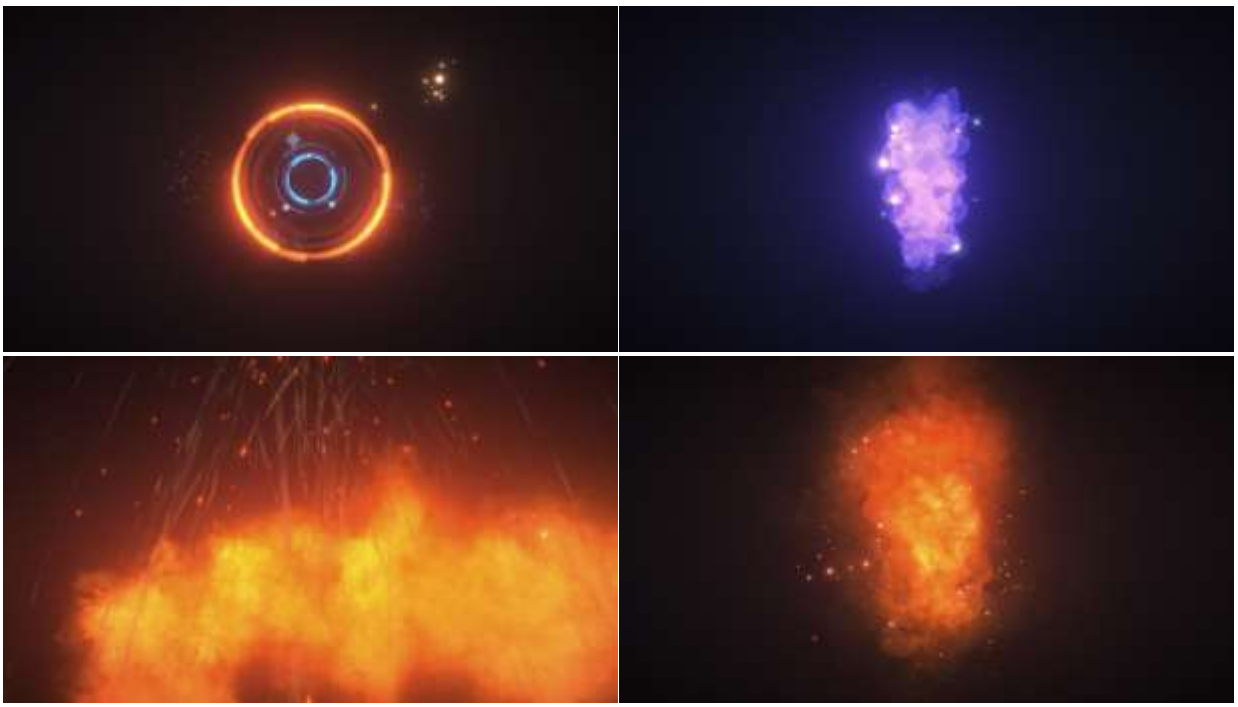
ULTIMATE VFX – (*V2.6.5*)

(DOCUMENTATION LAST UPDATED MARCH 05TH, 2017 – PLEASE VISIT THE LINK BELOW)
HTTP://WWW.MIRZABEIG.COM/PRODUCTS/ULTIMATE-VFX/DOCUMENTATION/

CONTENTS

This document will walk you through the entire Ultimate VFX library and how to best take advantage of it.

INTRODUCTION



SCREENSHOTS FROM INSIDE THE EDITOR USING **EXISTING PREFABS** FROM ULTIMATE VFX WITH ONLY **UNITY'S OWN IMAGE EFFECTS**. MASSIVE **4K WALLPAPER VERSIONS ALSO AVAILABLE HERE**!

Ultimate VFX ships with **over 300 prefabs** with intensive care put into the visual look and feel of each effect. Created from a **massive library of over 200 textures** (including several spritesheets/atlases, so the individual texture count is actually *much* higher – going into the ***thousands***), **with a combined size of over 600MB**, you can easily create your own effects by modifying and tweaking the existing prefabs, or creating your own from scratch.

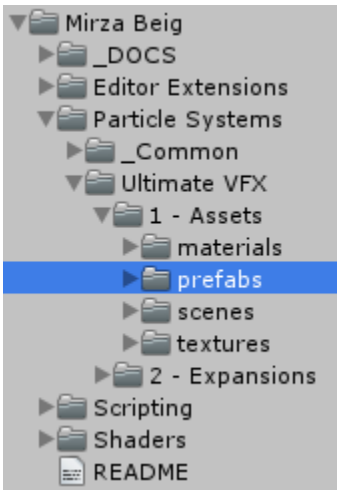
Originally released at the very start of 2015 with Unity 4.6, Ultimate VFX (aka UVFX), has since been steadily **increasing in both quantity and quality**, based on user experience, feedback, and even improvements that have been introduced into Unity itself. As **the majority of effects are created with the default Particle System component ("Shuriken"),** you'll have no problem working with UVFX's prefabs.

Although Ultimate VFX is *primarily* a collection of particle effects, it **includes *much* more**. From **custom scripts for manipulating particles with *realtime* forces**, to a **twister-creating editor extension** with simple controls allowing you to **easily make particle-based galaxies and tornadoes**! **UVFX is a complete solution**, even **including custom particle shaders**.

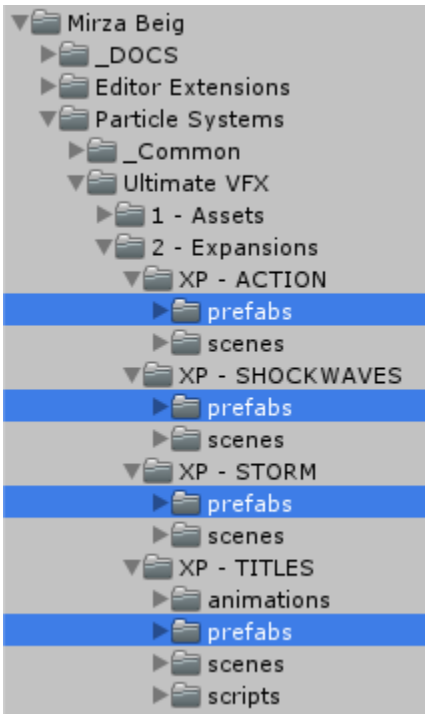
QUICK-START / LOCATING PREFABS

Because of the inherent nature of having a large asset such as this, along with other similar assets from myself that are designed to seamlessly import into the **same** project (cross-product compatibility!), the folder structure may, at first, be a little confusing.

If you'd rather just jump in to the **prefabs** and don't care so much about an explanation for the folder structure, simply refer to the screenshot of a fresh import below:

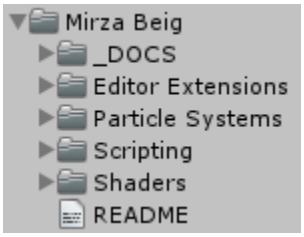


For even ***more*** **prefabs**, just open up the respective expansion folder(s):



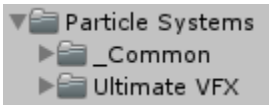
ASSET PROJECT HIERARCHY

When you first import UVFX into Unity, this is what you should see:

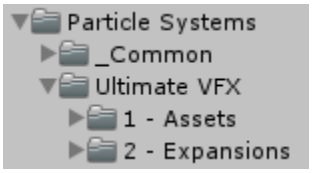


All my assets will import into the root folder of "Mirza Beig". UVFX ships with several additional bonus assets such as shaders, plugins, and even themed prefab expansion packs. These are sorted into their respective folders. If, for example, you download additional plugins I've made, you'll find these under "Editor Extensions". Likewise, **particle system assets from me on the store will import into the "Particle Systems" folder – this is where you'll find UVFX**.

Opening up the Particle Systems folder, you should see this:



"_Common" simply has assets that are not specific to a single asset package and will be shared by all my particle-based works for demo purposes only.

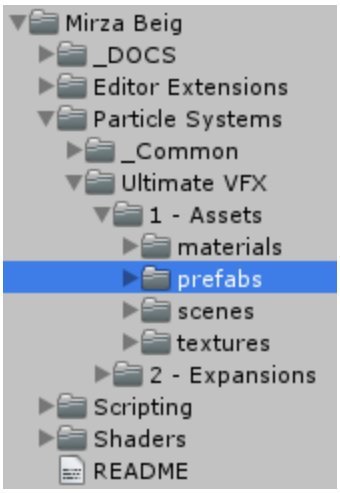


Opening up "Ultimate VFX" you'll see two additional, ordered folders…

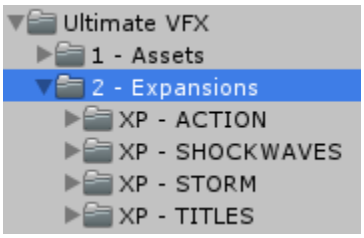
1. **Assets** – Where all the main materials, textures, and prefabs are.
2. **Expansions** – Where you'll find all the related expansion packs and their prefabs that use the main assets (created using the textures, materials, and scripts from "1 - Assets").

