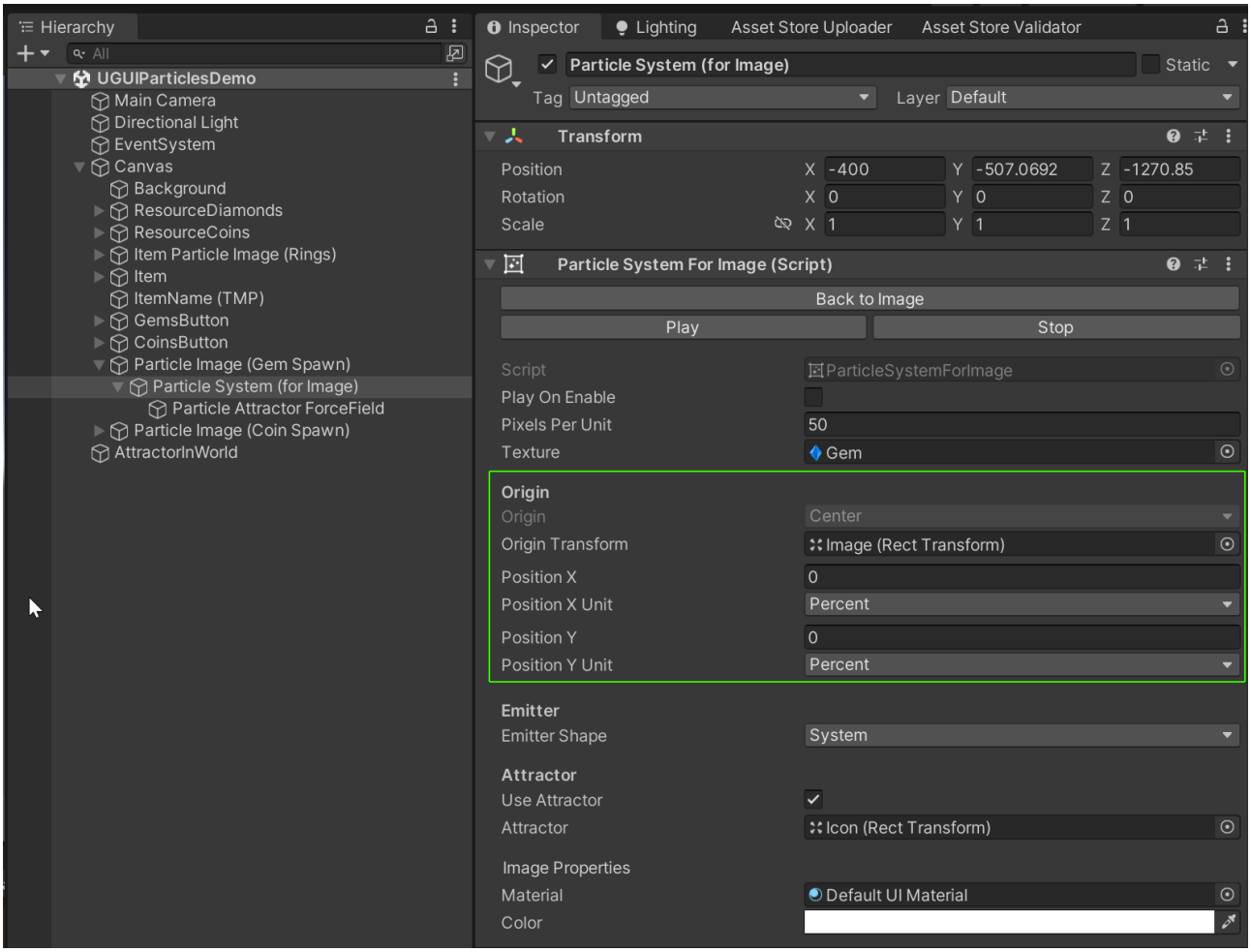
That does not mean all the other modules and all their options do work. For example the „Simulation Space" setting supports „Local" and „World" but not „Custom". The best way to find out what's working is to just try it out.


