ALL IN 1
VFX Toolkit

By Seaside Studios
For technical support, feedback and requests write to:
seasidegamestudios@gmail.com
Twitter of the creator: https://twitter.com/GerardBelenguer

Index

Overview

First of all thanks for downloading this asset! The asset was created with the goal of providing the best workflow for VFX artists. Both experienced and beginners. The asset offloads most of the technical knowledge needed to create VFX and shaders while providing powerful tools, custom editors, learning resources, extremely flexible shaders and more.

Everything was designed in such a way that experienced VFX artists will have everything they need and want while speeding up their workflow. And beginner artists will be provided with learning materials, examples, a huge asset library and much more! I really believe this is the best way to create VFX available in the asset store and it will only keep improving over time.

The asset has a ton of features, resources and examples but using the main features is extremely easy and intuitive. Here's a link to the youtube Tutorial playlist that explains how to use this asset in case you are a more visual learner, almost everything written in this document is explained there:
https://youtube.com/playlist?list=PLKS0HUbkxp-kigE_Km7CGd2qbfEaoxDLb

Once you've read this document or watched the playlist and you are familiar with the asset you can take a look at this video VFX course:
https://youtube.com/playlist?list=PLKS0HUbkxp-khlRjadbOvOnNUynTZ1pwv
(The course is still in development and more episodes will be added, if you have any idea or there's something you want to learn, please let me know and I'll probably make a video about it and even add it as a prefab to the asset :D)

I'm always open for questions, feedback and suggestions. The best way to contact me is by email. Please don't write questions in the Unity Forums, Youtube videos, Twitter or wherever else since I will probably miss them. I always reply much faster (in 24h or less) by email.
When reaching out please attach your invoice number too and make sure you have read this document or watched the playlist linked above. The email address is:
seasidegamestudios@gmail.com

If you like the asset please make sure to drop a review on the Asset Store page. It helps out a ton:
https://assetstore.unity.com/packages/vfx/all-in-1-vfx-toolkit-206665

First Steps (Must Read)

Before going over the very basics about the asset let's take a look at the install process for the different Render Pipelines.

Built-In Pipeline:

No setup is needed to use the asset in this Pipeline, everything is ready to go after importing the asset. But in order to get everything looking like in the store images, WebGL demo and trailer you'll need to add post processing Bloom and change a couple project settings, here's how:

1. Install Post Processing package from the Package Manager
2. In Project Settings, Graphics Settings enable HDR in all checkboxes

At this point if you open the Demo scene it will look just like in the Trailer, but if you want to replicate the same setup in another scene you can follow these steps:

1. Make sure you allow HDR in you camera
2. Add Post Process Layer and Volume component
3. Add the included AllIn1VfxPp Profile to the Volume component
3. Properly configure them make making it Global and setting the Layer to the same Layer of the Camera you are using
**You can also watch a video about it here:**
https://youtu.be/_zrsScNWXoU

URP Pipeline:

To use the asset normally you just need to follow steps 1 and 2. The rest of the steps are optional, since they are only needed to get the Demo to look like in the store images, WebGL demo and trailer. These are the steps:

1. Make sure that URP is properly installed on the project by either creating a URP project in Unity Hub or by going into the Package Manager, installing Universal RP, creating a Pipeline Asset and finally assigning it to the Graphics Settings tabs
2. Import the "UrpPackage" Unity Package included in the asset root folder
3. Set Color Space to Gamma in Player Settings, Other Settings, Color Space
4. Select the Pipeline Asset and make sure that both Depth Texture (used by Soft Particles and Intersection Glow effects) and Opaque Texture (used by Screen Distortion effect) are enabled:
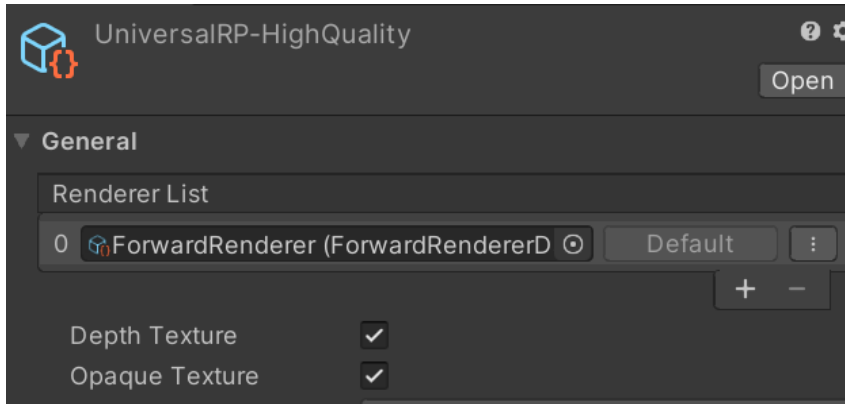
5. Convert Demo environment Materials to URP by going to Edit, Render Pipeline, Universal Render Pipeline, Upgrade Project Materials

6. Convert all Demo Prefab Materials to URP by opening the asset Window (Window, AllIn1VfxToolkitWindow), selecting the Others Tab and pressing Auto Setup Shaders for Materials in selected folder (by default the selected folder is the folder containing all prefab materials):
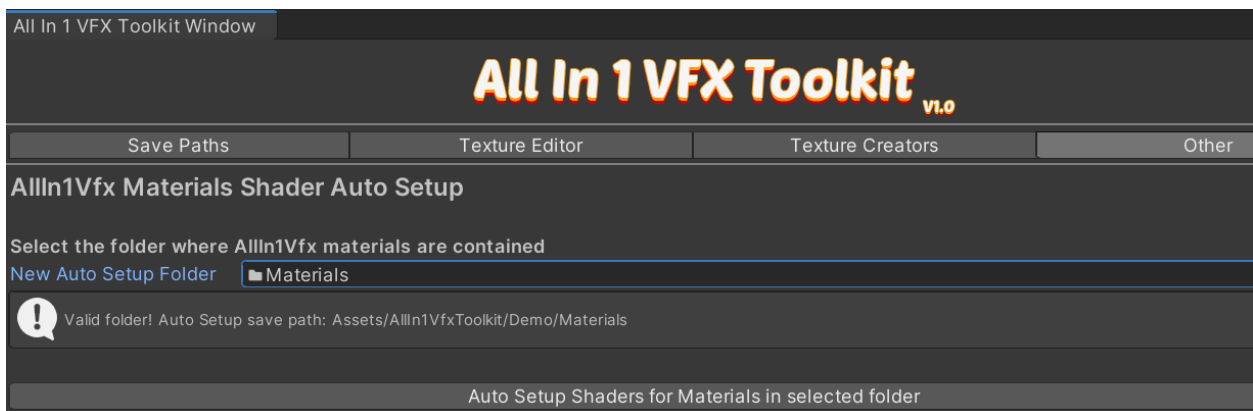
7. At this point you can delete the Built-In shaders (AllIn1VfxBuiltIn and AllIn1VfxGrabPass) to avoid error messages down the line:
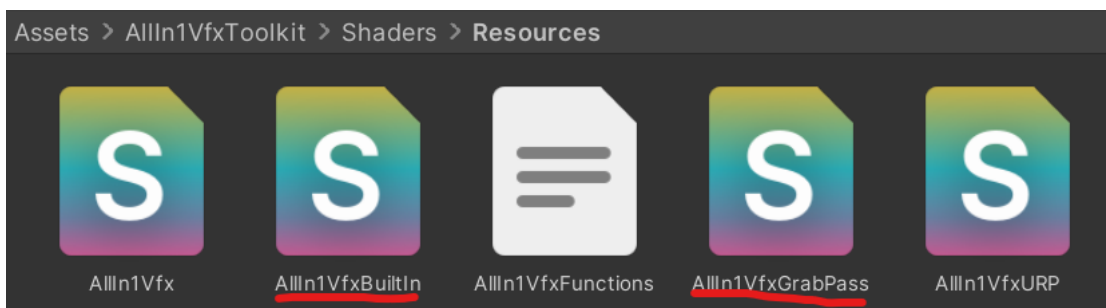
8. At this point you can open the DemoUrp Scene and everything will look like in the promotion material. In case it doesn't look exactly the same it means that you are in a more recent URP version where the Post Processing Bloom implementation has been changed. You can play around with the Bloom values in the Camera gameobject, inside

the Volumes Component.
You can also watch this video:
**https://youtu.be/SjdIMLoNOSw (URP Renderer Setup)**

HDRP Pipeline:
The asset works just as well in this pipeline as in the other 2 (it has complete feature parity and also has amazing performance), but the Demo scene and demo prefabs weren't designed to look good in this pipeline. This means that no matter what we do we won't be able to get the exact results of the store images, WebGL demo and trailer without modifying the properties of each material if the Demo.
Effects created for HDRP should be created inside HDRP and targeting HDRP. If you do so you'll get amazing top notch results. But the demo, unfortunately doesn't include any HDRP specific effect since the effects were created in the Built-In also having URP in mind. This means that the demo will look pretty bad and dull in comparison to what's shown in the promotion materials. Knowing this. , these are the setup steps:
1. Make sure that HDRP is properly installed on the project by either creating a HDRP project in Unity Hub or by going into the Package Manager, installing High Definition RP, creating a Pipeline Asset and finally assigning it to the Graphics Settings tabs
2. Import the "HdrpPackage" Unity Package included in the asset root folder. Note that there are 2 of them "HdrpPackage(pre2020)" and "HdrpPackage(post2020)". Choose the appropriate one according to the Unity Editor version you are using.

With these 2 steps you can already start using the asset. But if you really want to get the Demo working, even knowing and understanding that it will look bad and dull you can can do so following these extra steps:
1. Convert Demo environment Materials to UHDRP by going to Edit, Render Pipeline, Upgrade Project Materials to High Definition Materials
2. Convert all Demo Prefab Materials to HDRP by opening the asset Window (Window, AllIn1VfxToolkitWindow), selecting the Others Tab and pressing Auto Setup Shaders for Materials in selected folder (by default the selected folder is the folder containing all prefab materials):
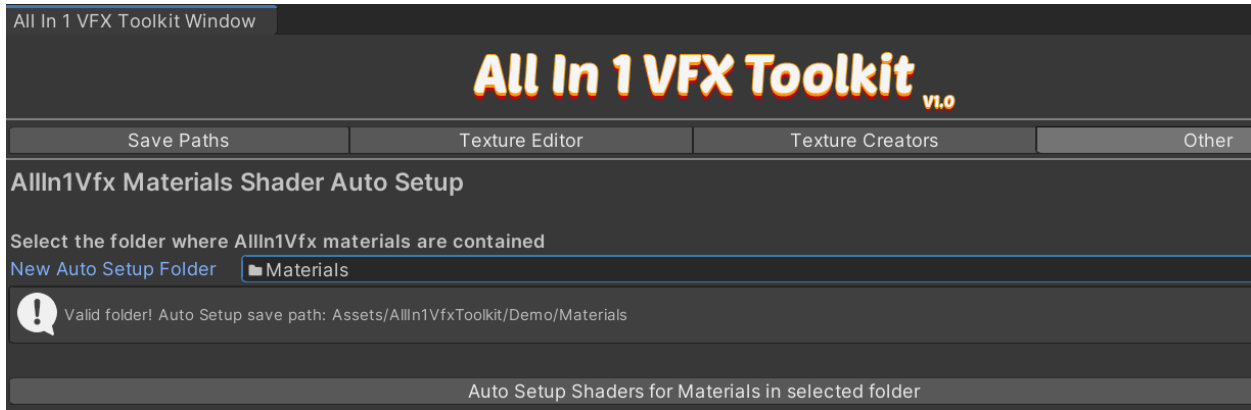
3. Change the Default Volume Profile Asset to AllIn1PostProcessingHDRP:
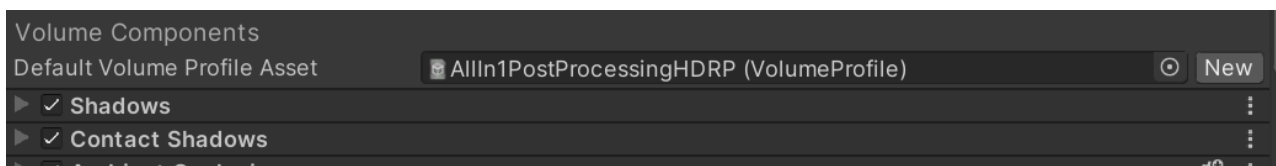
4. At this point you can delete the Built-In shaders (AllIn1VfxBuiltIn and AllIn1VfxGrabPass) to avoid error messages down the line:
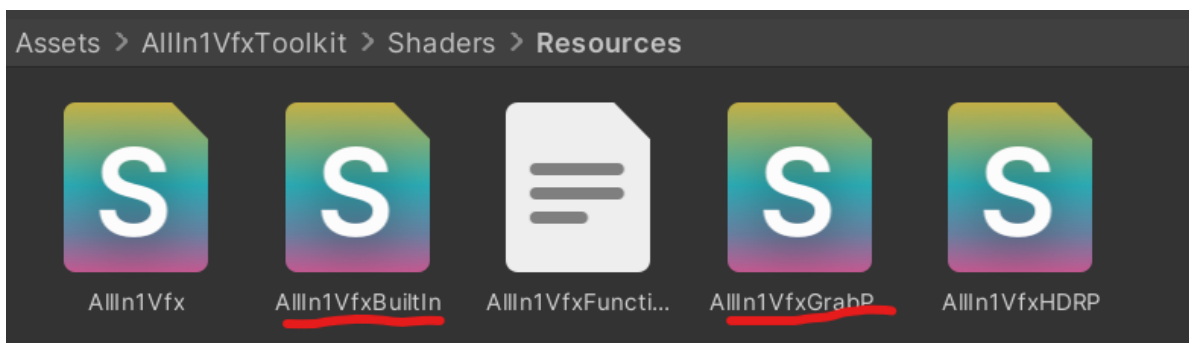
You can also watch this video:
https://youtu.be/I_JvyLNrjwY (HDRP Renderer Setup)

The asset includes a component that will do all the setup for you. The component is called "AddAllIn1Vfx":

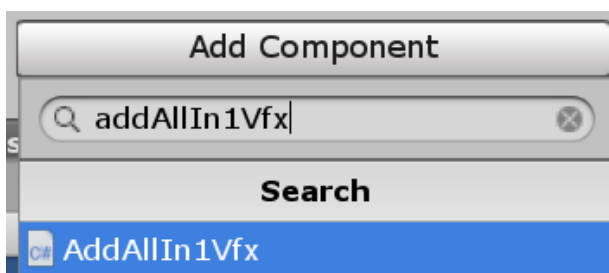When you add it, the component will swap the current material for a new instance of the AllIn1Vfx material of the current render pipeline you are using. The component also has some features that are overviewed in the next section.