EXPANSION PACKS – THEMED PREFAB SETS

With so many high-quality textures included as a baseline with UVFX, it would be a waste to not continue creating even more effects using them! Although the main showcase is constantly updated, those prefabs aren't *necessarily* production ready (too many particles, lack of attention to overdraw, etc.) and are more for showing what *can* be achieved with the content. Of course you may – and are encouraged! – to make your own effects using the showcase prefabs (the ones from the screenshot below) as a starting/reference point. But instead, perhaps you're someone who's looking for ready-to-use prefabs with minimal hassle.

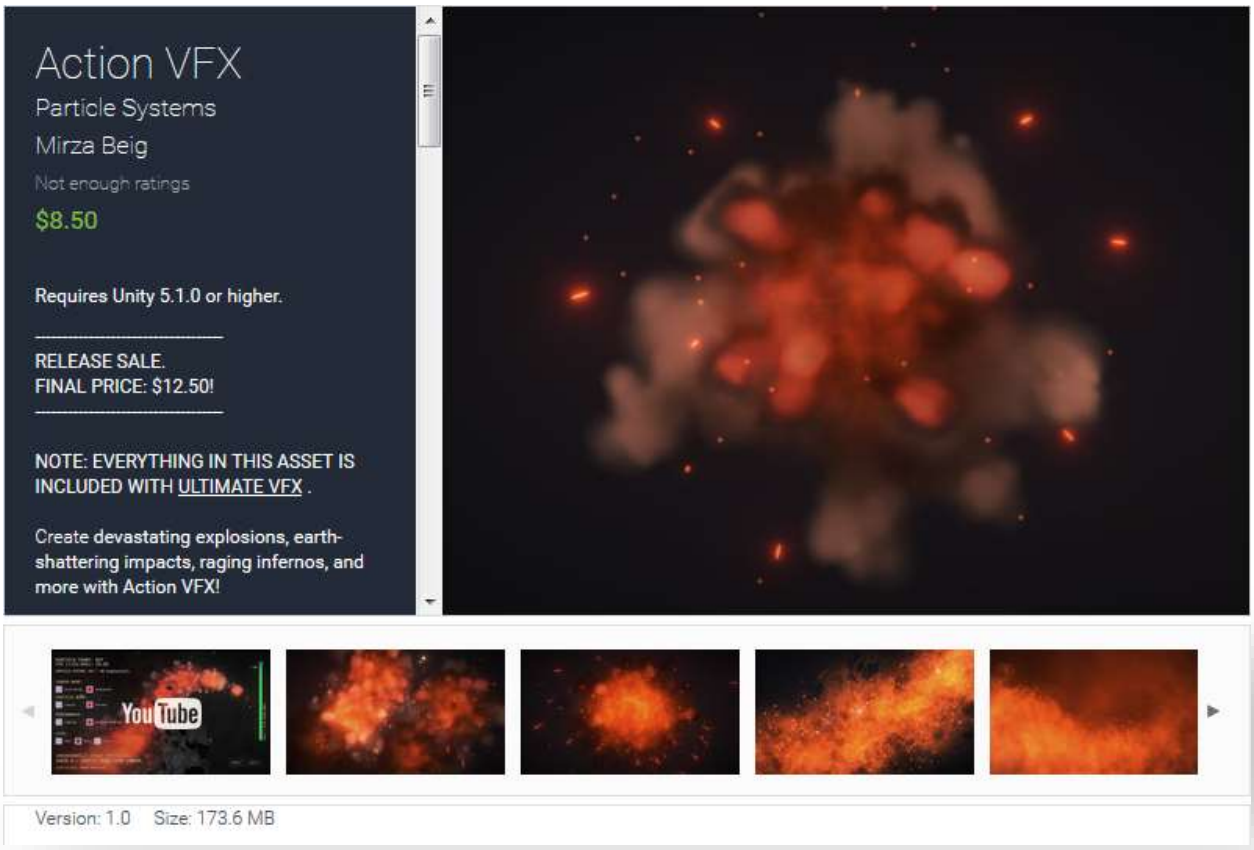


This is where expansion packs come in.



Similar to the prefab showcase, these are carefully constructed prefabs that are not only amazing in and of themselves, but are optimized so as to use less particles and reduce overdraw, all without sacrificing visual quality! They come in sets based on a popular theme, such as with **XP Action** for fiery explosions, and volumetric-looking smoke stacks, as well as **XP Storm** for torrential rainfalls, cold thunderstorms, foggy environments, and other weather/storm-based effects.

Some expansions, like the aforementioned **XP Action and XP Storm, are <u>included</u> with Ultimate VFX**, so you can get started right away without worry. In addition, they may *also* be sold separately as standalone VFX packages that would only contain the textures and materials required to build the effects included with them.



NOTICE HOW XP ACTION IS ALSO SOLD ON THE ASSET STORE AS A STANDALONE EXPANSION FOR NON-OWNERS OF UVFX. **YOU GET ALL THESE AWESOME PREFABS FREE!**

In the future, **more FREE expansion packs are planned to be added** based on feedback (such as a sci-fi themed set), along with paid expansions at **significantly reduced prices for UVFX owners only** (like **YOU!**) and at some other, "regular" price point for non-owners who may purchase the same expansion pack as a standalone asset.

These are heavily based on existing customer feedback, so please refer to the **forums** if you'd like to see a particular theme set created and added as a potential update in the future.

KEYWORDS – THE SECRET TO SUPER-FAST VFX CONSTRUCTION

With such a large collection of effects (along with several expansions), it helps to be able to quickly find what you're looking for without having to manually navigate through so many folders. With this in mind, all the prefabs, materials, and textures (the main components required for creating particle effects inside Unity) have a unique naming scheme that allows you to type in a few letters/characters to find what you're looking for in the project view search bar.

Below is a breakdown of the keywords you can use (don't include the quotations):

- "**pf_**" – Prefab.

- "**vfx-**" – Visual effect. Used alone, may result in materials, prefabs, and/or textures.
    - "**vfx-ult_**" – Ultimate VFX.

- "**xp-**" – Expansion pack.
    - "**xp-action**" – Action expansion pack.
    - "**xp-storm**" – Storm expansion pack.
    - "**xp-titles**" – Titles expansion pack.
    - "**xp-shockwaves**" – Shockwaves expansion pack.

- "**mat_**" – Material.
- "**tex_**" – Texture.

Note: You can sometimes **omit** the **dash or underscore** at the end, depending on your usage with other keywords. See compound keyword examples below for…examples. Generally speaking, however, you only need to type as much as it takes for the project view to show you what you wanted to find anyway.

COMPOUND KEYWORDS (SAMPLES)

"pf_vfx" – Show ***all*** the VFX prefabs from all my VFX-related assets. **No exceptions**!

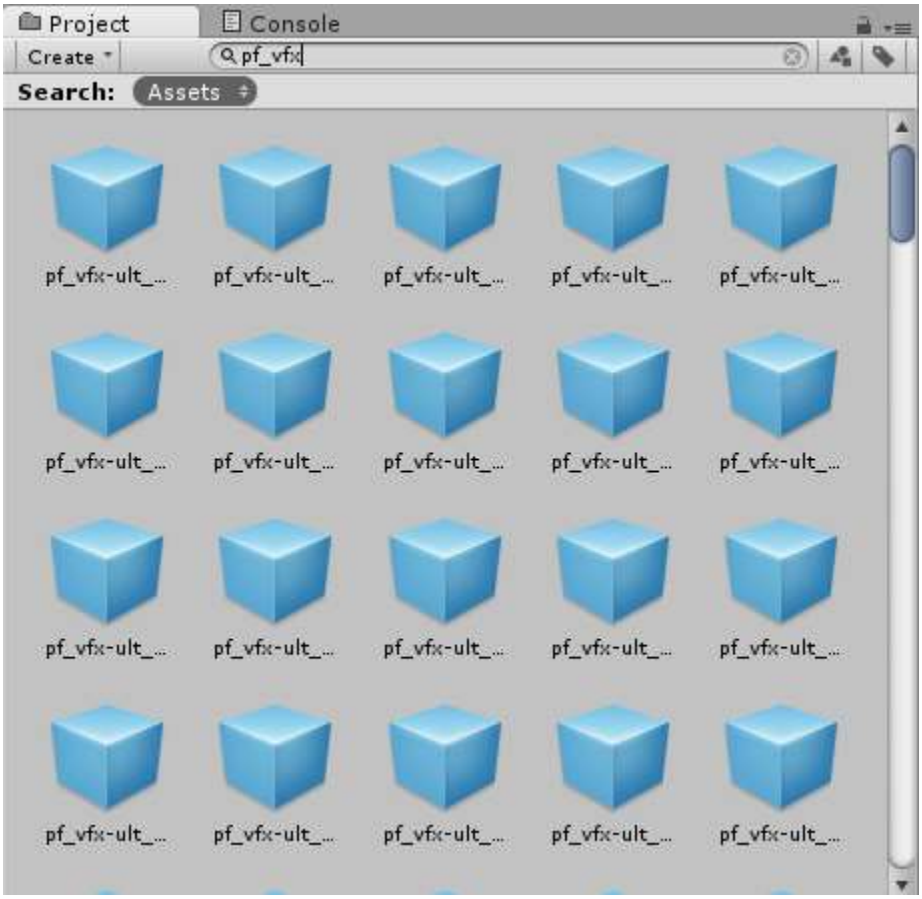
"pf_vfx-ult" – Show only the **Ultimate VFX** prefabs!