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

How to modify source and target positions

How to change the source (origin) position of particles

If you take a look into the UGUIParticlesDemo scene then you will find a „Particle Image (Gem Spawn)“ object. In there you can find the „Particle System (for Image)“. If you click it you will find a section titled „Origin“. This is where you can specify the origin position of the particles (please refer to the sections above on the details of each option).

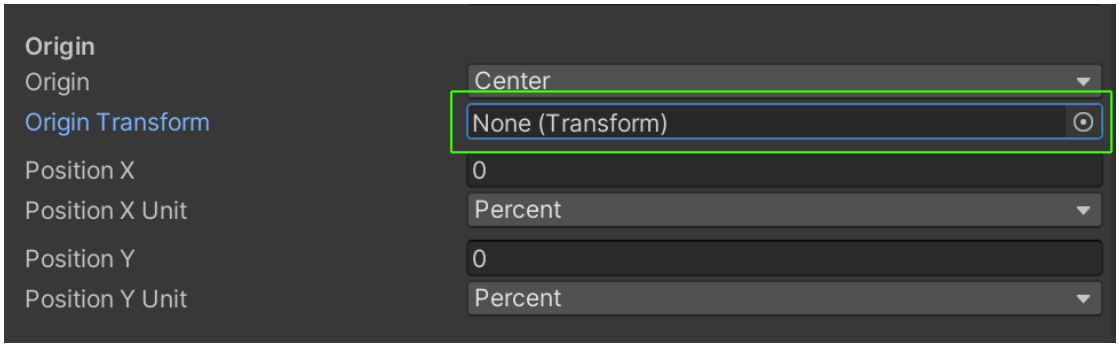


In the example it is set to spawn from the image inside the GemsButton:

If you hit play and click the gems button you will notice the particles emit from the gem inside the button:

Now, let's assume we want to change the spawn position to an arbitrary point instead of the gem image.

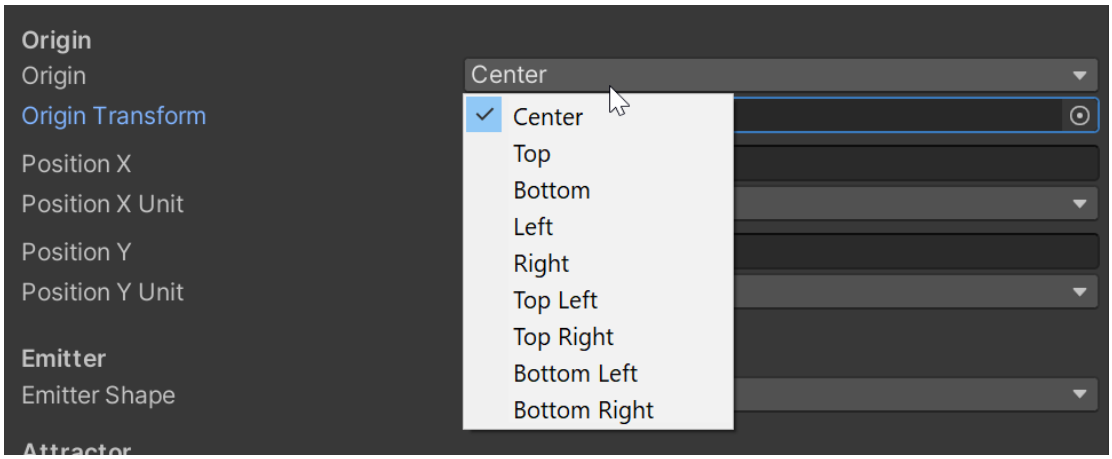
First we have to clear the origin transform because if that is set it will override the starting point from where the Position X and Y are calculated:



Since we no longer have a „Origin Transform" set you may wonder where the starting point will be? It will simply be the origin of the particle image. In the example the particle image covers