This will of course only work when the gameobject has a Renderer component such as a Mesh Renderer, Particle System, Sprite Renderer etc…

At this point you'll see that the custom Material Inspector of the asset is available below the components of the object, you'll be able to start using the Material, enabling/disabling effects and experimenting with the visuals.

Here you have a link to a video that gives an overview of the asset in case you prefer a visual explanation:
https://youtu.be/X8tkrm_HjwU

*If you want to import just the core functionality of the asset without any extra stuff just remove or avoid importing the Demo & Assets and the unity packages of URP or HDRP that you don't need. It's recommended to then go into the All In 1 Vfx Window (see Asset Window section) and set valid Save Paths

Asset Component Features

Also explained in the overview video: https://youtu.be/X8tkrm_HjwU

You can add it by pressing Add Component and searching for AddAllIn1Vfx:
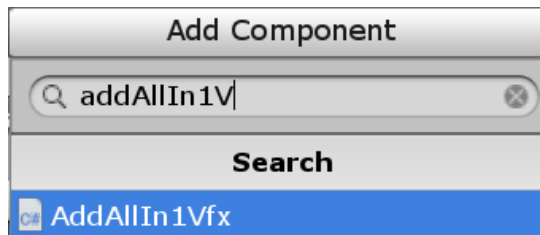
Once added the component will look like this:

The buttons do the following:
- **Deactivate All Effects**: It will deactivate all effects but won't modify any properties. So if you activate an effect again you will obtain your previous visual results.
- **New Clean Material**: It will create a new instance of the AllIn1VfxShader material and assign it to the Renderer.
- **Create New Material With Same Properties**: It will create a new instance of the AllIn1VfxShader material with the same properties of the previous one. This is useful when you want to create a variant of the current material.
- **Save Material to Folder**: Creates a Material asset with the name of the current GameObject and by default saves it in the following path: "Assets/AllIn1VfxToolkit/MaterialSaves" but the path can be changed in the Asset Window. This can be used to assign the same Material to many different Renderers.
- **Apply Material To All Children**: Applies the material of the current selected object to all the objects under its hierarchy.
- **Render Material To Image**: Renders current Texture + Material to an image texture. You can read more about this in the Render Material To Image section.
- **Add Particle System Helper**: Adds the Particle System Helper component. This option is only available when a Particle System is present in the current Gameobject
- **Remove Component:** Removes the component without changing anything else

- **Remove Component and Material**: Removes the component from the GameObject and sets the Sprite Material back to the Sprite/Default one.

Shader Structure and Usage

If you are more of a visual learner there's also a video tutorial:
https://youtu.be/aQ9L1ZE_rHI

The shader is structured in 3 main blocks:

1. **Configuration**: In this section you can decide all render settings of the material.

Changing these settings will be important to get the exact result you are looking for. This is the most technical part of the shader and will require a bit of technical understanding to always select the right configuration for each case.

Hopefully with the help of the presets at the very top you won't need to change any of the Advanced Configuration settings (read next section for more details on those) in most cases. Here's a description and use case of each preset:
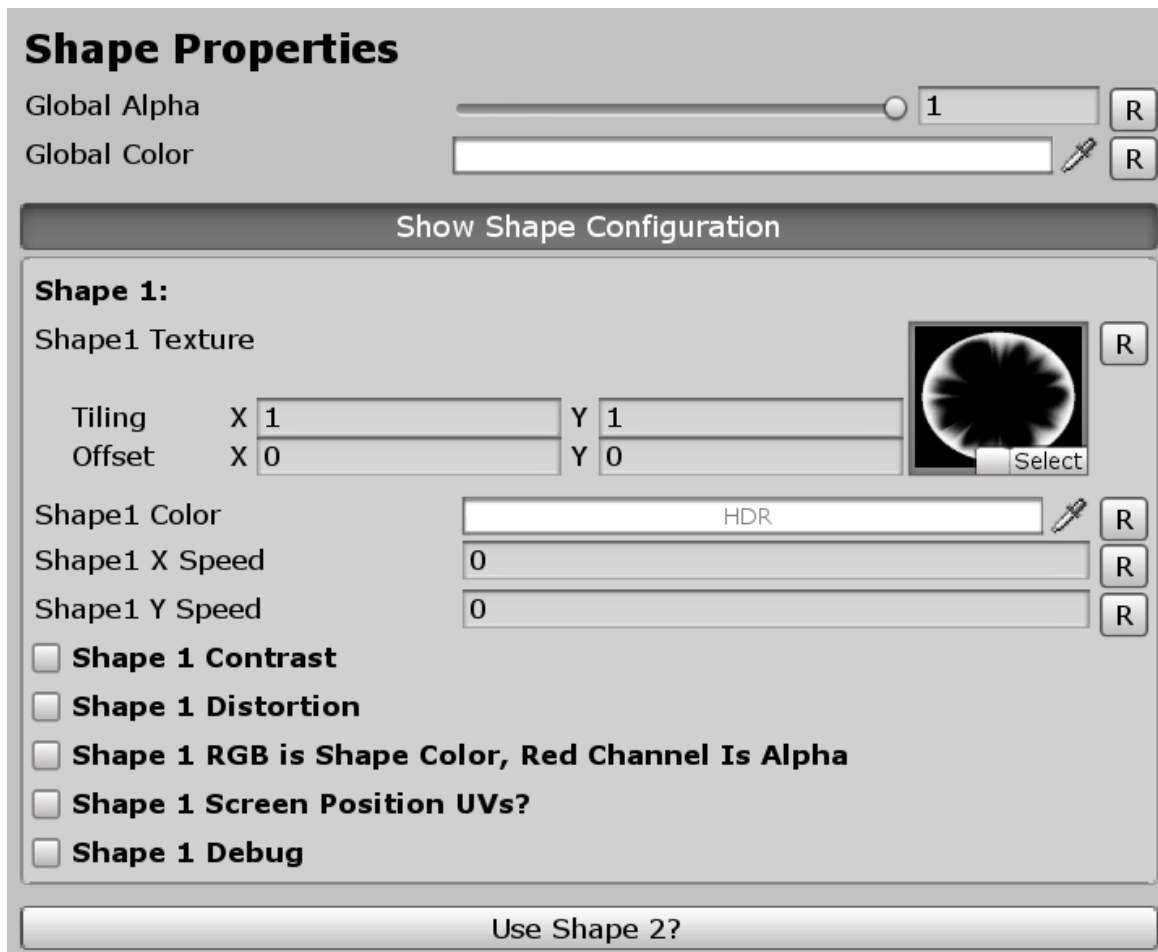
a. **Transparent**: The default preset. Uses regular Alpha Blending and it's great for transparent materials that use a texture with an alpha channel.

b. **Additive**: Additive Alpha Blending, it adds the final result color of the Material to the frame. This means that black will be invisible. So this blending mode is good for textures with no alpha channel+black background and for bright effects. Additive blendings tend to look very bright when the background isn't dark enough.
Keep in mind that Alpha effects when Additive Configuration is toggled will affect the greyscale of the global result instead of the alpha. In a way the shader reads the black color as alpha 0. This allows the shader effects to work in the same way with all presets.

c. **Soft Add (use with caution)**: Similar to the Additive preset but a bit softer and less bright. Here the black will also be invisible. So this blending mode is good for textures with no alpha channel+black background and that aren't very bright effects. You may encounter problems when several bright or glowing materials are overlapping, this is caused by the blending configuration this preset uses. If this is a problem for you please use the Additive preset and lower the alpha to get a similar look.
Keep in mind that Alpha effects when Additive Configuration is toggled will affect the greyscale of the global result instead of the alpha. In a way the shader reads the black color as alpha 0. This allows the shader effects to work in the same way with all presets.

d. **Blend Add / Premultiply**: This preset is a combination between Transparent and Additive taking the best from each configuration. This is the blending used in many games and the one mentioned in this very popular Diablo 3 talk that I 100% recommend watching:
(https://www.gdcvault.com/play/1017660/Technical-Artist-Bootcamp-The-VFX).
The advantage is that it has additive properties such as the black background being ignored but in this case the result won't be as bright and also the Material can be tinted black without fading.

e. **Opaque**: Use this when the Material isn't transparent. This is very performant and straight forward rendering wise but you probably won't use this for VFX.

2. **Shapes**: This is the core of the shader. The shader will calculate a shape result before applying the effects. The shape result will be formed by combining up to 3 different shapes. Each shape can have a different texture, can be scrolled, rotated, distorted and more. Shapes can be enabled and disabled at will, you'll

always have 1 Shape enabled and then you can add 2 extra ones with the button that you'll find on the bottom of the shape block (and that you can see in the following image ["Use Shape 2?"]).
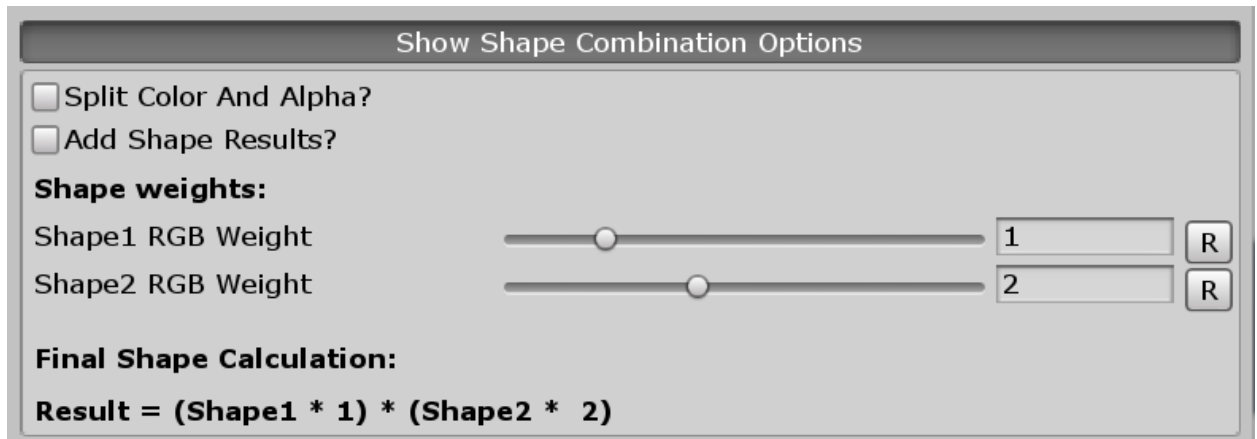

The workflow will consist on individually setting up each Shape you'll need using all it's properties and effects (you can see a full breakdown on the Effects and Properties Breakdown section) and then combine them to fit your needs. To choose how they combine you'll need to have more than 1 shape enabled and this button will appear:

If you press it you will have access to these options:

Show Shape Combination Options

☐ Split Color And Alpha?
☐ Add Shape Results?

Shape weights:

Shape1 RGB Weight ———————○———————— `1` `R`
Shape2 RGB Weight ——————————○—————— `2` `R`

Final Shape Calculation:

Result = (Shape1 * 1) * (Shape2 * 2)

You can see the shape combination operation that is taking place on the bottom in **bold** letters. By default shapes are multiplied together, but you can set the Add Shape Results toggle if you prefer to add them instead. With the weight sliders you can then fine tune how much influence each shape has.

If you want to see all the potential behind this combination options please take a look at this Gdc Diablo 3 talk that I also recommended watching when talking about the Presets above:

https://www.gdcvault.com/play/1017660/Technical-Artist-Bootcamp-The-VFX

3. **Effects**: Effects are applied after the shapes have been combined. The shape result is modified by the effects (to see how all effects work and what properties they have please read Effects and Properties Breakdown). There are 3 kinds of effects:
    a. Color effects: They affect the color (rgb) of the global shape result
    b. Alpha effects: They affect the alpha of the global shape result
    c. Uv and Vertex effects: They affect the texture coordinates of all textures (included shapes) and also there's 1 effect that will displace the target mesh vertices over time

Advanced Configuration and Key Rendering Concepts

If you are more of a visual learner there's also a video tutorial:
https://youtu.be/D7kZRHBUxu0

This is the advanced configuration window:
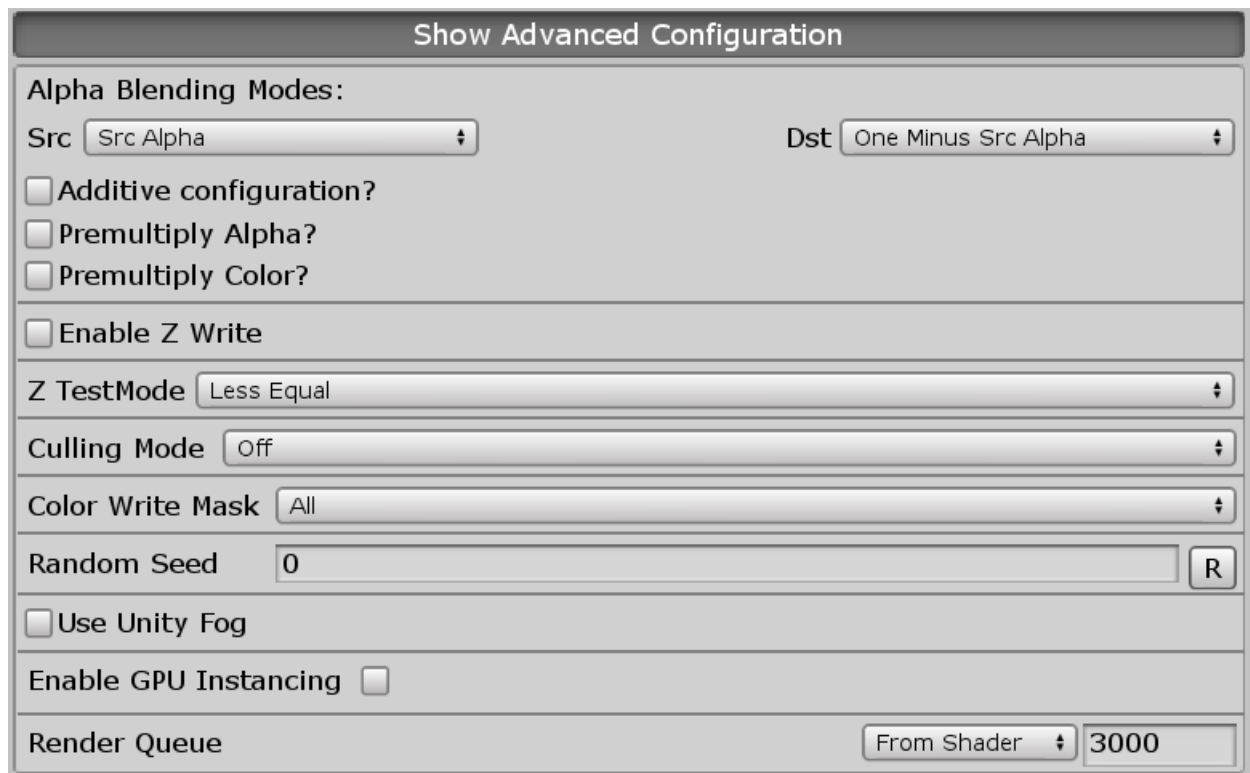
Let's go over each option to see what they do and how they can be used.