"pf_vfx-ult_xp" – Show only the **Ultimate VFX expansion pack** prefabs.

"pf_vfx-ult_xp-action – Show only the **Ultimate VFX Action expansion pack** prefabs.

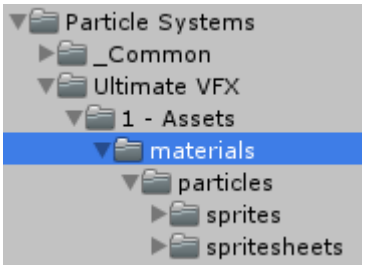
…and, well, you get the idea!

When you need to find something, use the appropriate keyword to have it itemized for you by Unity. In the example screenshot below, typing in "pf_vfx" will list out **all** the VFX prefabs in your project! It's a simple but powerful system for finding exactly what you need.



MATERIAL PRESETS

UVFX comes with several pre-configured material variations.



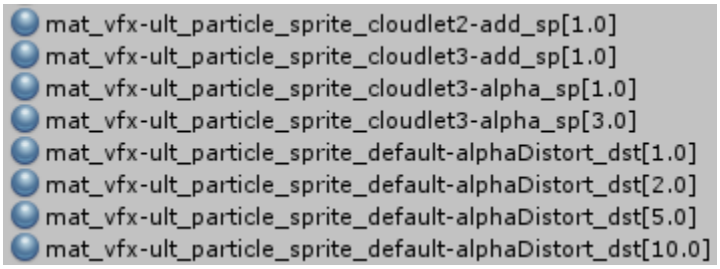
This is useful so you don't have to constantly duplicate and change the existing materials assigned to particle systems for a particular texture, and instead you can just swap the material with one of its variants.

As an example, let's say you want tweak an existing smoke-based particle system. Obviously, it would be using a material with a smoke-y texture applied to it. Even so, maybe the soft particle value isn't suitable for your needs. A soft particle factor of 1.0 is too much or too little for how you want to use this particular effect, so now you have a choice – either leave it as-is, or create a new material somewhere in your project, hence creating more clutter and disjointing the base assets used for the VFX package you've downloaded.

Even if you're creating your own effects from scratch using the textures in UVFX, you'll find that having **several material presets** available is a ***huge*** **timesaver**.

With UVFX, however, you can just swap out the material for the best matching variant and save yourself the hassle. This is a problem I consistently ran into myself, which is why I decided to create so many different (but common/most-likely) configurations.

Materials have their own keyword identifiers to help you quickly find out what settings are used:



Here you'll see just three textures (cloudlet2, cloudlet3, and the default Unity particle texture) having ***eight*** **different material configurations** available. Where add/alpha/alphaDistort are the names of the shaders, "_sp" stands for the soft particle factor, and "_dst" means the distortion amount (only for distortion based shaders – ***included*** with UVFX).

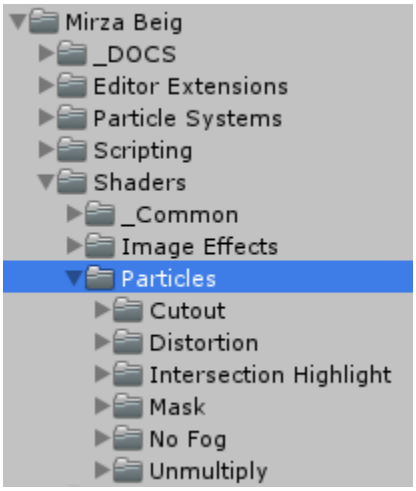
Generally, for materials, **the last keyword is the main parameter followed by its value**. Soft particle factor (_sp), distortion (_dst), etc.

PARTICLE SHADERS

Most of the particle materials use the default Unity particle shaders (additive, alpha blended). Sometimes, that's not just not good enough. Sometimes, you need ***more***.

More control, **more** options, **more** of that *wow*-factor.

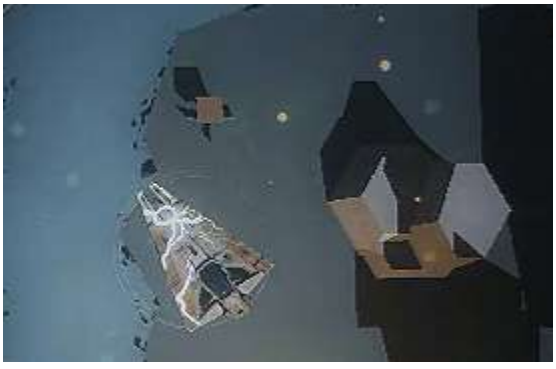
UVFX ships with several additional shaders specifically made to allow you to unleash your creativity.



These shaders are actually based on examples from AAA titles such as the "*Kingdom Hearts*" series and "*InFAMOUS Second Son*". After watching several VFX videos of commercial game titles, I decided to include shaders that would allow anyone to easily **recreate similar effects using just particles**.

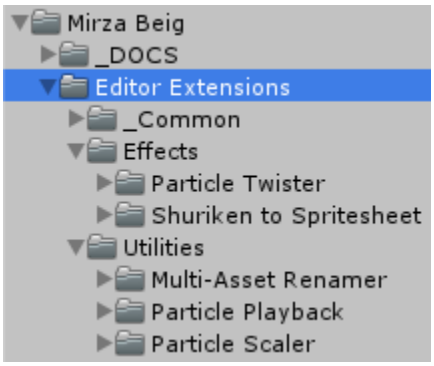
As an example, the distortion shader is common in many games (distorting anything rendered behind the particle), and the **intersect highlight shader** **(click for a GIF!)** was specifically inspired by *Second Son*.

Although these shaders provide several new ways to show off your VFX, they are nevertheless based off the default Unity particle shaders, so **you don't lose any of the original options** (soft particles, shader colour tinting, and so on). **You only get more!**

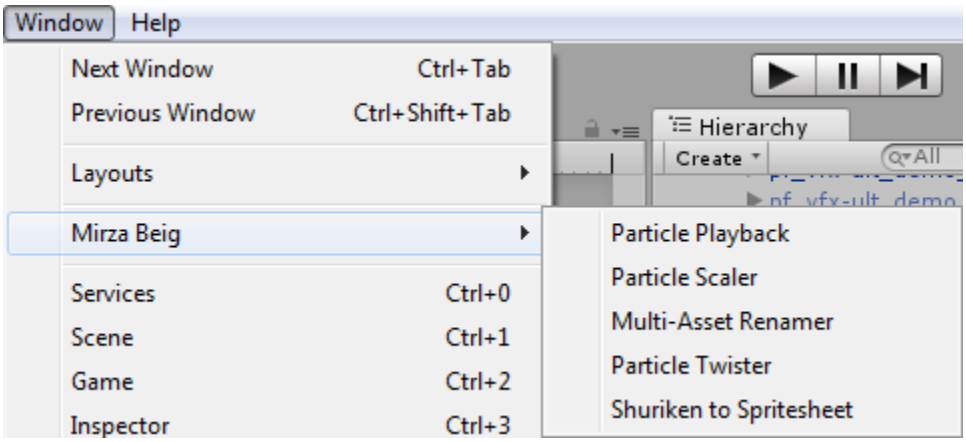


PLUGINS

Most of the plugins included with UVFX can be used to manipulate particle systems directly. Below is where to find them in your project:



And how to open up their respective editor windows:



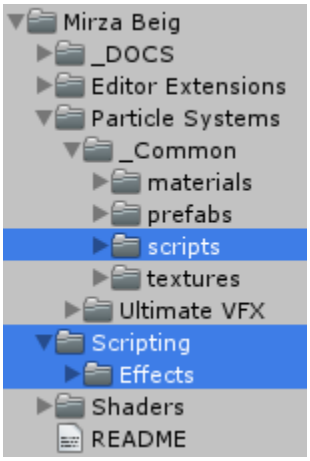
TODO: …explain each plugin in detail.