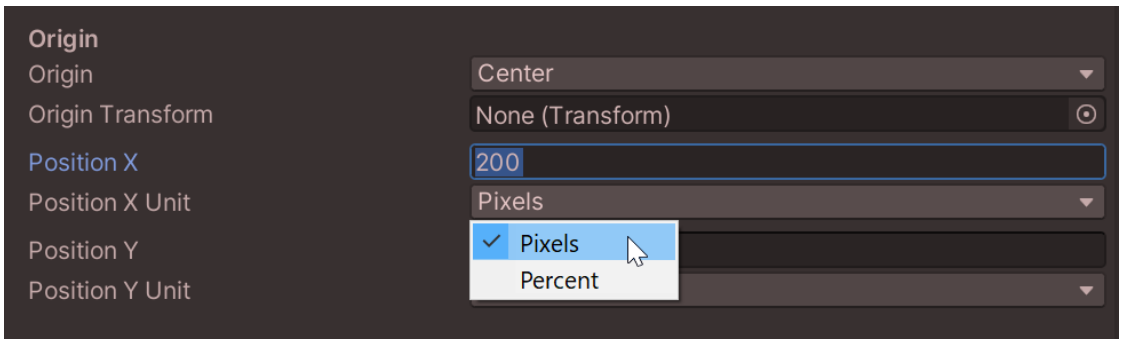
the whole screen (left). If you hit play and click the gems button you will notice the particles spawn from the origin of the particle image.

HINT: You can change the starting point via the „Origin“ Drop Down. It makes it easier to let the particles spawn at the sides or a corner.



HINT: The Position X and Y are added to the Origin Position.

Now let's say you want to manually set the origin point in pixels (where the term „pixels" is referring to the reference pixels of your canvas).



Then if you hit play you will see the particles origin will have shifted 200px to the right relative to the particle image origin:

Phew okay, that's it on how to change the „Origin. In the next section we will also change the Target by using an „Attractor".

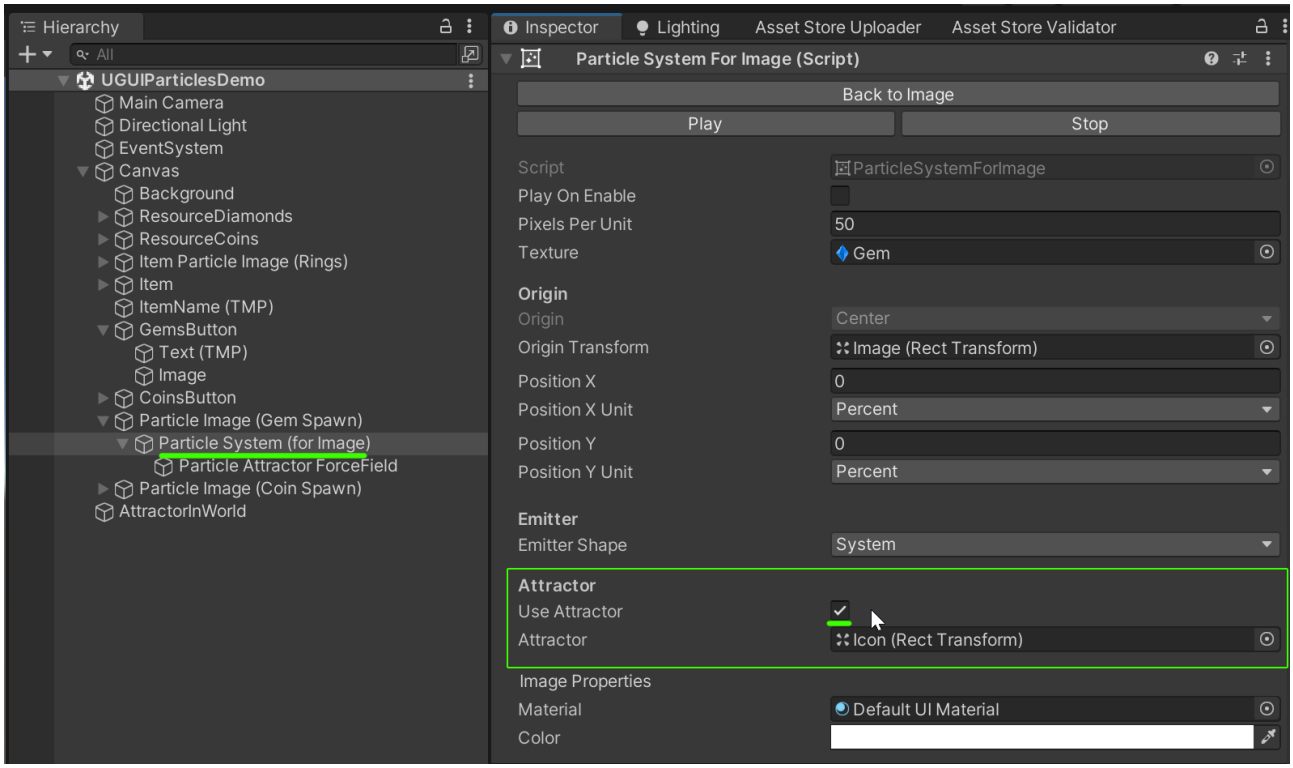
How to change the target position of particles by using an attractor