Alpha Blending Modes: Sets the blending of the shader output and will determine how the shader result is blended into the final frame. Since the Material will most likely be transparent we need to decide how the transparent parts will be blended with the rest of the frame. You can read more about it here:
https://docs.unity3d.com/Manual/SL-Blend.html

Additive Configuration: Tells the shader this is an additive configuration. This will make the global result of the shader grayscale be considered as alpha. This only makes sense when the Blending Mode is set to an Additive configuration.

Premultiply Alpha: This will multiply the shape result alpha into the color, effectively darkening the parts where the alpha is lower than 1.

Premultiply Color: This will multiply the shape result greyscale into the alpha, effectively making the result invisible where the color is black. The darker the result is the more invisible it will become.

Enable Z Write: When enabled the mesh we are rendering the material onto will write to the Z Buffer, meaning that other objects sorting will be affected by this mesh. This option is useful for meshes that aren't very transparent and to make sure that all faces on meshes that overlap with itself get drawn in the correct order. For example we'll want to use this on a Sphere so that the back faces are rendered after the front faces. You can read more about it here: https://docs.unity3d.com/Manual/SL-ZWrite.html

ZTest Mode: This will dictate how the material interacts with the values of the ZBuffer. We can choose when we want to render this material. By default the Test Mode is LEqual which means that we'll render this material when the current depth value is less or equal to the one in the ZBuffer. This will guarantee the object is occluded by other materials that are closer to the camera. Materials that are in front and write to the ZBuffer.
We can change the ZTest to Always if we want this material to always be visible. You can read more about this here:
https://docs.unity3d.com/Manual/SL-ZTest.html

Culling Mode: In graphics culling means discard and avoid rendering something. In this case culling refers to what faces are discarded based on its orientation. By default it is set to Off, which means that we don't discard any faces, all faces will always be drawn. We can switch to Front or Back and discard the faces that are facing towards or away from the camera. This is great to avoid rendering part of a mesh, for example on a transparent sphere we may want to discard Back faces for a cleaner look. You can read more about it here: https://docs.unity3d.com/es/2018.4/Manual/SL-CullAndDepth.html

Color Write Mask: Allows you to choose what channels the Material will write on. It decides the shader output channels. You'll probably never use this.

Random Seed: A number that will create variations on all texture scrolling, rotations and distortions. This is used to create variations for material instances you are reusing. You can break the repeating visual pattern with this. You can change this via script or through particle system custom data (read the Random Seed section of the documentation to learn how to do so).

Use Unity Fog: Makes the material be affected by Unity's fog of the render pipeline you are using. Note that in HDRP fog is a post processing effect, so this toggle will have no effect.

Enable GPU Instancing: Makes the Material be GPU instanced to save draw calls. For this to work the meshes must be the same in all instances too. You can read more about it here:
https://docs.unity3d.com/Manual/GPUInstancing.html

Render Queue: This will set the order in which materials are rendered. The higher the number the later it will render. This can be very important to guarantee that your materials get rendered in the order that works for your setup. You can choose how transparent objects are sorted in Project Settings, Graphics, Transparency Sort Mode and Axis, but in some cases we need to tweak this Render Queue value too to make sure we always get the render order we need.

Particle System Helper Component

If you are more of a visual learner there's also a video tutorial:
https://youtu.be/0knI_Ee4lBU

The Particle System Helper Component will dramatically accelerate your Particle System workflow by saving clicks and typing. By having everything in the same place and by automatically refreshing the particle system when a property is changed you can get the particle system you are looking for in a matter of seconds. On top of that you can create and load particle system presets to accelerate your workflow even further, create a template of the setups you use more often and reuse them.

To add the component you can press the Add Particle System Helper button on the asset component inspector (it will only work if a Particle System is present in the GameObject):

Or you can directly add it:
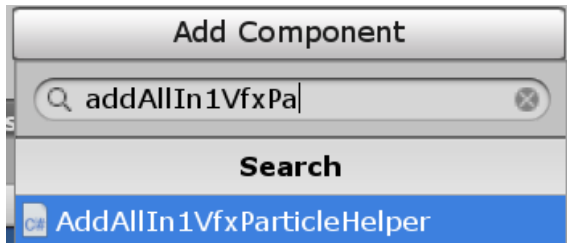
Once added it will look like this:
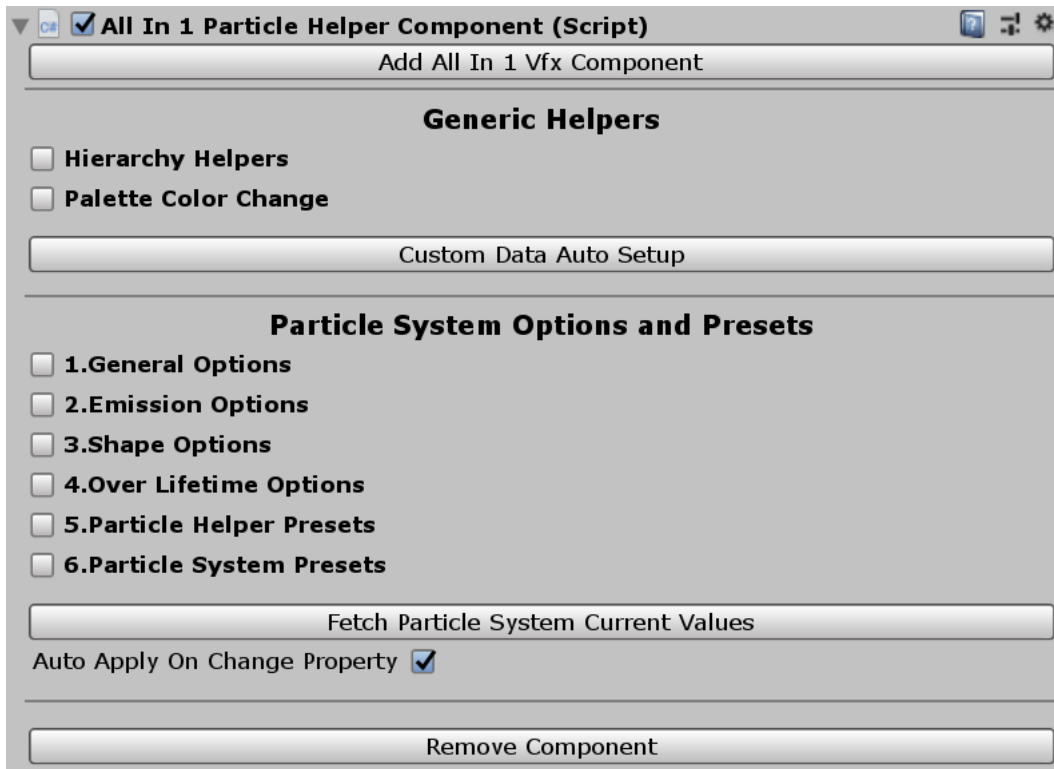
Hierarchy Helpers allow you to create a copy of the current Particle System as a Child or as a Sibling GameObject and will use the new copy number to name the new copy:
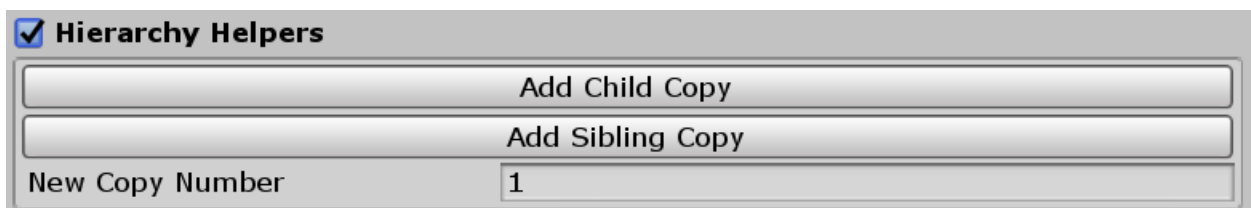
This is great to quickly add more sub particle systems inside a more complex effect.

Palette Color change takes a New Color as input and then will recolor all Colors and Gradients of the Particle System to fit the new color scheme you choose:
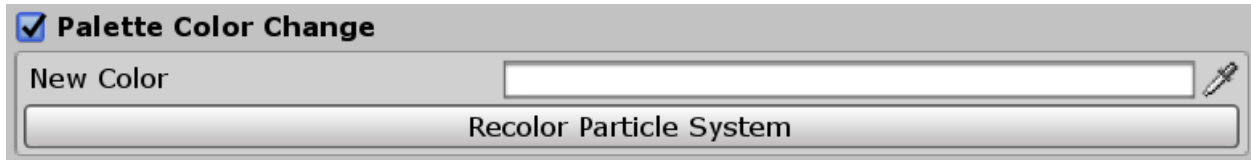
Use this to quickly create color variations of your particle systems.

Pressing the Custom Data Auto Setup will enable the Custom Data section of the Particle System and configure it to set a random timing seed to each particle and also will add the vertex streams you need depending if you have effects that use them or not. The effects that will use these vertex streams in some cases are the 2 different Fade effects and the Texture Offset Custom Stream effect. You can then use the generated curves to tweak the effect to your liking (see Custom Vertex Streams and Custom Data Auto Setup for more details).

1.**General Options** gives you quick access to all main properties of the particle system. This section will save you many clicks when used properly and will speed up your workflow:
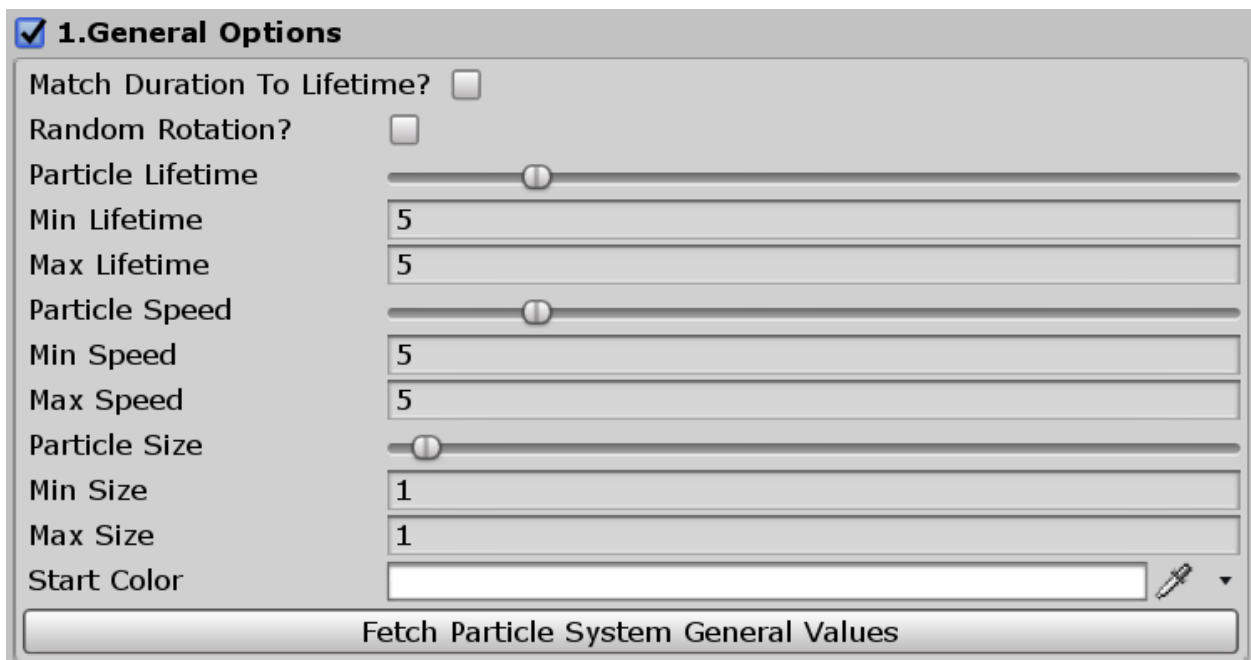
Values will be fetched when you add the component but you can press the Fetch button to fetch the values again if you need to.

2. **Emission Options** allows you to quickly set how the particles will be emitted. You can choose between 1 burst or constant emission rate and easily tweak the values:

17

3. **Shape Options** will change the emission pattern of the particle system. None means that the particles will spawn in the Transform origin.

4. **Over Lifetime Options** create a simple Color or Size Over Liftime gradient either ascendant or descendant, you can use this to quickly prototype the effect and then fine tune the curve to get the exact result you desire.

5. **Particle Helper Presets** refer to a copy of the data held by this component. This option will allow you to save and load the presets of this component so you can reuse it to fit your needs and accelerate your workflow. Note that when you enable this effect the script will look for this data all over your project and this can cause a couple second freeze if your project has many files.