SCRIPTS

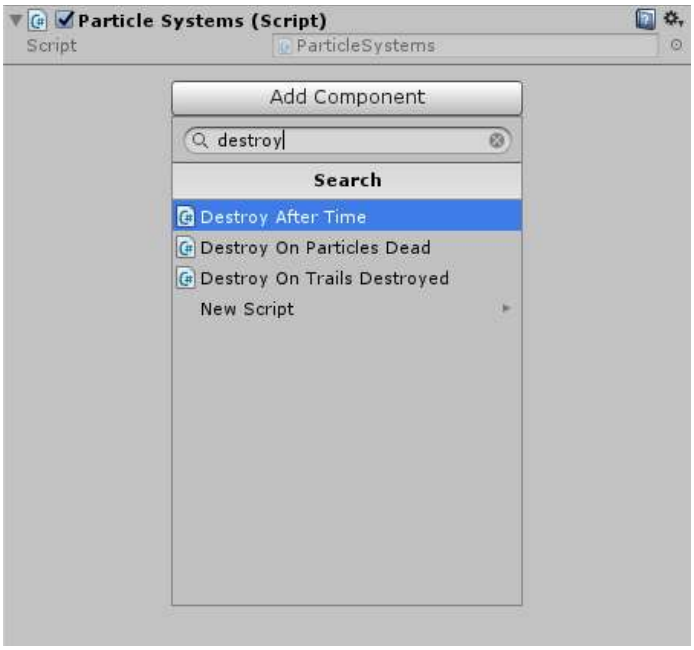
This section will walk you through the most useful of the many scripts included with this package, and how they can augment and accelerate your VFX.

There are two folders to consider where the majority of the components will be found:

1. "_Common" scripts.
2. "Scripting" Effects scripts.



The following pages will outline how to use these components, along with what they do. The easiest way to use them is to simply add them as a new component by name (like with any other MonoBehaviour):



- ParticleSystems

Notice the plural – it's ParticleSystem**(s)**, not ParticleSystem. Used to control several ParticleSystem components that are parented to a common GameObject.

For this component to be useful, there should be *at least* one particle system either on the game object itself, or on child objects:

Has several public functions that are shared with its single-particle system counterpart.

Almost every looping prefab in this asset has this component attached to it for *the int getParticleCount()* function which returns the combined particle count of every particle system in the hierarchy that is shown in the demo scenes. Feel free to remove this script.

Also contains an *isAlive()* method which can be used to determine if the entire hierarchy of particle systems is finished animating. See *DestroyOnParticlesDead* for a child of this class that makes use of this.

- DestroyOnParticlesDead : ParticleSystems

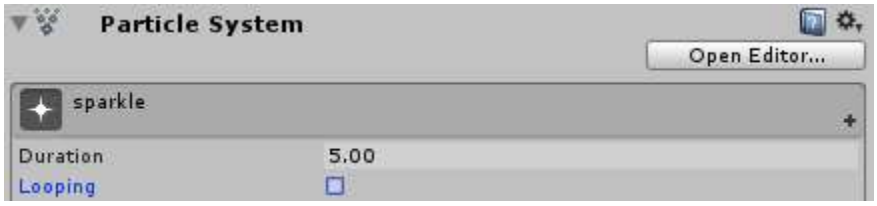
Inherits from ParticleSystems. **Generally used to create one-shot particle effects** which are instantiated, then automatically destroyed when the effect is finished playing.

Common examples: explosions, muzzle flashes, shockwaves, most weapon impact fx, gore, etc.

Attach this on either a single particle system, or a nested group (same as you would with the ParticleSystems component) to ensure the object is destroyed when all particle systems under its control are no longer "alive".

A particle system is considered "dead" when it has been alive for at least its duration (in seconds), it has no particles left to emit, and all particles that have already been emitted are dead.

For this reason**, looping *must* be set to false for EVERY system within the hierarchy,** and the duration may *optionally* be set to the lowest possible time required:

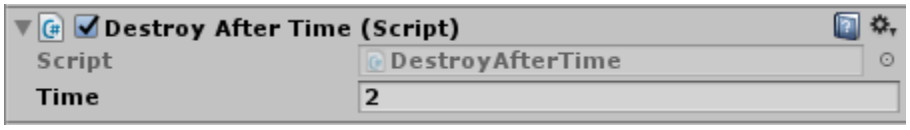


Almost every one-shot prefab in this asset has this script attached. If you plan on using these prefabs as-is, you should keep this component attached to prevent the object from lingering in the scene for longer than it has to.

Alternatively, use the *DestroyAfterTime* component for a non-particle specific, and less precise counterpart if you know the object shouldn't be alive for longer than a specified period of time after being instantiated.

- DestroyAfterTime

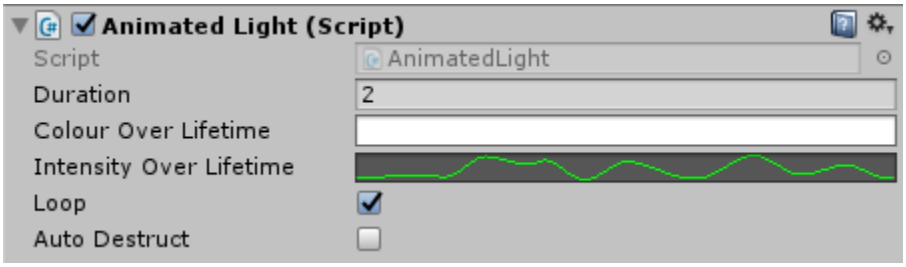
A generic component that will automatically destroy the object it's attached to after the specified amount of time after it is first created (either by already being in the scene at start, or through being instantiated).



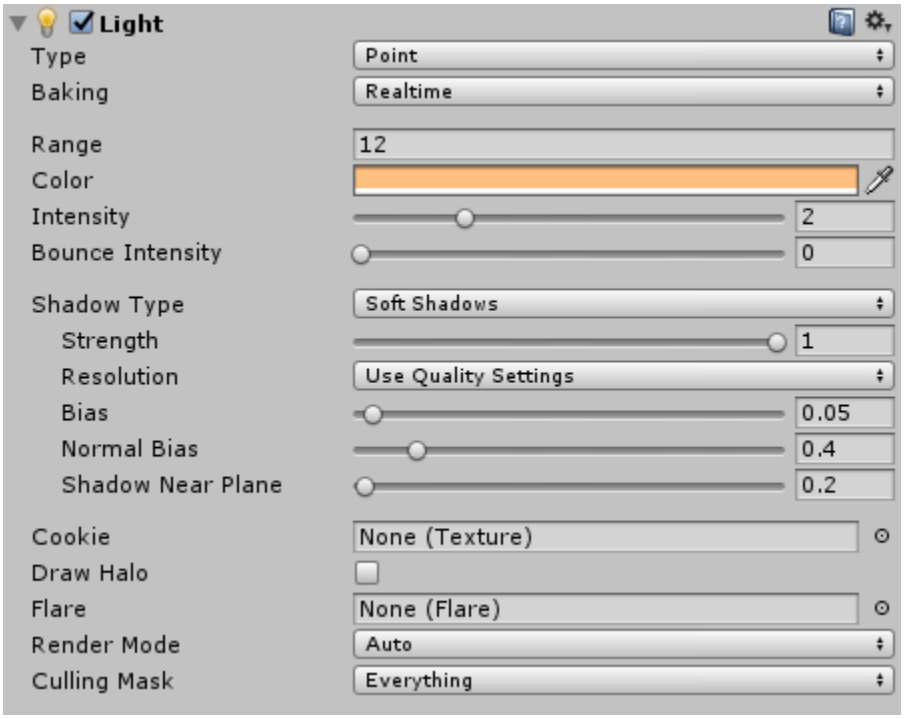
- AnimatedLight

Use this to easily animate lights for particle effects without having to delve into Unity's more complex animation system. Almost all prefabs in this asset have at least one (point) light attached with this component. **Lights, in general, can be very expensive**.

Set loop to false and auto-destruct to true for one-shot effects.