The first thing to know is that in order to make particles move to a certain location the Unity Particle system uses physics and force fields.

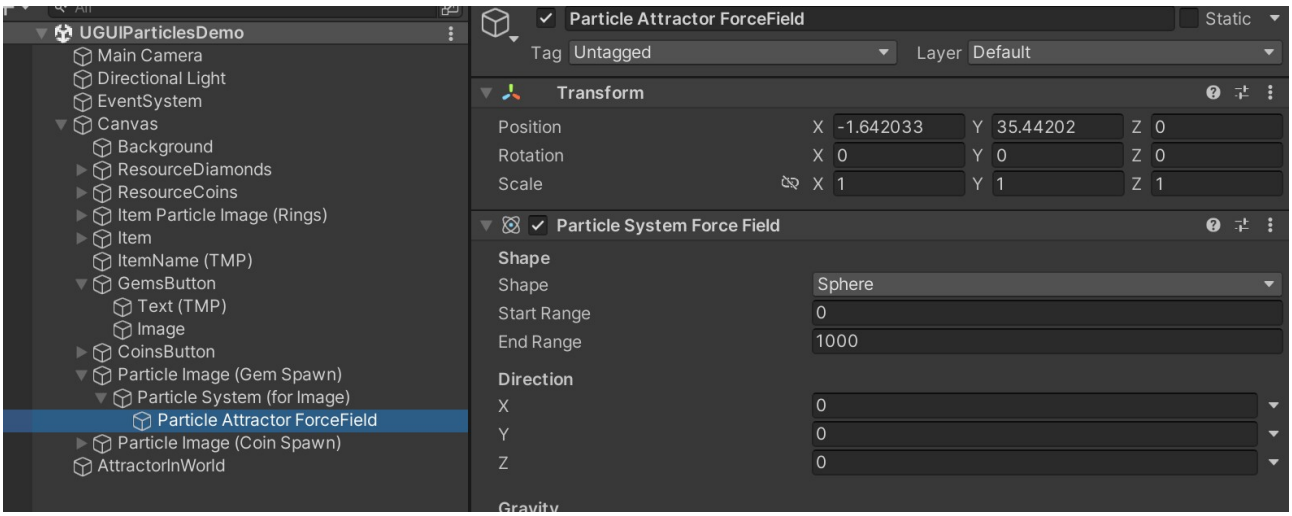
The particle image has an automated system that sets up one spherical force field for you (we call it an „Attractor"). It will suck in all particles that are nearby, giving the effect of particles moving towards it. That's how the gem and coin particles are made in the UGUIParticlesDemo.

Our goal in this section is to understand how it is done in the demo and change the target point of the force field to an arbitrary point relative to the particle image.