6. **Particle System Presets** is very similar to the previous option but it will save all the configuration of the Particle System, it will save a full copy of it that you can then apply

when you need it. Use this when you want to save some options that this component doesn't support. Note that when you enable this effect the script will look for this data all over your project and this can cause a couple second freeze if your project has many files.
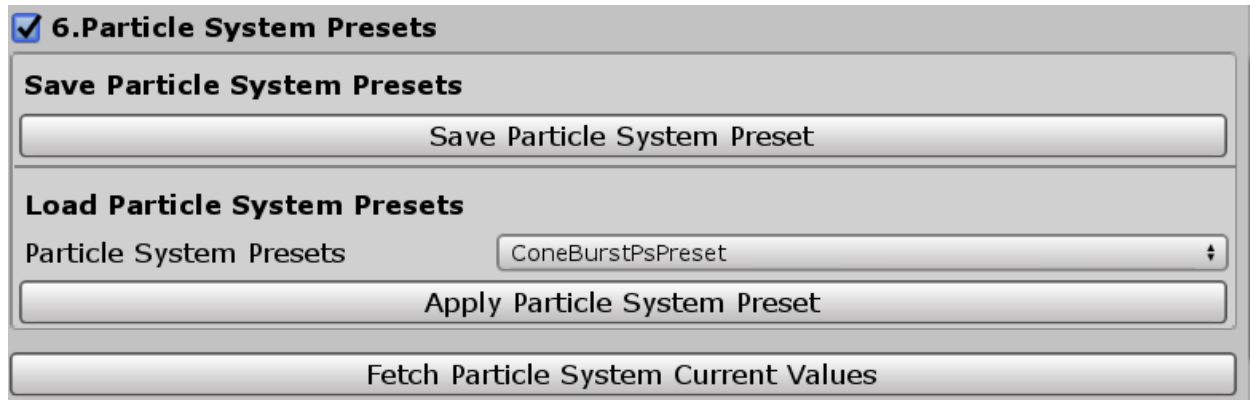
At the very bottom of the component you'll find the Auto Apply On Change Property, when this is enabled any property change will cause the Particle System to update. If you want to avoid this and manually apply the changes instead disable the toggle and use the Apply button instead:
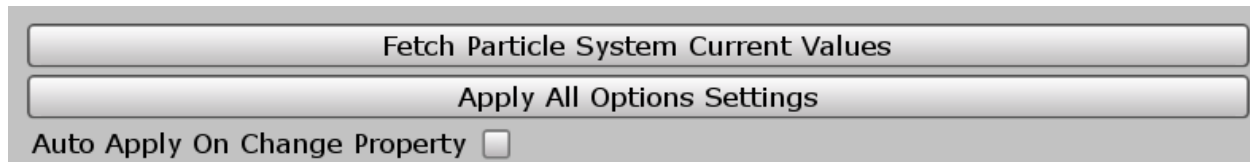
Asset Window

If you are more of a visual learner there's also a video tutorial:
https://youtu.be/jKA4xc0hA6k

You can access it by going to Window -> AllIn1VfxToolkitWindow.
The Asset Window offers you a bunch of settings options and utilities divided in 3 sections:

1. **Save Paths**: Used to set the different save paths of Materials, Presets and Textures the asset generates:
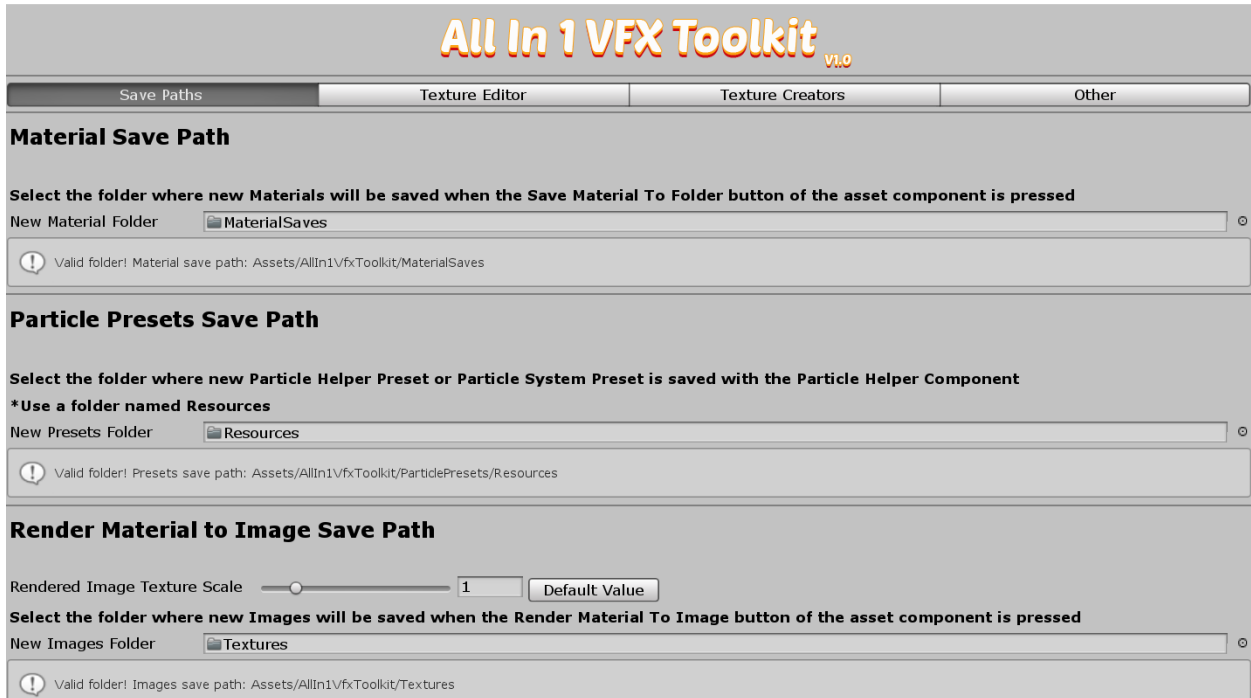
Material Save Path: Folder where new materials created with the "Save to Folder" button on the asset component will be saved to. See Asset Component Features section for more info.

Particle Presets Save Path: Route where the Particle System Presets will be saved to. See Particle System Helper Component section for more info.

Render Material to Image Save Path: Path for the Textures created with the "Render Material To Image" button on the asset component will be saved to. See Asset Component Features section for more info.

2. **Texture Editor**: Used to edit existing Textures. It can be used to quickly tweak the look and to quickly rotate and flip them.

To use it drag or assign a texture in the Image to Edit slot and then use the properties and buttons to change it to your liking:

Save Paths	Texture Editor	Texture Creators	Other

Image to Edit

Select

Brightness — 0 R
Contrast — 1 R
Gamma — 1 R
Exposure — 0 R
Saturation — 1 R
Hue — 0 R

Invert ☐   Greyscale ☐
Fully white ☐   Black background ☐
Alpha is greyscale ☐

Rotate Left 90°	Rotate Right 90°
Flip Horizontal	Flip Vertical

Press to show Original Image

*Preview is locked to 256px maximum (bigger textures are scaled down), but the image will be saved to its full resolution

Export Scale — 1 R

Current export size is: 512 x 512 (size snaps to the closest power of 2)

Save Resulting Image as PNG file

Finally press the Save Editor Resulting Image as PNG file to save the edited image into the folder of your liking, by default it will suggest you save it in the same folder of the original image with a different name, but you can change the save route and name to fit your needs.

3. **Texture Creators**: Here you can create Normal Map textures for the Screen Distortion effect (or for any other use really), Gradient Textures either for the Color Ramp effect or for regular greyscale gradients that are very often used as masks for other effects or shape textures, atlas textures to randomize your Particle Systems and Tileable noise textures to use on your effects.

Normal/Distortion Map Creator

Select the folder where new Normal Maps will be saved when the Create Normal Map button of the asset component is pressed (URP only)

*These Normal Maps can then be used with the Screen Distortion effect

New Normal Maps Folder 📁 DistortionNormalMaps

⚠ Valid folder! Normal Maps save path: Assets/AllIn1VfxToolkit/Textures/DistortionNormalMaps

Assign a sprite you want to create a normal map from. Choose the normal map settings and press the 'Create And Save Normal Map' butto

Target Image
None
(Texture 2D)
Select

Normal Strength: ─────○──────────── 5
Normal Smoothing: ──────────○─────── 1

Add a Target Image to use this feature

*This process will freeze the editor for some seconds, larger images will take longer

Color Gradient Creator

This feature can be used to create textures for the Color Ramp Effect

Color Gradient:
Texture Size: 128 ▾

Normal/Distortion Map Creator: Makes a Normal Map from a Target Image. Tweak the Strength property to make the normal map more pronounced and the Smoothing to blur the result. See Screen Distortion section for more info.

To use the **Color Gradient Editor** just edit the Color Gradient, choose a texture size (usually it can be very small if the filtering is set to Bilinear), choose the texture filtering and press the Save button. This can be used to create textures for the Color Ramp effect or black and white masks that can then be used in your effects.

The **Texture Atlas / Spritesheet Packer** can be used to pack multiple images into a single one. This is very useful to create variation on a Particle System for example. We can use a 4x4 texture for example and then have the System choose a random one each time it plays in the Texture Sheet Animation tab.
To use this tool, add the desired textures to the Atlas array. Choose the amount of Columns and Rows (Note that you'll need to make sure that there are enough slots to fit all your Atlas textures. For example if you have 2 rows and 2 columns you can fit up to 4 textures, so if the Atlas array has 3 elements one slot will be empty and if the Atlas array has more that 4 elements only 4 will show on the final result). Then choose the Atlas size and filtering and Save.

The **Tileable Noise Creator**, as the name says, allows you to preview, edit and save

tileable noise textures to use on your effects out of 8 different noise types. Having tileable noise that you can create and edit in a few clicks is a massive time saver. Not even Photoshop has a quick way of doing this. Here you can do it inside the Unity editor, with just a few clicks.

To use it select a Noise Type and once you move any slider you'll see the noise preview, play around with the properties and when you are happy choose the size, filtering and save.

4. **Others**: Here all other options will be placed, but for now here is where the AllIn1Vfx Material Shader Auto Setup can be done. First select the folder that contains the Materials you want to auto setup and then press the button to automatically select the correct shader variant the Material needs depending on Render Pipeline and the effects that are being used.

The asset has a few shader variants, 2 for Built-In Render Pipeline, 1 for URP, 1 for HDRP and 1 variant that contains the effects that will work out of the box across all 3 Pipelines. These shader variants will automatically be swapped and maintained by the asset during regular use. But this feature is needed to properly configure Materials that were created in some other Render Pipeline and that were imported into the project. A clear use case of this is converting the Demo Materials to URP or HDRP as described in the First Steps section, in those cases we want to convert the Built-In materials to SRP ones.

Textures Setup

Most of the time it will be a good idea to set the Wrap Mode of the Import Settings of the textures you use to Repeat. This will assure a proper result when using scrolling textures.


23
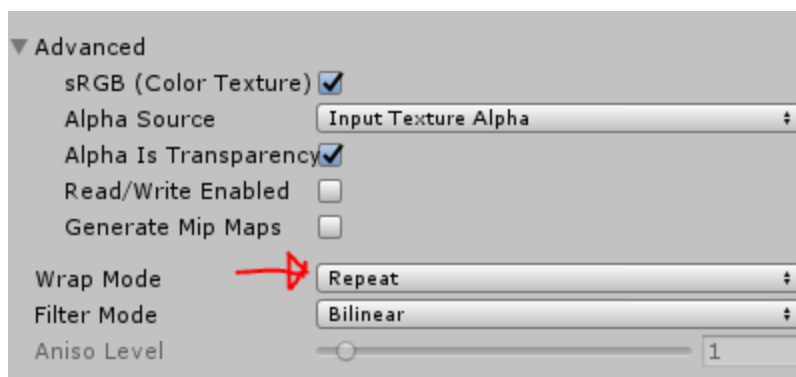
Saving Prefabs

By default this asset doesn't save the Material you are using, instead it keeps it as part of the Scene in order to avoid having too many objects cluttering your project. This means that by default, when you turn a GameObject with an AllIn1VfxShader material into a prefab, the prefab won't render correctly since it doesn't have a reference to the Material inside the Project Asset files.

In order to save a Prefab you first need to save its Material. You can do so with the "Save Material to Folder" button that you'll find on the asset component:
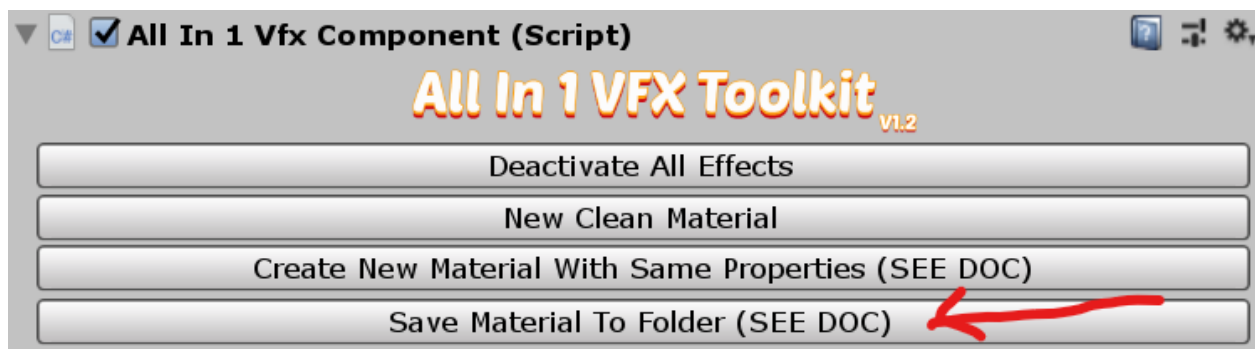

Screen Distortion and Creating Distortion Maps

If you are more of a visual learner there's also a video tutorial:
https://youtu.be/jKA4xc0hA6k?t=214

Screen Distortion is an effect that will distort the final frame based on a Distortion Map Texture. Note that this effect is potentially the most performance intensive effect in the asset, specially in the Built-In render pipeline. Also note that in Built-In Render Pipeline

Here's an example of the effect:
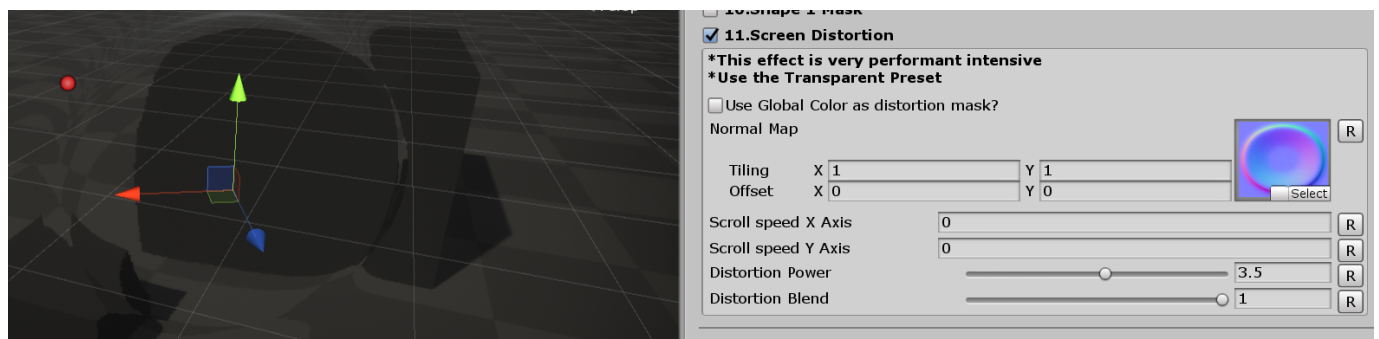
In the left you can see a test setup that is distorted, it has a fish eye shape in the middle. This distortion is set by the Normal Map on the right.