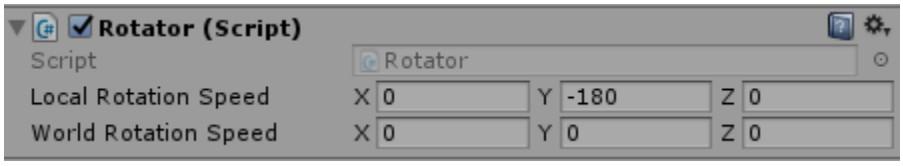


Adding this component to an object will automatically add a Unity light component if there isn't already one attached. For performance, you may wish to **disable shadow casting**, as these can be **extra expensive for low-end hardware.**

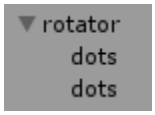


- Rotator

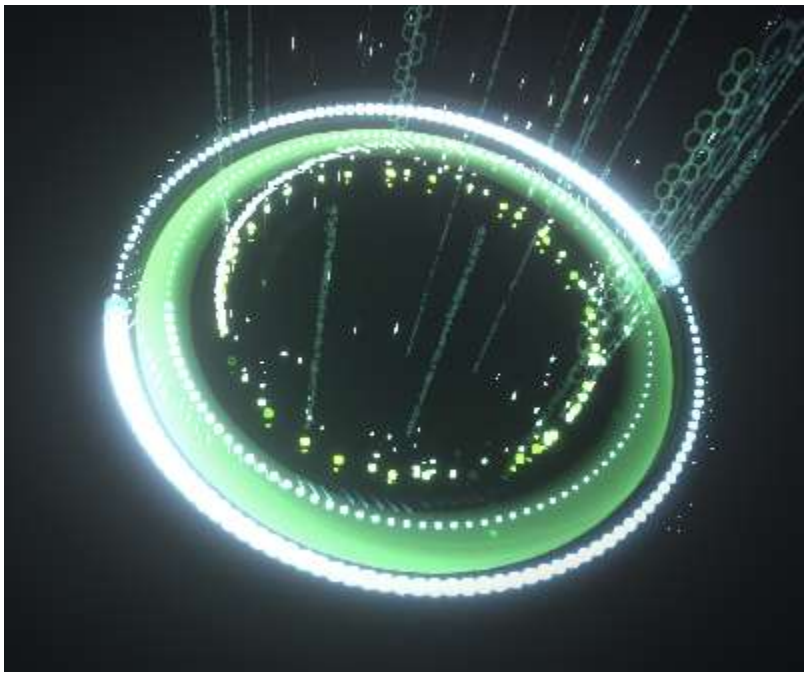
A generic script component used to rotate an object by some amount (in degrees) per second around an axis in either local or world space. In the image below, the object will rotate -180 degrees per second around its own up axis (same as transform.up).



Rotator can be used to create interested rotating trail effects with particles. For this effect, the rotator game object has two child particle systems:

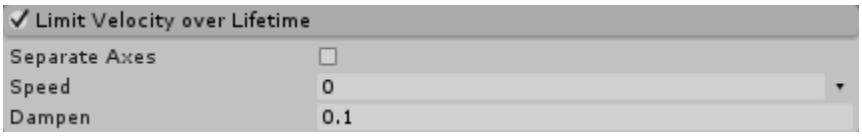


...with their **simulation set to world space**, and their **emission to some amount by distance** (not time). The first has a Z-offset of 1, while the other has a Z-offset of -1, such that they are on opposite ends.



ADVANCED EFFECTS: PARTICLE AFFECTORS

All particle affecters apply their own *additive* forces to particle systems. It's best to enable and tweak the Limit Velocity over Lifetime module for all particle systems affected to prevent overdriving their speeds.



As a start, try these values.

- VortexParticleAffector

…

- TurbulenceParticleAffector

…

- AttractionParticleAffector

…

Animated GIF #1 (*turbulence field*):

http://gfycat.com/OrganicBarrenDikkops

Animated GIF #2 (*turbulence "missiles"*):

http://gfycat.com/TheseFemaleIrishwolfhound

Animated GIF #3 (*turbulence* + *vortex* + *attractor*):

http://gfycat.com/AdmirableFreshAmericantoad

Animated GIF #4 (leaf storm):

http://gfycat.com/InsidiousThreadbareAfricangroundhornbill

Animated GIF #5 (leaf storm 2):

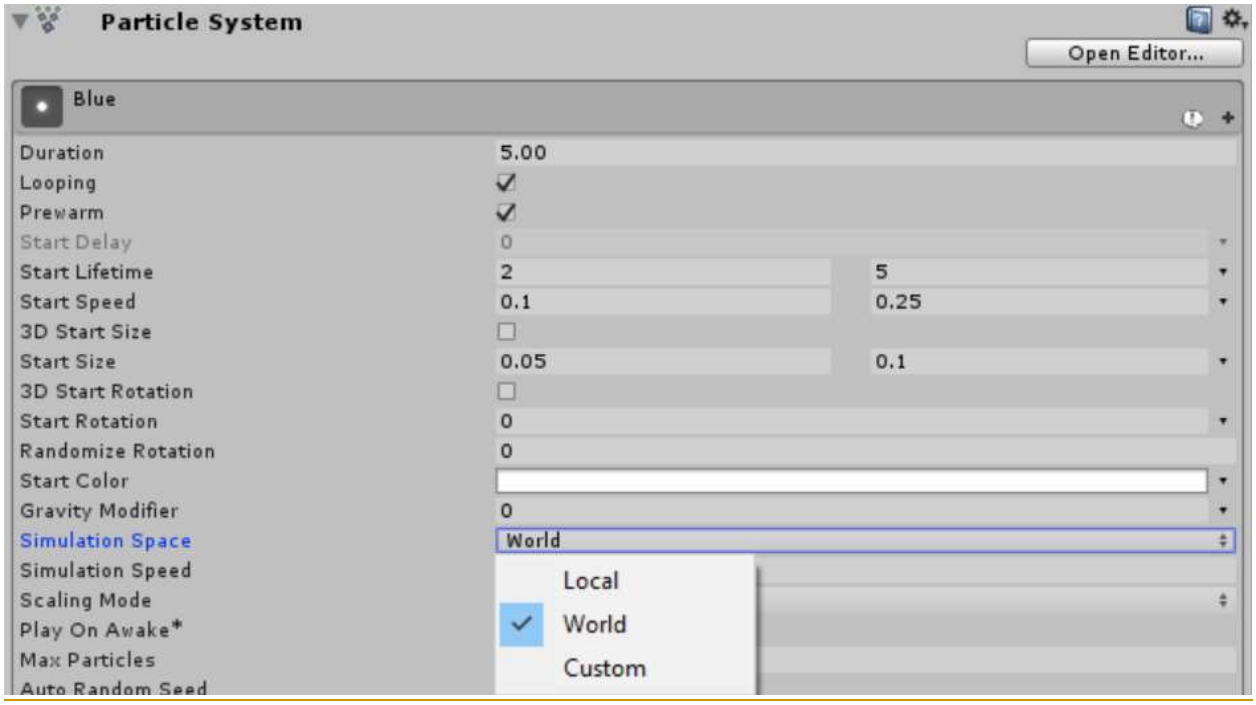
http://gfycat.com/ExaltedSecondaryGalapagospenguin

Animated GIF #6 (leaf storm + lit particles):

http://gfycat.com/EnragedFirmBaldeagle

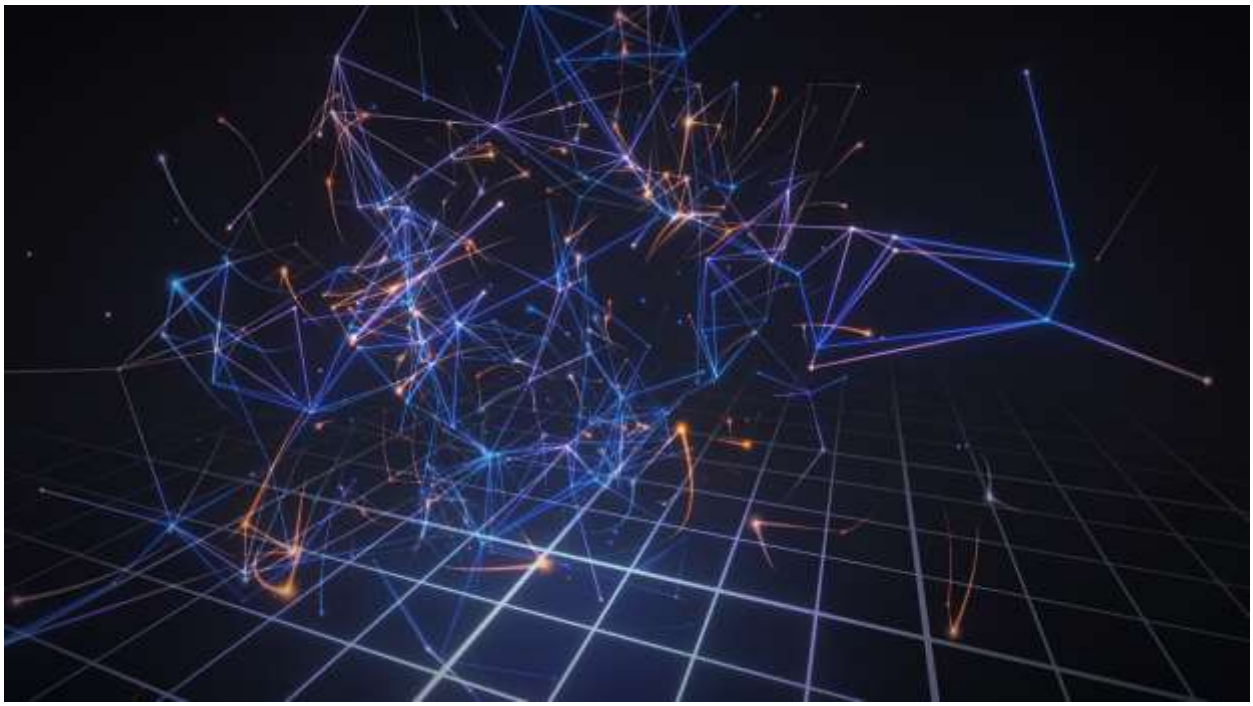
PERFORMANCE TIP:

- If possible, set the particle system's simulation space to World for better performance. This saves the component from having to calculating individual particle positions, which can be a significant performance impact for large sets.