The first thing you would have to do is enable the „Use Attractor" checkbox on the Particle System (for Image). In the demo this is already done for you.



This will create (or activate) a ParticleAttractor ForceField in the System:



It really is just a normal particles force field with some predefined settings. As you can see it is a huge sphere that attracts particles to it`s center. Anyways, we will leave it alone for now.

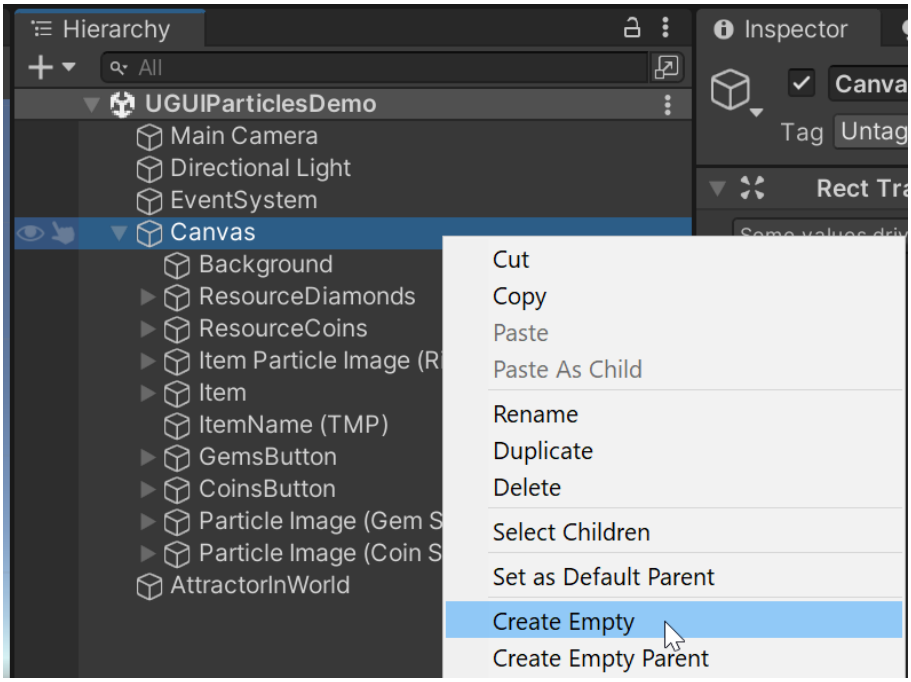
HINT: If you want to change it please notice the „Tipps & Tricks“ section above. It's very hand to see the particles move while fiddling around with the settings. Documentation on the settings themselves can be found in the official Unity documentation.

Back on the „Particle System (for Image)“ you will see that there also is an „Attractor“ slot for a transform. This transform will define the center of the spherical force field (attractor) that sucks in the particles.

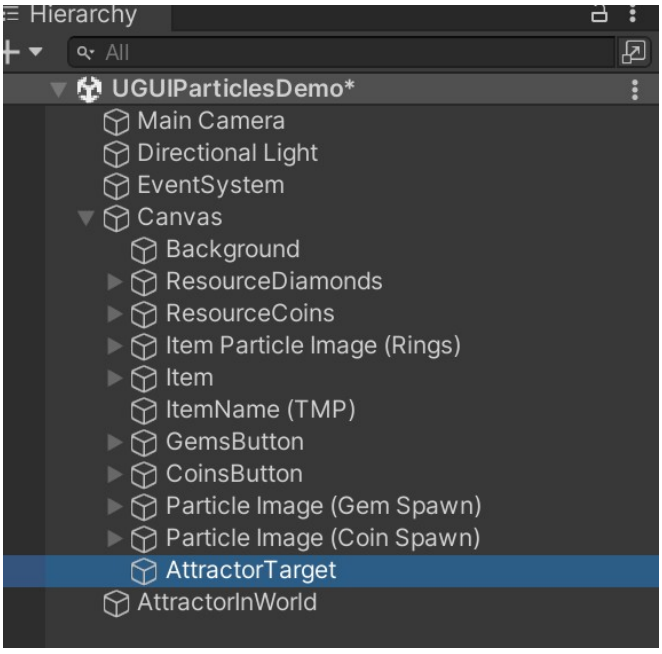
As you can see if you hit play the particles are attracted to the center of the „Icon“ transform inside the „ResourceDiamonds“.