This Normal map can be created outside of Unity or you can create it inside Unity with the All In 1 Vfx Window Texture Creator (see Asset Window section for more details). For example we can create a fisheye Distortion Map like this:
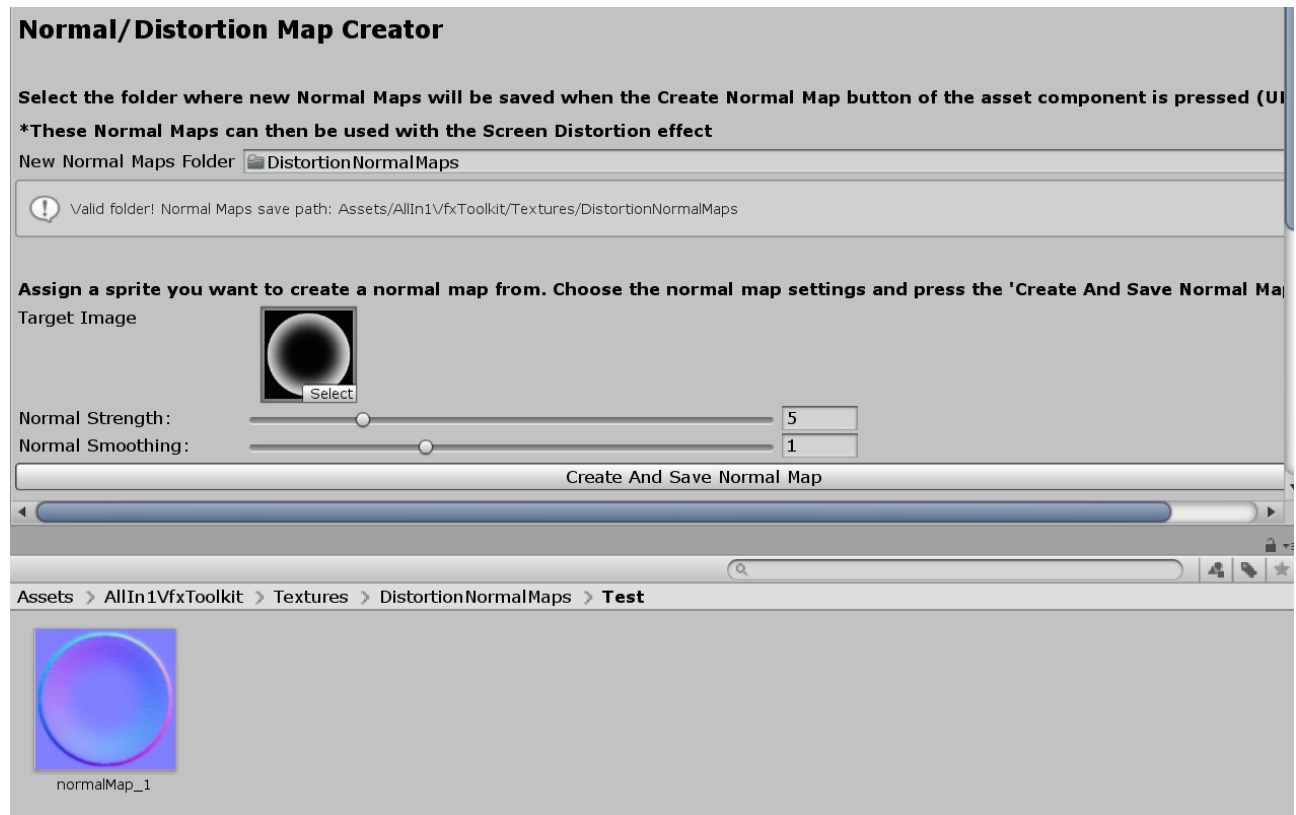
Custom Vertex Streams and Custom Data Auto Setup

If you are more of a visual learner there's also a video tutorial:
https://youtu.be/-xVtAoS_s7k

Some effects can use the Particle System Custom Data to control them. Those effects are both Fades, Texture Offset Custom Stream and Shape Weights Custom Stream. By using the Custom Data curves we have fine control over how much we want the particles to fade over time or scroll over time.

Let's first see how to setup the effects and then we'll take a look at how to use the Custom Data.

To use this features on Fade From Noise Texture or Fade From Final Shape effects we'll need to check the Fade Amount Driven By Vertex Stream Toggle :
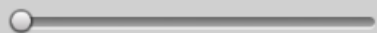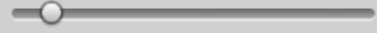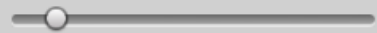
As the toggle says when this is set to true the Custom Data of the Particle System will be responsible of changing the Fade Amount over time. If the toggle is set to false the alpha of the effect will drive the Fade Amount.

The Procedural Dissolve works just the same:

The Texture Offset effect will only be visible on Materials that are on a Particle System and will only show the shapes that are enabled, in this case all 3 shapes are enabled:

We'll then choose how much each shape gets affected by changing the multiplier property of each shape. But on the Custom Data we'll have only 1 property to control all 3 shapes, the only way of having the offset be different for each shape is by changing the multiplier properties.

The Shape Weights Custom Stream is similar to the Texture Offset one that we just saw. It will only appear in the Material Inspector when used on Particle Systems and also will only show Shapes that are enabled:

Negative values will make the corresponding Shape less visible while positive values will make the corresponding Shape. This can be used to fade partial parts of the particles in and out.

To set the Custom Data that you need for any particular Material you can use the Particle System Component and press the "Custom Data Auto Setup" button to automatically add the needed vertex stream channels:
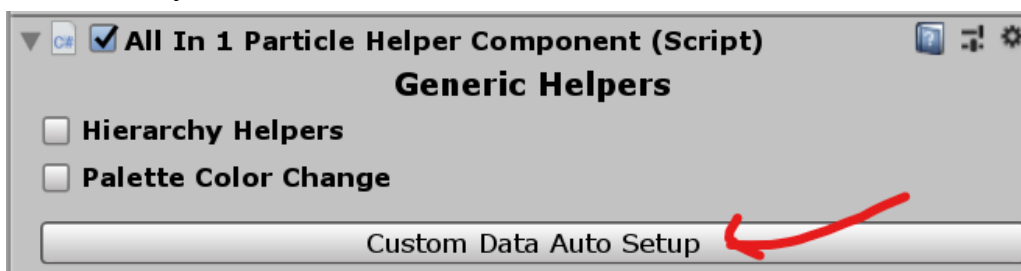
After pressing the button the Custom Data setup of the Particle System will look like this (if a row is missing it means that the effect that uses it is disabled):
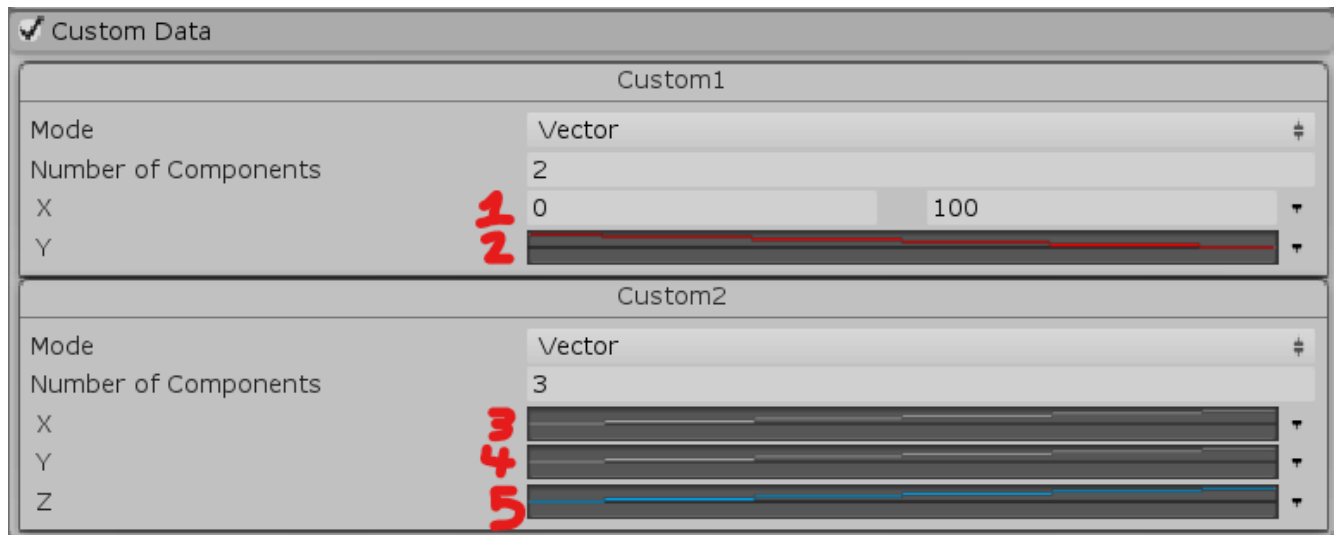
The row numbered with a red number 1 is the random timing seed explained in the Random Seed section.

The row numbered with a red number 2 is the Fade Amount of both Fade effects (it's shared, but you'll be using one or the other, not both at the same time).

The row numbered with a red number 3 and 4 are the Texture Offset amount in the X and Y axis respectively. This effect is very useful to scroll a particle texture in a very controlled way, for example for a sword slash. Remember to use the Texture Offset mult properties to choose how much each shape gets affected by these values.

Finally, row 5 corresponds to the Shape Weight Offset. This value will be added to the Shape Weights that you can find in the Shape Result part of the Material Inspector (it will only appear when 2 or more Shapes are present):

Shape Result:

Show Shape Combination Options

☐ Split Color And Alpha?
☑ Add Shape Results?

Shape weights:

Shape1 RGB Weight	○————	0.33	R
Shape2 RGB Weight	○————	0.33	R
Shape3 RGB Weight	——○——	0.9	R

Final Shape Calculation:

RGB = ((Sh1.rgb * 0,33) + (Sh2.rgb * 0,33)) + (Sh3.rgb * 0,9)

A = max(max(Sh1.a * 0,33 , Sh2.a * 0,33), Sh3.a * 0,9)
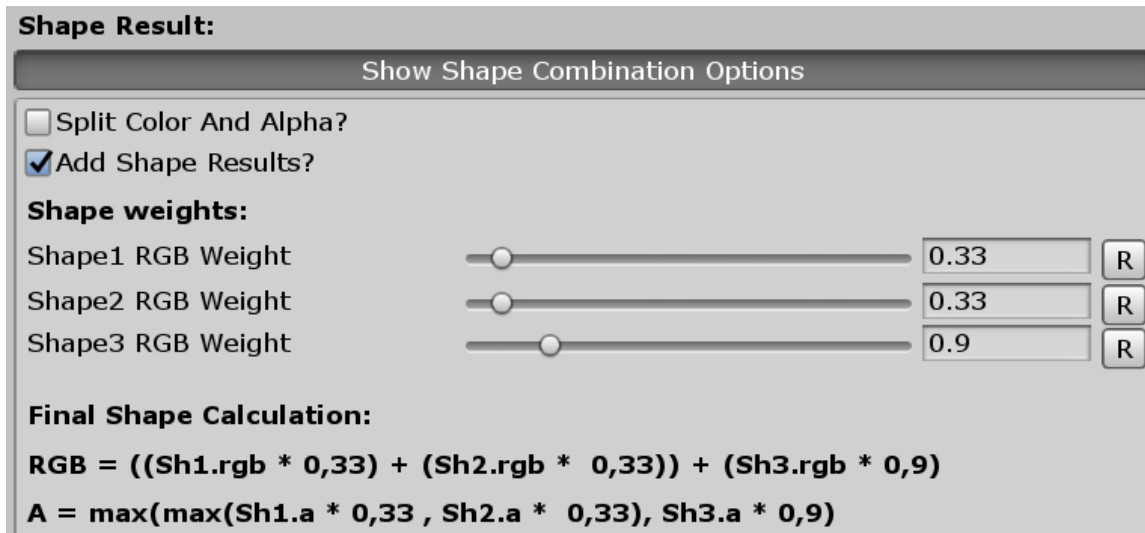
Use the Shape Weight Offset properties to choose how much each Shape will get affected by the values on row 5.

How to Animate Materials

If you are more of a visual learner there's also a video tutorial:
https://youtu.be/zZX1dbJTsPg

The custom material inspector properties can be animated through the Animation window as any other Unity component.

Please keep in mind that UI material properties can't be animated using the Animator, the reason being that Unity won't allow you to animate shared material properties. Unity UI Images materials are always shared, which means that all Images use the exact same material instance of a particular Image and therefore if a property is changed for one Image material all the other Images that share material will change too. Since Unity won't allow this behaviour it doesn't support using the Animator in UI Material properties. And unfortunately I can't do anything about it.

I recommend using an amazing free asset in the store called DoTween to animate the UI material properties through code or if you prefer you can use the function calls described in the following section.