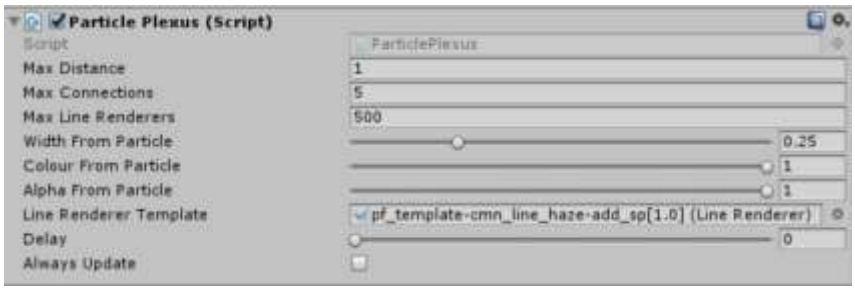
- ParticlePlexus

Simulates the "Plexus" effect (an After Effects plugin), connecting all particles within a user set range to each other using Unity's LineRenderer component.

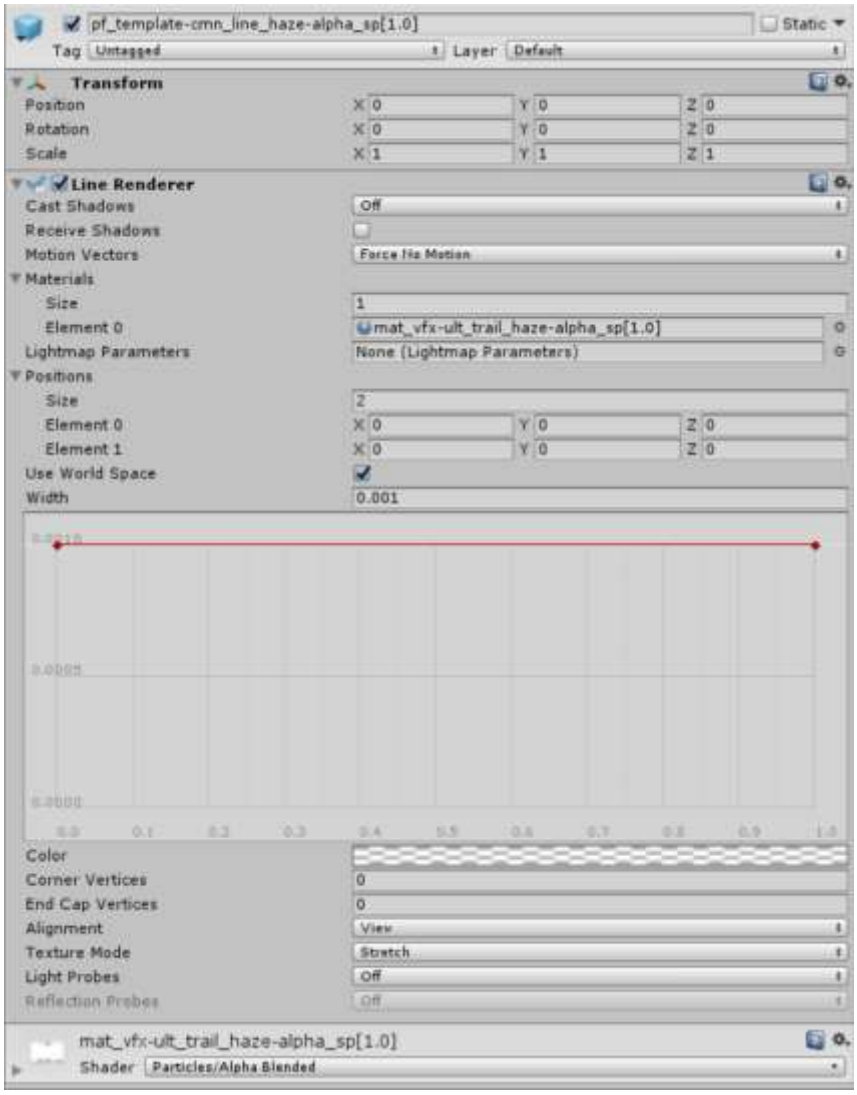


Simply add the component to any game object with a ParticleSystem component attached to it (one will automatically be added if it doesn't exist) and tweak the settings until you get the look you're going for. You'll need to be in play mode to see the effect.

The component requires a line renderer template, from which it will create the connections, but you can use the included templates (look at the image below) to save time:



If you want to create your own template, simply create a new game object, attach a LineRenderer component to it, and save that as a prefab. You can tweak the settings as you please at any time:



Max Distance: the radius around a particle to look for other particles.

Max Connections: the maximum number of line connections to make per particle.

Max Line Renderers: the total limit of lines that can be made within this system.

Width From Particle: 0.0 = use only the template width, 1.0 = use only the particle width. Any value between 0.0 and 1.0 will be a linear mix between the value from the template and particle.

Colour From Particle: 0.0 = use only the template colour, 1.0 = use only the particle colour. Any value between 0.0 and 1.0 will be a linear mix between the value from the template and particle.

Alpha From Particle: 0.0 = use only the template alpha, 1.0 = use only the particle alpha. Any value between 0.0 and 1.0 will be a linear mix between the value from the template and particle.

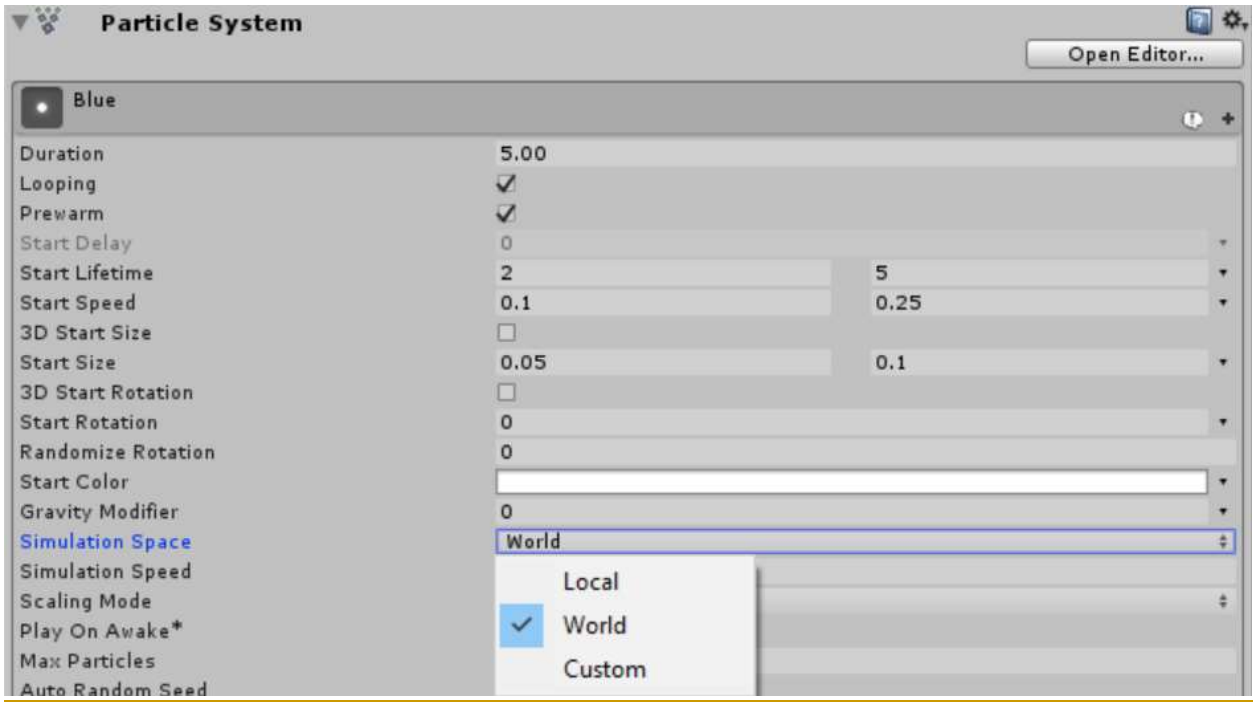
Line Renderer Template: the basis from which all line connections will be instantiated. Ideally a prefab, but can be a local game object in the scene (make sure not to leave a null reference).

Delay: update delay in seconds. Can be useful for performance, but is mostly an aesthetic parameter.

Always Update: this component will not update when all particles are out of view (culling). Check this to force an update even when the particles are not seen (*not recommended* for performance).

PERFORMANCE TIP:

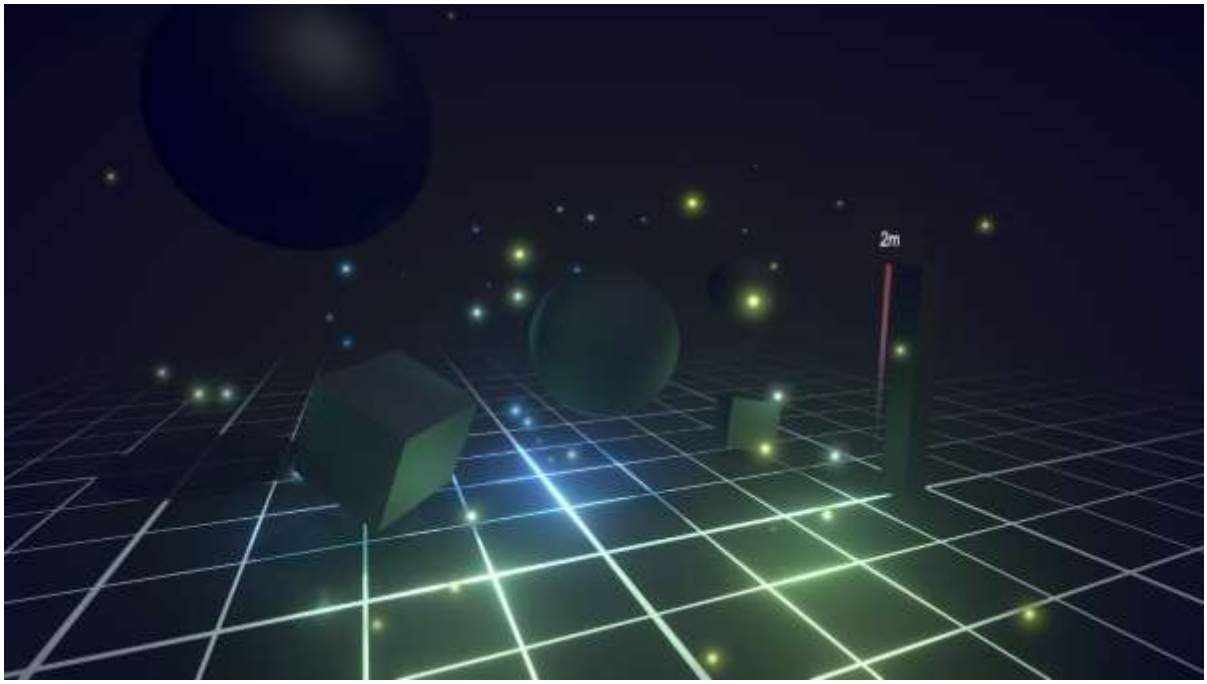
- If possible, set the particle system's simulation space to World for better performance. This saves the component from having to calculating individual particle positions, which can be a significant performance impact for large sets.



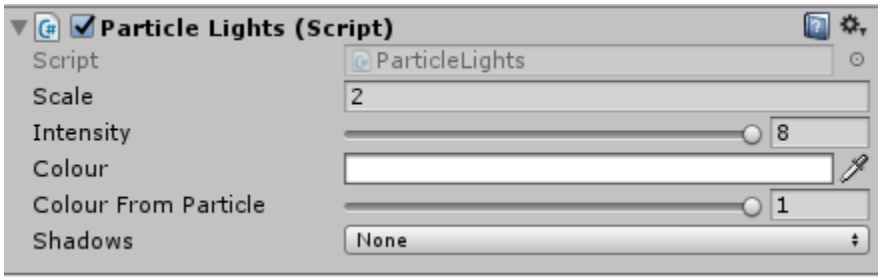
- ParticleLights

Best used with deferred rendering.

This extremely easy-to-use, versatile component enables real light emission from particles. Simply add it to any game object with a pre-configured particle system attached to it; alternatively, if this isn't done, one will be added for you automatically.



For every active particle in the system, a light source will follow it, adapting its position, colour, and size (as the light range). The remaining common parameters can be controlled from the ParticleLights component itself. This ensures maximum flexibility as you can easily tweak the particle system to your liking and the respective light sources will change accordingly.



Scale: used to adjust the relative range of your lights (range = particleSize * scale).

Intensity: affects the immediate brightness of your lights.

Colour: used to tint the particle light colour.

Colour From Particle: 0.0 = use only the component colour**,** 1.0 = use only the particle colour. Any value between 0.0 and 1.0 will be a linear mix between the particle colour and the colour set in the component.

Shadows: change the shadow type for each light. **This can be <u>extremely</u> expensive!**

PERFORMANCE TIP:

- As a general rule**,** it's better to have large numbers of small lights and/or a small number of large lights. And don't enable shadows per-particle light unless you absolutely need to!

TEXTURE IMPORT SETTINGS

In Ultimate VFX, Sprites (single texture images) and spritesheets (multi-frame, animated image sequences) have their own common texture import settings which are shared by the same types. Sprite have one setting, and spritesheets have another, **regardless of the *actual* resolution and data of the textures.**

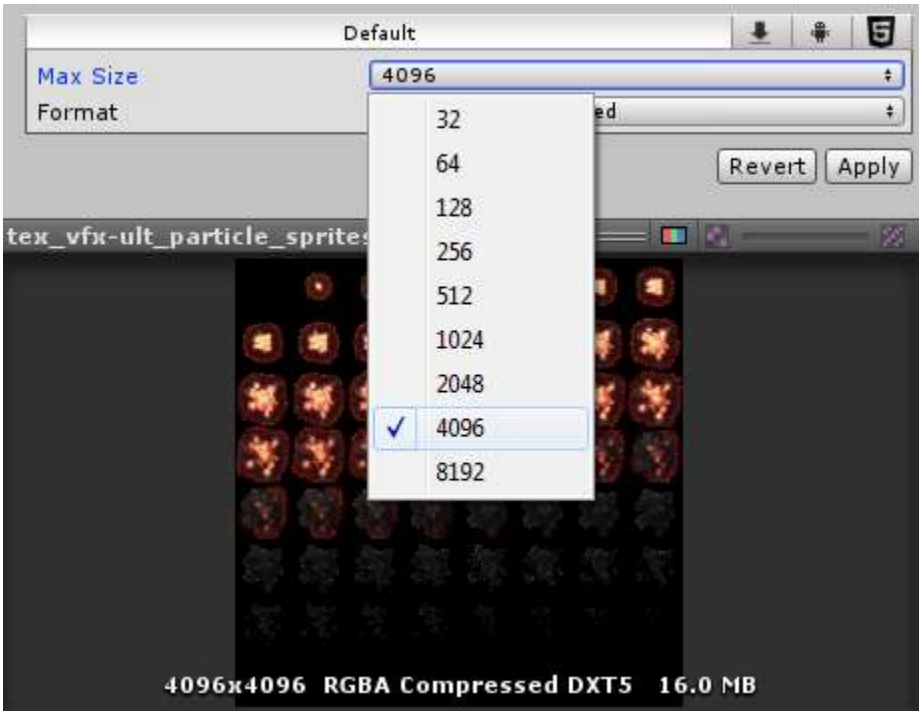
For **sprites**, the maximum **import** resolution is **1024px**.
For **spritesheets**, the maximum **import** resolution is **4096px**.

However, this is *only in the editor*.

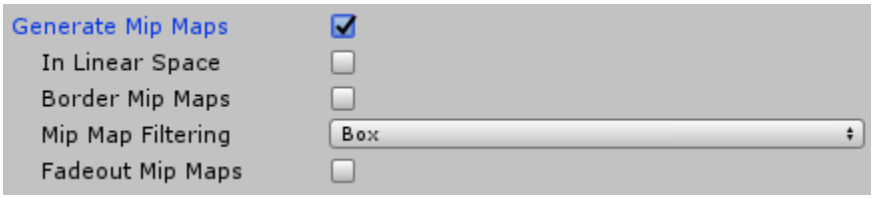
The ***actual*** maximum resolution (the largest texture size) for both sprites and spritesheets is actually much higher. It is only reduced in Unity's importer to allow for faster load times when first importing this asset into your project.

For **sprites**, the maximum **actual** resolution is **4096px**.
For **spritesheets**, the maximum **actual** resolution is **8196px**.

This means that, should you need a higher resolution texture, such as if you're working on a next-gen title with massive target display resolutions (1080p, 4k, etc.), **you can easily change the resolution in the importer to the texture's actual resolution for higher quality**.



Additionally, you should enable mipmaps for extra-smooth textures:



Note that doing so for each of these options will increase the memory requirements of your build.

Of course, **the reverse is also true**. If your target platform is lower-end hardware with smaller screens (such as for mobile) and less texture memory, **you can safely decrease the resolution in the texture importer**. *This does not affect the original texture and you can always change this back*!

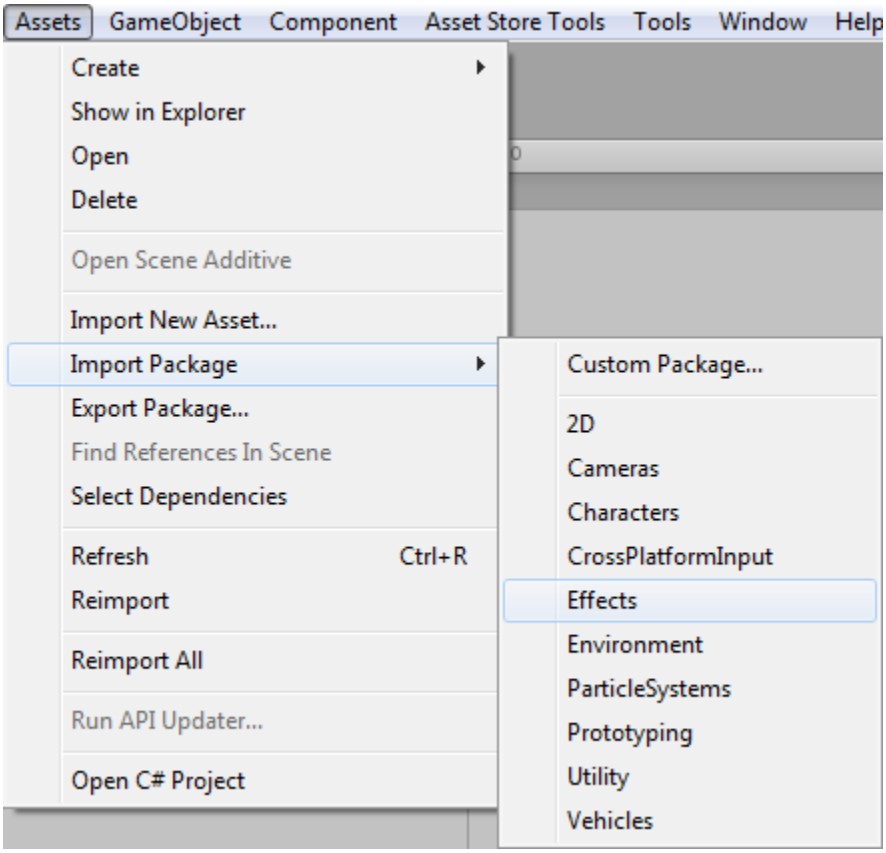
You can also force a stylized look by disabling mipmaps, turning down the resolution to the lowest, or near-lowest settings, and using point filtering – all from the texture importer! See below:

STYLIZED/PIXELATED LOOK USING LOW-RESOLUTION TEXTURE IMPORTER VARIANTS WITH POINT FILTERING. **ALL FROM WITHIN UNITY!**

IMAGE EFFECTS & POST-PROCESSING

To keep this asset clean, easy to update, compatible for end-users, and independent, I do not include any external (read: not created/owned by me!) assets, including the ones from Standard Assets.

That being said, the image effects *components* are still attached to the camera, so importing the Standard Asset image effects from the top-most menu under *Assets* ➔ *Import Package* ➔ *Effects* along with the Cinematic Image Effects from Unity will allow you have the exact same setup as those used in the demos.



If these are not imported to your project *prior* to importing UVFX, you will receive some missing script component warnings *on the **first** import only*.

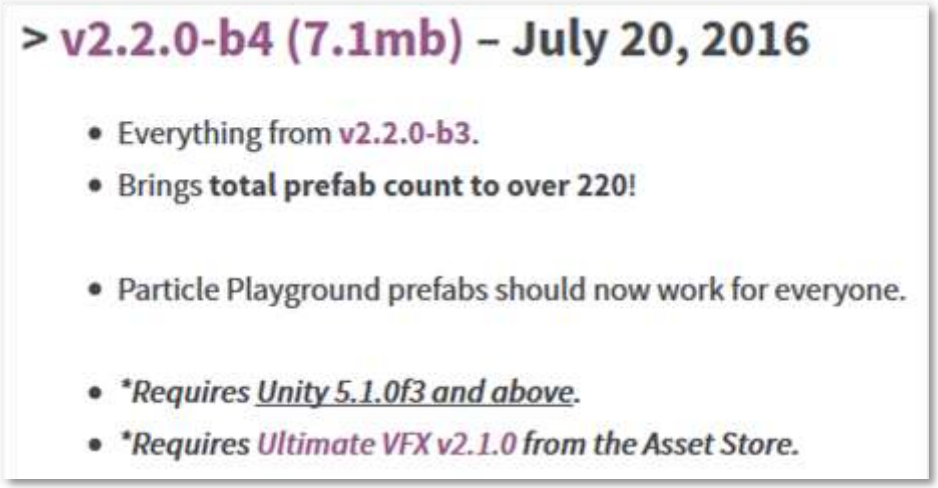
UPDATES / PATCHES

Updating a large asset on the store can take some time. Along with having to carefully setup a new draft and then uploading the **entire** asset again, **the review process for updates can be lengthy** at times.

For this reason, this asset is structured in such a way that makes it extremely easy to apply small patches to either fix issues or add new content to the set.

You can find these **HERE**.

Simply download the update/patch *.unitypackage and apply it to the same project in which you have ***already*** imported Ultimate VFX from the Asset Store.



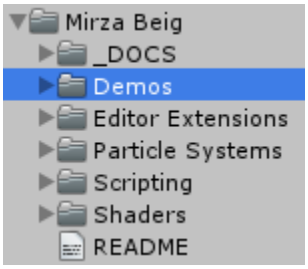
A TYPICAL EXAMPLE OF AN UPDATE.

With these patches, you **save valuable resources** that would be otherwise wasted having to wait for new content or known fixes to be pushed to the Asset Store and/or constantly re-downloading large files. Below is the full web address:

➤ **HTTP://WWW.MIRZABEIG.COM/PRODUCTS/ULTIMATE-VFX/PATCHES/**

EXTERNAL DEMOS

Ultimate VFX comes with additional demos that can be downloaded separately and imported into your projects. You must first import Ultimate VFX, *THEN* the demo package. The demos are downloaded to the folder highlighted in the image below:



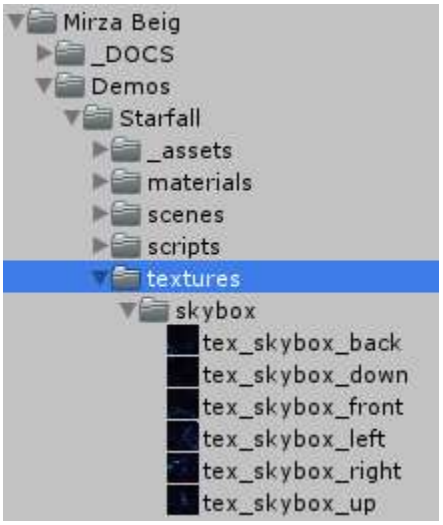
These packages come with additional assets that are either owned by me, or I have the rights to redistribute freely for commercial use (such as assets on the store by Unity or those in the public domain). They are not included with the main UVFX package download to keep the asset hierarchy clean and to prevent any licensing conflicts.

Use these demos to understand how to better use Ultimate VFX in real game environments!

SCREEN CAPTURE FROM THE DOWNLOAD-ABLE **'STARFALL'** DEMO.

IMPORTANT: All external assets not owned by me for the demo will be in the "_assets" folder and are NOT covered by the same license as Ultimate VFX:

Assets *outside* this folder for the demo are owned by me and are covered by the same license as Ultimate VFX (you may use them commercially for your own game(s), etc). Such as the textures below:



TODO: setup demo download page…

TUTORIALS

Several tutorials exist for working with particles in Unity. I also have a few that deal with particles, all the while being more relevant to my own assets (because I use them in the tutorials themselves).

Feel free to check them out!

➤ **http://www.mirzabeig.com/tutorials/**

➤ **https://www.youtube.com/user/TheMirzaBeig**

You'll learn to create effects such as particle-based storm clouds (using XP Storm, the included expansion pack in Ultimate VFX) and ambient glowing particles.

LEGAL / LICENSING INFORMATION

Please refer to the **Asset STORE End User License Agreement**.

CONTACT | FEEDBACK | SUGGESTIONS, ETC.

Ultimate VFX is an evolving asset. It's open to feedback and suggestions to help drive it towards becoming an even better product. The best way to leave feedback is to involve the entire community, so please check out the **forum thread**!

If you'd like to communicate with me directly, please don't hesitate to get in touch with me using either a private message (PM) on the forums, or through e-mail (**mirza.realms@live.ca**).

A FINAL SCREENSHOT WHERE **THE CONTENTS OF THE SCENE ARE MADE UP ENTIRELY OF PREFABS FROM ULTIMATE VFX** (WELL, EXCEPT FOR THAT GRID).