Now let's say you want to change the position of the attraction center. How would we do that?

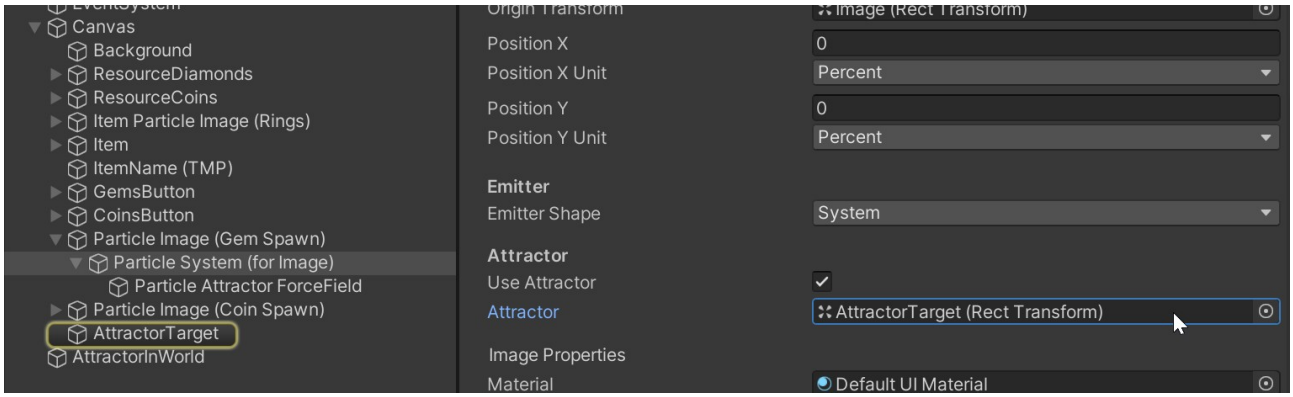
Sadly the Attractor target does not (yet) have inputs for arbitrary coordinates but we can trick it by adding an empty RectTransform, like so:



Let's call it „AttractorTarget":



Let's use this new transform as the new attractor target like this:



If we hit play now the particles will move towards this new transform.

If you want to change the position you can either change it directly in the Canvas or via code.

And that's it. I hope this helps with configuring the particles movement.

HINT: Please notice that the Forcefield settings of the attractor can be tricky to get right. That's just in the nature of the forcefield physics. I tried to make the default as easy to use as possible but sometimes it may take a bit of fiddling around until you get the results you want.

HINT 2: The attractor target can be any transform, not just a RectTransform. You can plug in a transoform of a world object too and the particles will move towards the 2D position of that object. That's how the sphere in the video was done.

HINT 3: Check out the „Debugging the particle system" section below. It is very handy for tweaking force fields.

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.

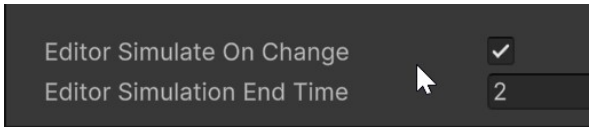
By default the „renderer" module of the particle systems is disabled. Yet for debugging you can turn it on.

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.