Also consider that since UI material instances are shared you may want to create a copy of each material through script on an Awake method

(AllIn1GraphicMaterialDuplicate.cs):
private void Awake()

{

    Graphic graphic = GetComponent<Graphic>();

    graphic.material = new Material(graphic.material);

}

Scripting

If you prefer avoiding animations or want to change properties through code you also have the possibility. This will be mostly used on Materials assigned to Mesh Renderers, it won't make much sense to use this on Particle Systems.

You can find the property names by hovering the mouse over any property in the Material Inspector:
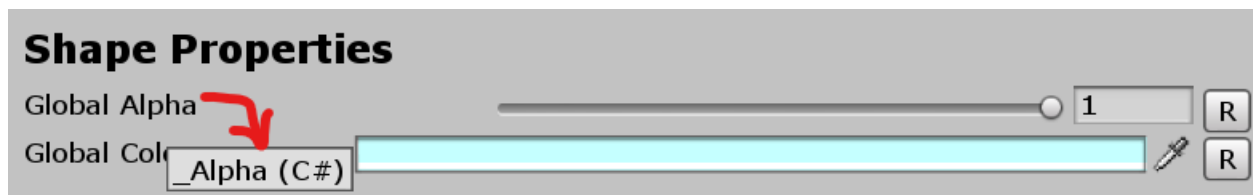
To do so you'll need to use the following Unity functions:
- Material.SetFloat:
  https://docs.unity3d.com/ScriptReference/Material.SetFloat.html
- Material.SetColor:
  https://docs.unity3d.com/ScriptReference/Material.SetColor.html
- Material.SetTexture:
  https://docs.unity3d.com/ScriptReference/Material.SetTexture.html

You can find all property names on:
AllIn1VfxToolkit\Shaders\Resources\AllIn1Vfx.shader
All properties are located from line 5 to 185 and can also be found at the Effects and Properties Breakdown section.

Here an example code snippet:
Material mat = GetComponent<Renderer>().material;
mat.SetFloat("_Alpha", 1f);
mat.SetColor("_Color", new Color(0.5f, 1f, 0f, 1f));
mat.SetTexture("_MainTex", texture);

*Note that there is an important distinction to be made between a "material" and a "sharedMaterial" of a Renderer. You shall use "material" if you only want to change a property of that instance of the material. And "sharedMaterial" if you want to change the property of all the instances of that material

Visual Effect Graph (Vfx Graph)

The new Vfx Graph package (currently still in an experimental phase and only available as a package in the Package Manager) currently doesn't support custom shaders, just Shader Graph shaders. If custom shaders support eventually get supported I'll make an update to show how to use the feature as soon as possible.

How to Enable/Disable Effects at Runtime

There are 2 ways of achieving this:

1. All effects have a property value combination that makes them look deactivated (usually by reducing the amount or blend property to 0, but it may vary depending on the effect). So the most clean way of deactivating and activating effects is by enabling all the effects you'll use and then dynamically changing the property values either by animating the properties or by modifying the values by script as seen in the Scripting section.

2. This other way is less efficient, messier and will cause materials to become invisible in the final build if you set a combination of effects that isn't included in some other material in your project. **So be warned, use this with caution (in fact you should probably avoid this option and use option 1 instead)** and test it on the target platform. If material effects disappear or you get a graphics error (object turns pink) at some point make sure to have some material in your project that includes the same set of effects than the material that isn't showing.
   This method consists on enabling and disabling the shader compilation flags at runtime, so Unity will compile and replace the shader at runtime (on a final build shaders can't be compiled, so a shader variant with the new keywords will need to be available to avoid the errors mentioned). If you are sure to have a shader variant for the resulting toggle combination you can use the Enable/Disable Keyword method like so:

Material mat = GetComponent<Renderer>().material;

…

mat.EnableKeyword("GLOW_ON");

mat.DisableKeyword("GLOW_ON");

(Keyword names of every effect can be found at the <u>Effects and Properties Breakdown</u> section)

If you really want to use this feature and you really know what you are doing and how to prevent the aforementioned errors you can also find the effect name by hovering it with the mouse in the Material Inspector:
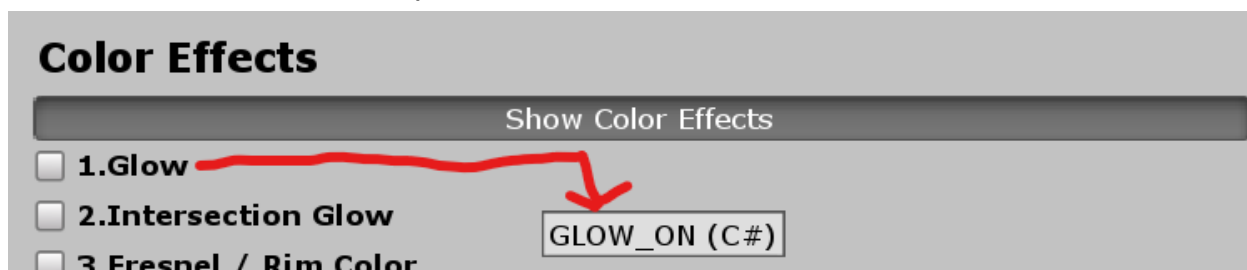
Random Seed

The asset shader has a random seed property (_TimingSeed). This property will give some variation to the material. Different seed values will cause different scrolling, rotations and distortions on the shader, therefore causing a visual distinction between the same materials with different seeds. This can help avoid repetition in between particles using the same Material or meshes using the same Material.

For Particle Systems you can add the Particle System Helper Component (see it's section of the documentation) and press the Custom Data Auto Setup button. The 0 to 100 X axis property of the Custom Data section of the Particle System is the random timing seed:
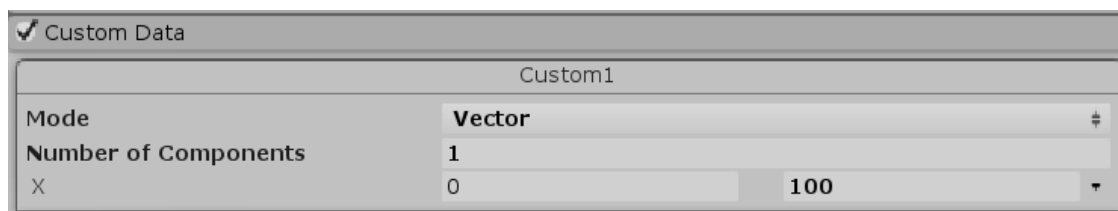
For Meshes you can just add the All1VfxRandomTimeSeed script to the GameObject and the Random seed will be assigned on start. Note that the shader is prepared to GPU instance this property, so you could enable GPU Instancing in the Material Inspector Advanced Configuration and the instancing won't break even if you use this script on all mesh instances.

Render Material To Image
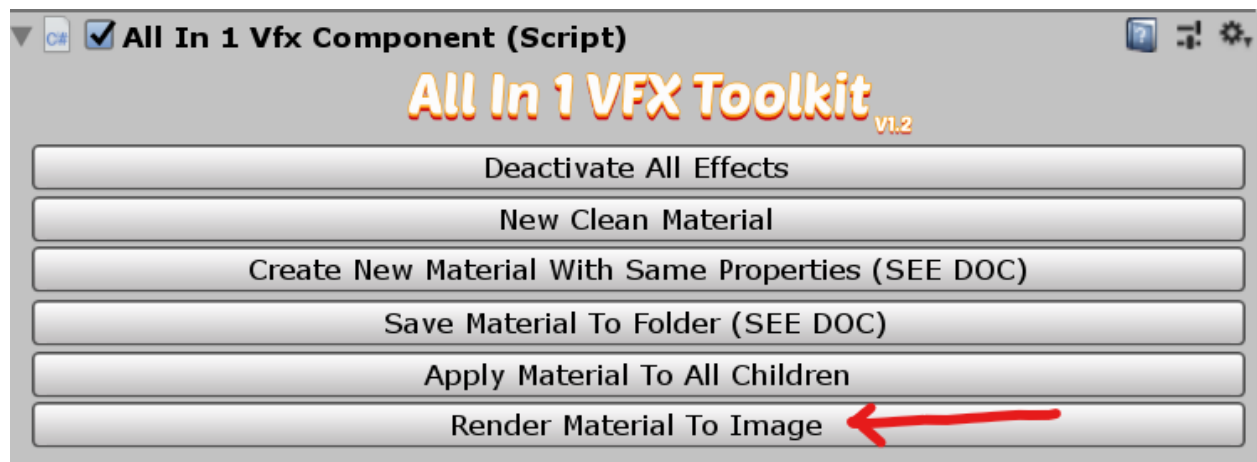
If you press the Render Material To Image the current Texture + Material of the current Gameobject will get rendered into a texture that you'll then be able to save wherever you want:
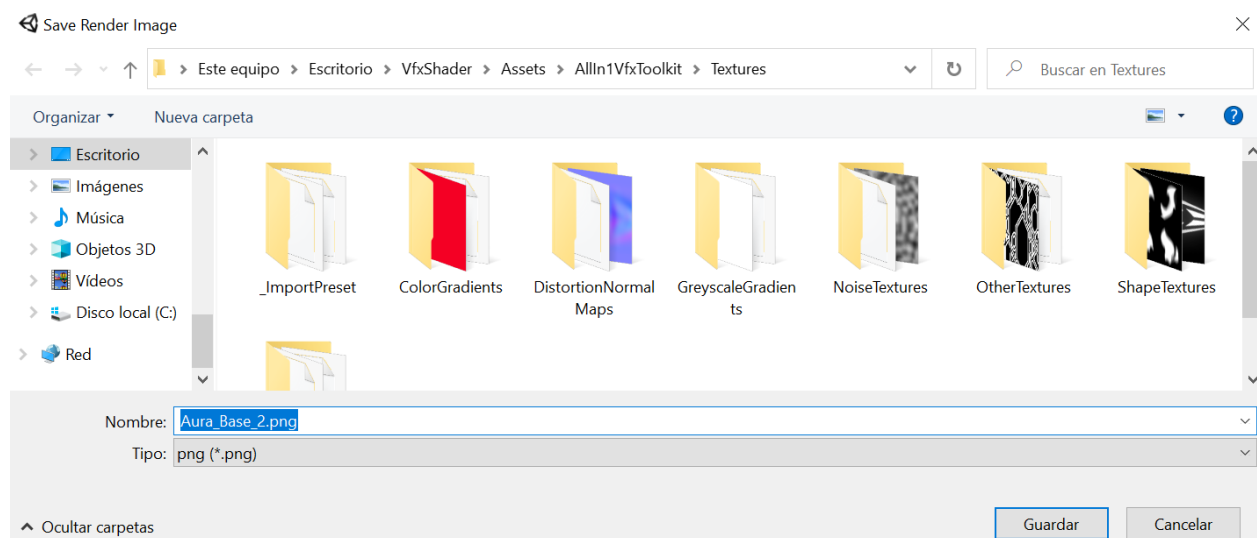
In the asset window you can change the default save route and the Rendered Image Texture Scale. Since the output will look slightly less sharp than the in-engine version you can upscale the output to make it more crisp in case you need to.

This feature is useful to offload some work from the gpu or to create texture variations. By baking the result into a texture it means that the gpu won't need to compute all effects every single frame. In any case please keep in mind that the asset is made with efficiency in mind, even in low end devices. This is just one more tool at your disposal.

Please don't try to optimize early by swapping out all Materials by a rendered image. Only use this for performance reasons after running into performance problems (you most likely never will, even in low end devices).

You can also use this feature to pre-render a certain image for some VFX or to pre-render a texture that you can then modify with the asset shader. This feature allows you to stack and re-apply the asset effects recursively as many times as you need.

Premade Textures, Meshes and Materials

The asset includes many textures, meshes and materials that you can use on your project and that will allow you to prototype effects and even create full awesome professional effects without creating your own assets and saving time on the process.

Textures Route: Assets\AllIn1VfxToolkit\Demo & Assets\Textures
Meshes Route: Assets\AllIn1VfxToolkitDemo & Assets\\Meshes
Materials Route: Assets\AllIn1VfxToolkit\Demo & Assets\Demo\Materials

There's also many pre-made prefabs of complete effects on the Demo scene that you can also use on your projects.

Helper Scripts and Other Utilities

The asset includes some extra scripts and utilities that can come in handy in some cases. Here's a list of all of them:

1. **Particle System Helper (AllIn1ParticleHelperComponent.cs)**: Covered in the Particle System Helper Component section.

2. **Look At (AllIn1LookAt.cs)**: Used to make a GameObject face in a particular direction. You can choose if you want the facing direction to update every frame or only on start. You can choose a Transform to be the target or target the Main Camera. Finally you can choose what axis of the transform should face the target, by default the Forward vector will be the one to face the target. But you can choose whatever axis you prefer.

3. **Bounce Animation (AllIn1VfxBounceAnimation.cs)**: It will animate the transform position back and forth over time. The starting position will be the origin point and then you can choose a Target Offset and a speed to tweak the animation.

4. **Auto Rotate (AllIn1AutoRotate.cs)**: Rotates the transform over time around the axis of your choosing.

5. **Auto Destroy (AllIn1VfxAutoDestroy.cs)**: The GameObject will get destroyed in N seconds after getting instantiated. This can be used when instantiating certain meshes that need to be destroyed after some time for example. If you want to clean up Particle Systems, set the Stop Action to Destroy instead (it's a better practice since it's cleaner and more performant).

6. **Scroll Shader Property (AllIn1VfxScrollShaderProperty.cs)**: Takes the name (as a string, see Effects and Properties Breakdown to see the properties names) of the shader of the current Material used in the Renderer of the current GameObject or the Material we pass in as a parameter and increases or decreases its value over time. If needed it can apply a modulo operator, this is useful if we always want to keep the property in a certain range, like 0-360 for example. A new back and forth toggle has been added too, this will ensure that the property goes from the initial value to the max value back and forth, use the Scroll Speed property to choose the speed of the back and forth.

7. **Screen Shaker and DoShake (AllIn1Shaker.cs and AllIn1DoShake)**: These 2 components combined can shake any GameObject Transform, but it was created as a simple but effective camera shaker that is currently used in the Demo. You can take a look at the Demo scene to see how it's setup but the idea is to have an empty parent object for the camera and then have a child object with the actual Camera component in the 0,0,0 position. This will allow the camera to shake correctly regardless of the position and rotation of the parent object.

The DoShake component can be added to any effect that needs to shake the camera when instantiated. This component will call AllIn1Shaker.i.DoCameraShake(shakeAmount); on Start. Since the Shaker component follows a singleton pattern you can use the above function call to activate a shake wherever needed.

8. **Scale Tween (AllIn1DemoScaleTween.cs)**: This component will create a Scale Down, Scale Up procedural code animation when the function ScaleUpTween() or ScaleDownTween() is called. This is used in the Demo buttons and can be handy to polish UI and gameplay interactions. There are many ways to do this, but I decided to document in case it is useful for someone.

Effects and Properties Breakdown

The AllIn1Vfx shader has a custom Material Inspector that allows you to activate and deactivate effects. When an effect is activated it displays it's properties so that they can be modified:
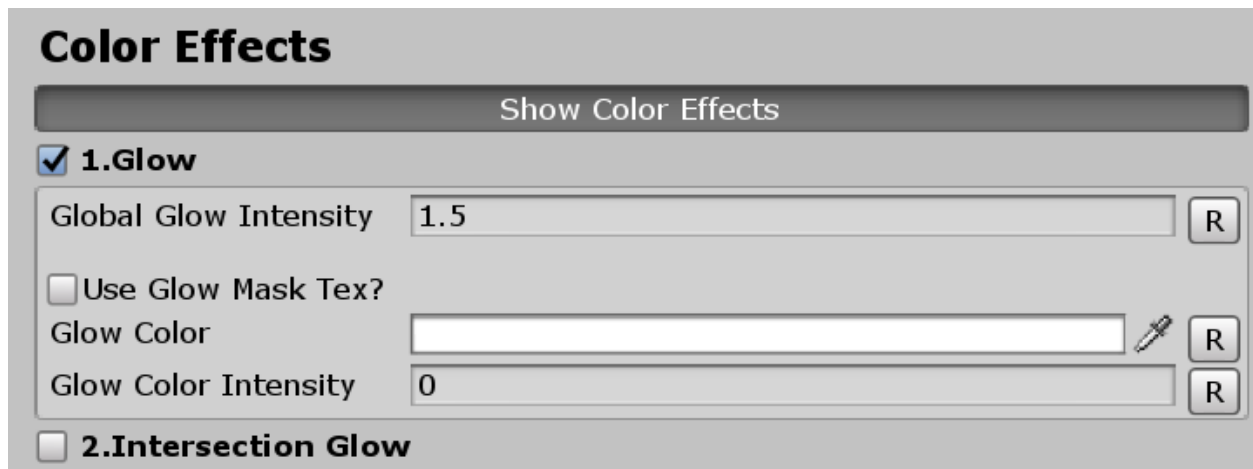
In this image we can see how the Glow Color Effect is enabled and all its properties.

Remember that effects will only be computed and therefore cost performance when enabled, otherwise the shader won't compile the effect code and therefore it will cost 0 performance since the code won't even be present on the shader.

Also note the "R" button next to each property that stands for "Reset" . Pressing this button will restore the property to its default value.

Keep in mind that there's an example of every effect on one of the Demo scene sample effects. Looking into the Demo is probably the best way to discover ways of using the asset and to see what each property does

Down below all the shader properties are explained. In between [] (ex:[GLOW_ON]) you can find the shader keyword name of each effect (see How to Enable/Disable Effects at Runtime section to see how to use it). In between () (ex: _MainTex) you can find the shader property names in case you want to modify them in a script (see Scripting section).

Before reading this please make sure to be familiar with the shader by reading the

<u>Shader Structure and Usage</u> section or by taking a look at the asset Youtube playlist before moving on.

Reminder of how the asset works: First the shapes get combined, this shape result will have the rest of effects applied to it. We'll call Global something that affects the final result at the end of this process.

- **Global Properties**
    - Global Color (_Color): The Global Tint color
    - Global Alpha (_Alpha): Global Transparency

- Shape Properties and Effects (N means the shape number but Shape 1 omits the number in properties. So _ShapeNColor would be _ShapeColor for Shape 1, _Shape2Color for Shape 2 and _Shape3Color for Shape 3. With effects such as SHAPE_N_CONTRAST_ON the N doesn't omit any number, so shape 1 will be SHAPE1CONTRAST_ON. Sorry if this is confusing but changing this meant losing all the demo material values. In any case these names are only used for scripting and the asset will very rarely be modified via script)
    - Shape1 Texture (_MainTex): Shape 1 Texture and also the image that will be automatically assigned in Sprite Renderers and UI Image components. Renderer components that have an Image property will override this texture. This will be the base shape texture that will be used to get combined with the other 2 shapes
    - Shape 2 and 3 Texture (_Shape2Tex and _Shape3Tex): These 2 shape textures will get combined with shape 1 and with each other if they are both enabled. These 2 textures will always be set through the material inspector
    - ShapeN Color (_ShapeNColor): Tint Color of the Shape Texture. This will be applied before combining the shapes and the color is HDR so can be used to affect color intensity and therefore glow
    - ShapeN X/Y Speed (_ShapeNXSpeed and _ShapeNXSpeed): These 2 properties are in charge of scrolling the shape texture over time. The higher the number, the faster the scroll will be
    - ShapeN Contrast [SHAPE_N_CONTRAST_ON]: Used to apply more or less contrast and more or less brightness to the shape texture before mixing the other shapes and applying effects
        - ShapeN Contrast (_ShapeNContrast): High values mean high contrast, low values mean low contrast (the shape texture turns gray when untinted)

- ■ ShapeN Brightness (_ShapeNBrightness): Brightness amount we add to the shape texture
- ○ ShapeN Distortion [SHAPE_N_DISTORT_ON]: Used to distort the shape texture before mixing the other shapes and applying effects
    - ■ Distortion Texture (_ShapeNDistortTex): Noise texture that determines how the distortion is done
    - ■ Distortion Amount (_ShapeNDistortAmount): How much the image is distorted following the distortion texture pattern
    - ■ Scroll Speed X and Y (_ShapeNDistortXSpeed and _ShapeNDistortYSpeed): Scroll speed of the distortion texture in the X axis and Y axis
- ○ ShapeN Rotation [SHAPE_N_ROTATE_ON]: Used to rotate the shape texture, this will work regardless of the tiling and offset of the textures
    - ■ Rotation Offset (_ShapeNRotationOffset): The initial rotation of the shape texture (measured in radians). This will be applied before the rotation over time of the next property
    - ■ Rotation Speed (_ShapeNRotationSpeed): How fast the shape texture will turn (measured in radians)
- ○ ShapeN RGB is Shape Color, Red Channel Is Alpha [SHAPE_N_SHAPECOLOR_ON]: When enabled the greyscale value from 0 to 1 of the red channel of the shape texture gets used as the alpha (transparency) of the shape texture and the RGB of the shape will be the Shape N Color property
- ○ ShapeN Screen Position UVs [SHAPE_N_SCREENUV_ON]: When enabled the shape texture will be samples using screen space coordinates instead of the model or quad UVs
    - ■ Scale With Dist Amount (_ScreenUvShNDistScale): If set to 1 the screen space UVs will scale the further/closer we get, so the shape texture will have a constant size. If 0 it will look as if the shape texture shrinks and expands as we get further/closer
- ○ ShapeN Debug [SHAPEDEBUG_ON]: When enabled only the current shape we have toggled will be rendered. This is extremely useful to see how each shape looks before combining with the other shapes and before applying effects
- ○ ShapeN RGB Weight (_ShapeNColorWeight): Affects how the shape RGB will be combined with other shapes RGB. See Shader Structure and Usage section for more detailed info

○ ShapeN A Weight (_ShapeNAlphaWeight): Affects how the shape alpha (A) will be combined with other shapes alpha. See <u>Shader Structure and Usage</u> section for more detailed info

- **Color Effects**: These effects apply color changes to the shape combination result
1. Glow [GLOW_ON]: Needs Post Processing Bloom for better visuals
    a. Glow Color (_GlowColor): Color of the Glow, has HDR and can also affect how bright the material shines by increasing the HDR intensity
    b. Glow Intensity (_Glow): Indicates how bright the Glow Color will shine
    c. Glow Global Intensity (_GlowGlobal): Indicates how bright the global result will shine
    d. Glow Texture (_GlowTex): Acts as a mask. The glow will only be applied where the alpha of this texture is greater than 0
2. Color Ramp [COLORRAMP_ON]: Takes a gradient as an input and maps the color of the shapes result to this gradient. When the Material is Saved To Folder we can use a live editable gradient instead of a static gradient texture
    a. Color Ramp Texture (_ColorRampTex): Static gradient texture that the user needs to provide. The resulting colors will get mapped to this gradient. Dark colors will get mapped to the left of the gradient and bright colors to the right of the gradient
    b. Color Ramp Luminosity (_ColorRampLuminosity): This will get added to the greyscale value the shader uses to map the colors. The higher this value the more we'll skew results to the right of the gradient
    c. Color Ramp Gradient (_ColorRampTexGradient): If the Material is Saved To Folder and the Use Editable Gradient box is toggled this property can be used to dynamically edit and create a gradient color
    d. Color Ramp Blend (_ColorRampBlend): Used to blend the gradient in and out. A Blend amount of 0 will make this effect invisible
3. Color Grading [COLORGRADING_ON]: Similar to Color Ramp but more lightweight and slightly more limited. It works just like Color Ramp but it interpolates the result color across 3 input colors instead of a full gradient
    a. Light Color Tint (_ColorGradingLight): Equivalent to the very right of the color ramp. Light color values will be mapped to this color
    b. Mid Tone Color Tint (_ColorGradingMiddle): Equivalent to the center of the color ramp. Middle range color values will be mapped to this color
    c. Dark/Shadow Color Tint (_ColorGradingDark): Equivalent to the very left of the color ramp. Dark color values will be mapped to this color

d. Mid Point (_ColorGradingMidPoint): Used to skew the result towards the Light Color or the Dark Color input
4. Hue Shift and Saturation [HSV_ON]:
   a. Hue Shift (_HsvShift): How much the colors will be shifted
   b. Hue Shift Saturation (_HsvSaturation): Saturation of the hue shift result
   c. Hue Shift Bright (_HsvBright): Brightness of the hue shift result
5. Fresnel / Rim Color: Creates a rim light / fresnel effect around the target Mesh. This effect can add color or make the Mesh transparent around the rims
   a. Rim Color (_RimColor): Color of the effect, has HDR and can also affect how bright the material shines by increasing the HDR intensity
   b. Rim Bias (_RimBias): Adds this amount to the "amount of rim" the shader detects and therefore makes all parts of the mesh be considered part of the rim (in most cases you'll want to use the Rim Scale instead)
   c. Rim Scale (_RimScale): Multiplies this amount to the "amount of rim" the shader detects and therefore makes the rim wider
   d. Rim Power (_RimPower): Exponent of the "amount of rim" the shader detects and therefore can be used to further fine tune the rim width
   e. Rim Intensity (_RimIntensity): Indicates how bright the Rim Color will shine
   f. Add Amount (_RimAmount): 0 means that the rim color we calculate will be multiplied against the shape result, this is useful when we want the effect to not affect darker parts, it looks like it follows the texture. 1 means the rim color will be added and always visible regardless of the shape color. Any value in between will be an interpolation between both results
   g. Rim Erodes Alpha (_RimErodesAlpha): Increase this value and set Rim Intensity to 0 to fade the model rim
6. Intersection Glow[DEPTHGLOW_ON]: Applies glow to intersecting geometry that writes to the depth buffer. Needs Post Processing Bloom for better visuals. For this effect to work as intended ZWrite should be disabled
   a. Depth Distance (_DepthGlowDist): How sensible the depth difference is. The higher the value the sharper the glow will be
   b. Depth Power (_DepthGlowPow): Exponent of the depth difference that creates the glow mask
   c. Glow Color (_DepthGlowColor): Color of the Glow, has HDR and can also affect how bright the material shines by increasing the HDR intensity
   d. Glow Color Intensity (_DepthGlow): Indicates how bright the Glow Color will shine
   e. Global Glow Intensity (_DepthGlowGlobal): Indicates how bright the global result will shine (needs Bloom in the scene)

7. Posterize [POSTERIZE_ON]: Limits the amount of colors creating a banding effect
    a. Posterize Number of Colors (_PosterizeNumColors): The higher the number the more different colors the material will display
8. Backface Tint [BACKFACETINT_ON]: Tints the front and back face of the target mesh. We consider the backface the face that has the normal vector pointing in the opposite direction of what we would regularly expect on a surface
    a. Backface Tint (_BackFaceTint): Tint Color back face of the mesh. The color is HDR so can be used to affect color intensity and therefore glow
    b. FrontfaceTint (_FrontFaceTint): Tint Color front face of the mesh. The color is HDR so can be used to affect color intensity and therefore glow. Usually we'll keep this white to not affect regular facing mesh faces
9. Fake Light And Shadow [LIGHTANDSHADOW_ON]: As the name says, this isn't real lighting that will work with Unity lighting components, instead this is a simplified performant lighting approximation that can be used to give some more depth to your effects. For this to work you'll need to have an active AllIn1VfxFakeLightDirSetter component in the scene, this component will tell the shader what the direction of the light is. Configure said component to choose when the light direction is updated and to choose the target transform that will determine the light direction with its forward vector. Video using and explaining the effect: https://youtu.be/F24wH7q34Xs
    a. Light Amount (_LightAmount): This is how much extra luminosity we want. 0 means no extra luminosity and 1 means that we want to light the object based on the direction of the target light set in AllIn1VfxFakeLightDirSetter
    b. Light Color (_LightColor): The color of the fake light, use this to choose the tint of the light set by the previous property. Note that this is an HDR color and that can be used to make the object glow by increasing the intensity
    c. Shadow Amount (_ShadowAmount): This tells the shader what is the darkest shadow value. So 0 will mean that the shadows can be completely black and 1 will mean that the shadows won't be visible at all
    d. Shadow Min and Max (_ShadowStepMin and _ShadowStepMax): This is used to choose the banding on the shadow, the closer these values are together the more cartoon the shadow will look
10. Shape 1 Mask [SHAPE1MASK_ON]: Prevents Shape1 from being affected by other shapes and distortions. This is used to keep certain parts of the shape intact, for example in a trail to make sure the beginning of the trail has no gaps
    a. Shape 1 Mask Texture (_Shape1MaskTex): Mask that tells the shader what parts of Shape 1 to keep. White means keep Shape 1 intact and black means keep shape result, in between values get blended

b. Shape 1 Mask Power (_Shape1MaskPow): Exponent of the Mask Texture, used to fine tune the results

● **Alpha Effects**: These effects apply color changes to the shape combination alpha result
1. Alpha Mask [MASK_ON]: Will make certain parts of the global result transparent
   a. Mask Texture (_MaskTex): White on the texture means unaltered original opacity. Black means fully invisible
   b. Mask Power (_MaskPow): Exponent of the Mask Texture, used to fine tune the effect without doing any change to the Mask Texture
2. Fade From Noise Texture (Dissolve) [FADE_ON]: Uses a Fade Texture to fade/dissolve the global result. Particle system alpha or Custom Data Streams can be used to control this dissolve. A burn texture can be added to have a different color around dissolve edges
   a. Fade Amount Affects Global Transparency (Toggle Box) [ALPHAFADETRANSPARENCYTOO_ON]: When checked the global transparency will decrease with the Fade Amount
   b. Fade Amount Driven By Vertex Stream? (Toggle Box that only appear when a Particle System is present) [ALPHAFADEINPUTSTREAM_ON]: When enabled the Fade Amount can be driven with Custom Data streams. Check the Custom Vertex Streams and Custom Data Auto Setup for more details
   c. Fade Texture (_FadeTex): Maps how the fade will be made. The fade will be made from black to white
   d. Fade Amount (_FadeAmount): How much fade to apply. -0.1 is no fading and 1 is completely faded
   e. Fade Transition (_FadeTransition): How smooth the dissolve edges are. The higher the number the smoother it will be
   f. Fade Power (_FadePower): The exponent of the fade amount, used to tweak how soon/late the fade happens
   g. Speed X/Y Axis (_FadeScrollXSpeed and _FadeScrollYSpeed): Scroll speed of the fade texture in the X axis and Y axis
   h. Fade Burn Texture (_FadeBurnTex): Texture of the burned edge
   i. Fade Burn Color (_FadeBurnColor): Color of the burned edges, has HDR and can also affect how bright the material shines by increasing the HDR intensity
   j. Fade Burn Width (_FadeBurnWidth): How smooth the burn edges are. The higher the number the smoother it will be

k. Fade Burn Glow (_FadeBurnGlow): Indicates how bright the texture will shine (needs Bloom in the scene)

3. Fade From Final Shape (Procedural Dissolve) [ALPHAFADE_ON]: Almost identical to the previous effect but takes the final shape as fade mask instead of taking a fade texture

   a. Fade Amount Affects Global Transparency (Toggle Box) [ALPHAFADETRANSPARENCYTOO_ON]: When checked the global transparency will decrease with the Fade Amount

   b. Fade Amount Driven By Vertex Stream? (Toggle Box that only appear when a Particle System is present) [ALPHAFADEINPUTSTREAM_ON]: When enabled the Fade Amount can be driven with Custom Data streams. Check the Custom Vertex Streams and Custom Data Auto Setup for more details

   c. Use grayscale as alpha [ALPHAFADEUSEREDCHANNEL_ON]: Enable this on additive configurations or when the shape result has no alpha. This will premultiply the red channel of the shape result into the alpha channels giving better results on the described cases

   d. Use Shape1 as fade mask [ALPHAFADEUSESHAPE1_ON]: When checked the shader will take Shape1 as fade mask input instead of using the combined shape result (only useful when using more than 1 shape)

   e. Fade Amount (_AlphaFadeAmount): How much fade to apply. -0.1 is no fading and 1 is completely faded

   f. Fade Transition (_AlphaFadeSmooth): How smooth the dissolve edges are. The higher the number the smoother it will be

   g. Fade Power (_AlphaFadePow): The exponent of the fade amount, used to tweak how soon/late the fade happens

4. Soft Particles / Intersection Fade [SOFTPART_ON]: Fades the global result when close to other meshes that write to the depth buffer. Used to fade things that get close to the floor for example

   a. Soft Particles Factor (_SoftFactor): The higher this value is the thinner the fade transition will be

5. Camera Distance Fade [CAMDISTFADE_ON]: Fades the global result when the camera is either too close or too far away

   a. Far Fade Start Point (_CamDistFadeStepMin): At this distance from the cam the material will start to fade

   b. Far Fade End Point (_CamDistFadeStepMax): At this distance from the cam the material will be completely faded

   c. Close Fade Start Point (_CamDistProximityFade): When the camera is closer than this distance the material will start fading

6. Alpha Remap [ALPHASMOOTHSTEP_ON]: Remaps the global result alpha to a custom range of your liking
    a. Smoothstep Min (_AlphaStepMin): Low bound of the new alpha range, where the previous alpha was 0 now it will be Smoothstep Min
    b. Smoothstep Max (_AlphaStepMax): High bound of the new alpha range, where the previous alpha was 1 now it will be Smoothstep Max. Everything in between gets interpolated
7. Alpha Cutoff [ALPHACUTOFF_ON]: Used to discard pixels and reduce overdraw
    a. Alpha cutoff value (_AlphaCutoffValue): Pixels that are more transparent than this value are not drawn. This is useful to make more cartoon looking effects and to discard unwanted transparencies from certain effects

- **UV and Vertex Effects**: These effects will affect the texture coordinates of the textures used by the shader or the vertex position of the mesh the material is using
1. Global Distortion [DISTORT_ON]: It will distort all textures used by the shader
    a. Distortion Texture (_DistortTex): Noise texture that determines how the distortion is done
    b. Distortion Amount (_DistortAmount): How much the image is distorted following the texture pattern
    c. Distortion scroll speed (_DistortTexXSpeed and _DistortTexYSpeed): Scroll speed of the distortion texture in the X axis and Y axis
2. Global Polar Coordinates [POLARUV_ON]: Transforms the uv coordinates into polar coordinates (this effect looks goods with tiling on the textures + texture scrolling)
    a. Polar Coords affects Distortion textures [POLARUVDISTORT_ON]: We may or may not want the distortion textures to be transformed to polar coordinates. This toggle allows you to choose
3. Shape Weights Custom Stream [SHAPEWEIGHTS_ON]: This effect will only be visible when viewing the Material from an object that has a Particle System on it or if the effect was already active. The effect can be used to change the weights of each Shape through a Particle System Custom Data, the ones that you can find in the Shape Result tab of the Material when more that 1 Shape is active
    a. Shape N Blend Offset (_ShNBlendOffset): This value will be multiplied by the offset value that comes through the vertex stream. It allows us to choose the direction and magnitude of the offset
4. Texture Offset Custom Stream [OFFSETSTREAM_ON]: This effect will only be visible when viewing the Material from an object that has a Particle System on it

or if the effect was already active. The effect can be used to scroll or offset each Shape texture through a Particle System Custom Data

    a. Shape N Offset Mult (_OffsetShN): This is the multiplier of the offset value that comes through the vertex stream. It allows us to choose how much effect the weight change should have over each shape and what direction they scroll on. So with only 1 vertex stream value we can affect each shape differently

5. Shape Texture Offset [SHAPETEXOFFSET_ON]: This effect will help us avoid repetition on objects that use this same Material. It will use the Random Time Seed (_TimingSeed) to offset Shape textures

    a. Shape N Mult (_RandomShNMult): Controls how much the Timing Seed will offset each Shape, 0 means no variation, 1 means that it will fully use the Timing Seed but it will probably give you the same results than 0 since the texture coordinates may loop around perfectly. Try using values between 0 and 1 for better results

6. Global Texture Scroll [TEXTURESCROLL_ON]: It will scroll all textures the shader uses

    a. Texture Scroll Speed X (_TextureScrollXSpeed): Scrolling speed on the X axis

    b. Texture Scroll Speed Y (_TextureScrollYSpeed): Scrolling speed on the Y axis

7. Twist [TWISTUV_ON]

    a. Twist Amount (_TwistUvAmount): How much all textures are twisted

    b. Twist Pos X Axis (_TwistUvPosX): Position of the center of the twist on the X axis (0 is left and 1 is right)

    c. Twist Pos Y Axis (_TwistUvPosY): Position of the center of the twist on the Y axis (0 is bottom and 1 is top)

    d. Twist Radius (_TwistUvRadius): The radius of the twist effect

8. Wave [WAVEUV_ON]: Distort waves from left to right

    a. Wave Amount (_WaveAmount): How many waves we make

    b. Wave speed (_WaveSpeed): How fast the wave scrolls

    c. Wave Strength (_WaveStrength): How much the wave affects the textures

    d. Wave X Axis (_WaveX): Position of the wave origin on the X axis (0 is left 1 is right)

    e. Wave Y Axis (_WaveY): Position of the wave origin on the Y axis (0 is bottom 1 is top)

9. Round Wave [ROUNDWAVEUV_ON]: Radial distort waves

   a. Round Wave Strength (_RoundWaveStrength): How much the wave affects the textures

b. Round Wave Speed (_RoundWaveSpeed): How fast the wave scrolls

10. Hand Drawn [DOODLE_ON]
    a. Hand Drawn Amount (_HandDrawnAmount): How much of a distortion we apply to make it look hand drawn frame a frame
    b. Hand Drawn Speed (_HandDrawnSpeed): How often we distort the textures

11. Pixelate [PIXELATE_ON]
    a. Pixelate size (_PixelateSize): The lower the number the more pixelated the textures get. This effect looks bad when combined with distortions

12. Trail Width [TRAILWIDTH_ON]: Offers you fine control over the scale of the vertical texture coordinate. This allows you to scale a trail without the common Trail Renderer component artifacts. When using this please keep a constant width across the whole trail in the Trail Renderer component. **This component needs the material to be Saved To Folder in order to work**
    a. Trail Width Power (_TrailWidthPower): The exponent of the trail width set by the following property. Allows to fine tune the results without editing the gradient
    b. Trail Width Gradient (_TrailWidthGradient): A custom gradient property that allows you to edit a texture with the Unity gradient window (read the Custom Gradient Property Drawer for more info). Black means scale 0 and therefore 0 width trail. White means scale 1 and therefore maximum scale width trail

13. Shake [SHAKEUV_ON]: Shakes all texture coordinates
    a. Shake Speed (_ShakeUvSpeed): How fast it shakes
    b. Shake X Multiplier (_ShakeUvX): The higher the value the more it will move on the X axis while shaking
    c. Shake Y Multiplier (_ShakeUvY): The higher the value the more it will move on the Y axis while shaking

14. Vertex Offset [VERTOFFSET_ON]: Displaces the vertices of the mesh
    a. Offset Noise Texture (_VertOffsetTex): Tells the shader how to offset the vertices. 1 means maximum offset, 0 means no offset
    b. Offset Amount (_VertOffsetAmount): The offset amount, this will set the max offset distance
    c. Offset Power (_VertOffsetPower): Exponent of the offset amount used to fine tune the effect
    d. Scroll Speed X/Y (_VertOffsetTexXSpeed and _VertOffsetTexYSpeed): How fast the Offset Noise Texture will scroll on the X and Y axis

Custom Gradient Property Drawer

The asset shaders use a custom Gradient Property Drawer on some shader properties that allows you to use the built-in Unity gradient inspector to create a texture that will be saved inside the Material that uses it.

The properties that use this custom drawer are the Color Ramp Gradient in the Color Ramp effect (when Use Editable Gradient toggle is enabled) and the Trail Width Gradient property on the Trail Width Effect.

If you create a gradient texture but then you change your mind and you don't use it any more you can delete it by right clicking on the Material that holds the texture and press AllIn1Vfx -> Remove All Gradient Textures:
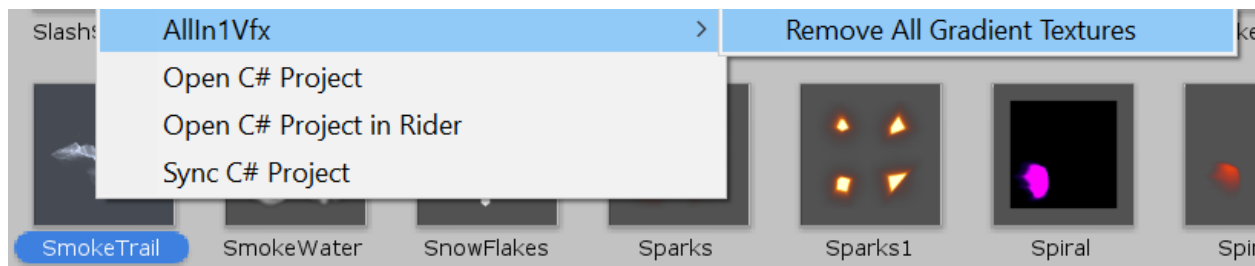
Running out of Shader Keywords

If you are using other assets or if you've written some complex shaders yourself you may run out of shader Keywords. Unity hass 256 possible global Keywords for shaders, Unity itself takes around 60 of them, so the user has around 190 available Keywords. This asset uses many Keywords, so running out of them may be a possibility if you are using other assets.

So what's the solution? Since Unity 2019.1 Unity has included local Keywords. This asset is prepared to work with any Unity version and that's why these local Keywords aren't used. But if you are on Unity 2019.1 onward this is what you can do:

1. Go to: AllIn1VfxToolkit\Shaders\Resources
2. There you'll all shader variants there
3. Open the shaders you use
4. Change all shader_feature for shader_feature_local (in visual studio ctrl+f will open the search and replace bar)

*Unity will only accept 64 local keywords (shader_feature_local), with new updates and features the shader has slightly surpassed this number. You will need to leave out a few keywords as global keywords (shader_feature). Keep in mind that you will only run out of keywords if you have other assets with big shaders in your project and that in any case you can just replace the keywords on those assets to be local too.

Credits

This asset has been made possible thanks to the collaboration of 2 amazing professional VFX artists that made the majority of the Demo examples and provided valuable feedback after using early versions of the asset.

These artists are:
- Dmitry Bogatov (also known as Destroyeer):
  - https://twitter.com/DestroyeerVFX
  - https://www.artstation.com/Destroyeer
- Yos Ytomacedo:
  - https://twitter.com/YYtomacedo
  - https://www.artstation.com/yosdevvfx

A few assets under CC0 1.0 Universal license are sourced from Kenne:
https://www.kenney.nl/assets

Special thanks to my friend Antón Miranda for putting together an amazing cover art image, trailer and screenshots for the storefront.