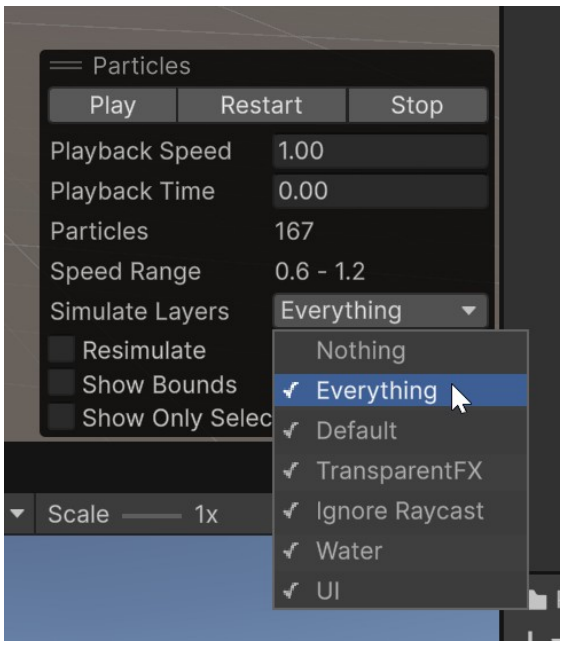
Previewing the particle system while changing the size

If you want to see the particle system state updated while you change the image size you can enable the "Editor Simulate On Change" flag. This will simulate the system for N seconds every time you change the width or height of the image. This is handy to guage how the system will look like in the UI.



Keep playing the particle system while changing the size

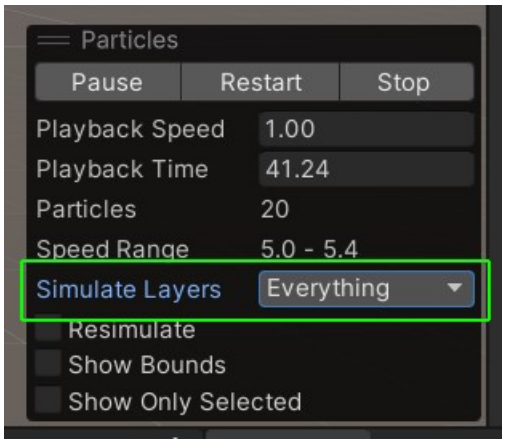
Usually particle system stop playback once their parent game object is no longer selected. To avoid this you can set the simulate layers flag in the scene view particles tool to “Everything”. once enabled the system will always play. This is handy to preview the particles even while editing the image.



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.

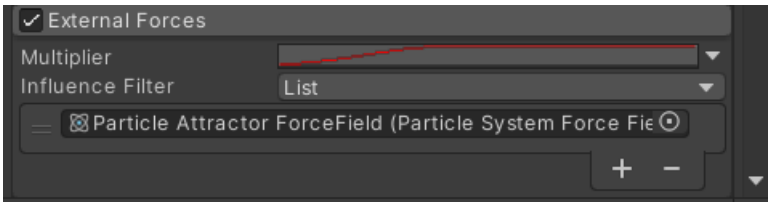
This is due to the Unity internal allocation of the geometry for the particles. Sadly this can not be avoided, but after about 5 frames the internal pools are warmed up and then every new frame has zero GC allocations.

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.



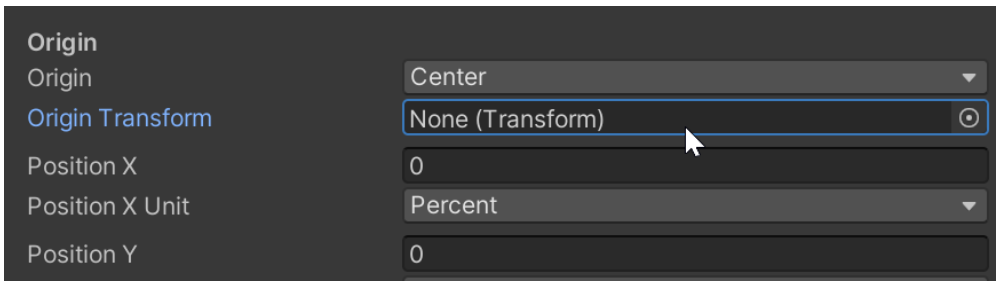
World Space particles move in an odd way?

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: