If you are reading this documentation in PDF or TXT formats, unless you have trouble connecting to internet or accessing Google, preferably use the online version:
https://docs.google.com/document/d/14b6Kv7fOuIA79X3YKMCDRyqOds4JD5QVpqiQUhNYS94

Unity Asset Store Link: https://www.assetstore.unity3d.com/#!/content/101494
Email us: moonflowercarnivore@gmail.com
Our Facebook group: https://www.facebook.com/MoonflowerCarnivore
Unity General Graphics forum: https://forum.unity3d.com/forums/general-graphics.76/

Disperse Pixels by Moonflower Carnivore creates a visual effect that disintegrates an object, either mesh, sprite or text, into particles. The concept and technique are inspired by Elvis Deane's tutorial for Particle Illusion (now acquired by Boris FX) created by Alan Lorence.

To use this manual more efficiently for troubleshooting, you can press Ctrl+F to look for the keyword of your question. If nothing matches your question about this asset, you can email us (moonflowercarnivore@gmail.com) for help. Also follow us on Facebook and Tumblr for latest news and knowledges.

Table of contents

Change log

- V1.00 (January 2018):
    - First release for Unity 2017.3
- V1.01 (January 2018):
    - Minor fix for Unity 2017.3.0p2 and p3
- V1.1 (March 2018):
    - Removed chroma keying.
    - Fixed Particle Opaque shader.
    - emitterBoxSize now uses an array of Vector3 instead of one Vector2 for all targetObjects. Please redefine the values otherwise in edit mode there will be lot of error messages.
    - Added “transparent target” option.
    - Added support for UI image and UI text.
    - Added options to “reinstantiate” or “destroy on disable”. If both are unchecked, the script reuses existing particle system instances instead of instantiating new instances.
    - Optimized code for reducing GetComponent.
    - Added gizmos box for indicating particle emitter box dimension in scene view.
    - Added GUI handles for adjusting Emitter Box Size and P System Offset position.
- V1.12 (April 2018):
    - Bug fix for execution during zero time scale.

How it works

The Disperse Pixels script handles 2 different tasks simultaneously: Capturing a screenshot for tinting particles and fading the material of the renderer.

First the target mesh is temporarily assigned to an unimportant layer, a temporary camera, assuming the same position of the Main Camera, captures a screenshot of object only appears in that layer. The screenshot is used to apply its RGBA value on the Start Color of each particle of same screen projection position converted from its world position. The script has commented lines for HDR mode but it would only work when the Start Color of particle is no longer restricted by the 8-bit (LDR) color vertex input stream. Currently there is no way to circumvent this limitation as the custom data property of each particle is not yet exposed to user.

The screenshot is transparent base, if the particle is tinted by the any color with an alpha value lower than 1, it is destroyed instantly. For transparent objects, the script takes a second screenshot with the object's background of its original layer, each particle is than tinted by the second screenshot but use the alpha value of the first screenshot. Shader for the transparent target object must use Mobile/Particles/Alpha Blended for correctly inheriting the color from the second screenshot. For a mesh model with both opaque and transparent parts and both require dispersing effect, you need to assign two separate Disperse Initiators and Particle prefabs.

Because the tinting happens on 2D projection, particles emitted from the backside of the 3D mesh will be tinted by color of the frontside, in other words, when the Disperse effect just beings, viewing the effect from behind will expose the oddity (easily observable in Scene view by not aligning the viewport with the main camera). The additional procedures, like capturing the second screenshot and calculating the dot products of over hundreds to thousands normalized vectors of particles, to handle the backside tinting are not cost-effective. You can rotate the camera during "in" Disperse effect, but make sure the camera resumes the angle when the effect is triggered near the end of the particle lifetime for the sake of consistency.

The fading of object simply does a linear interpolation of alpha of tint color or cutoff threshold provided in the shader of the material. You may use Material Substitute and Original properties to switch material temporarily, this is especially useful for transparency fading from opaque material when transparent shading causes z-fighting glitch. If no matching property is found in the shader, the system will give an error log, in such case, consider to use the "instantly" fading mode. For fading UI image and UI text, it simply fade their color property to which only work with "Out By Transparency Color" and "In By Transparency Color", otherwise you would use "Out/In Instantly".

After the Disperse effect is finished and the script is disabled, the screenshot, clones of Disperse Camera and Disperse Particle are destroyed. If Animator is assigned to be disabled by this script, Animator is re-enabled. If Material Substitute is called, it will be reverted to the original material as “Shared Material” to prevent instancing. In case you want to instance the material, which is mostly undesirable, you need to change the command with “sharedMaterial” to just “material” in the Disperse Pixels script.

The Particle Homecoming script is attached on any particle system object for “in” Disperse Particle effects. It works by storing the start position of each particle in an array with a size same as the current particle count because the particles are not yet allowed to move which is controlled by the Speed Modifier curve in Velocity Over Lifetime module. After the particles are allowed to spread (or contract in some case), the script orders all particles to move back to their start positions. Often time the desktop particle effect is given some noise movement but faded to zero noise strength at about 0.5 to 0.7 of their lifetime, otherwise the particles will keep straying at the end of effect. Because of a by-design limitation of particle system, the lifetime of all particles should use only the same constant value instead of random value within a range, the latter will cause the particles to be refreshed when they are supposed to be dead. Like the Disperse Pixels script, Particle Homecoming runs in a coroutine for performance’s sake and works on particle system with a one time burst from the very beginning.

During the “in” Disperse effect, the Fading Delay and Fading Duration works slightly differently than the “out” counterpart. The Disperse Particle effect plays from the beginning anyway, but the material fading interpolation (either cutout or transparency) begins per Fading Delay. The resumption of animator begins after the interpolation is finished. Theoretically the values of Fading Delay plus Fading Duration should equal the lifetime of the Disperse Particle prefab. But you can set smaller Fading Delay and/or Fading Duration so the animator is resumed earlier than the end of the Disperse Effect. There is a trade off that if the sum of Fading is nearly identical to the particle lifetime, the resumption of animator may look delayed too much. This can be offset by allowing the material Fading to end slightly earlier or fine tuning the Particle Homecoming speed, but the drawback is that the particles at the end of their lifetime may look offset from the target. A workaround would be to set a slightly earlier timer in the Self Destroy script so the virtually static particles do not linger than they should be.

When using Out-In series for teleport effect, reference Toggle Control script, send the new Vector3 position value of teleport destination to disperseScript.outInNewPosition. Also if you move the target object, preferably do a Vector3.Lerp after the fade out of the material instead of changing the position instantly.

By design, Disperse Particles are not affected by shadow cast by other objects. As a workaround, duplicate the shadow-casting objects which are expected to affect the Disperse Pixels effect, but change the cast shadow mode to “shadow only”, assgin these duplicated objects to the layer used by Disperse Pixels script for screenshot capturing, usually the 31st layer. To minimize performance impact, only activate these shadow-only objects right before the Disperse Pixels effect and deactivate them immediately afterward.

Usage

You drag the prefab containing Disperse Pixels script of your choice to be the child of the target object. If you do not want to create new object just for containing this script, you can right-click “copy component” on the script title in the prefab and “paste as new component” to your target object which will work as well. If you do not want to contain this script in the same object as Renderer or as a child object parented to the target object, you can instantiate the script prefab as a standalone object and assign the object to the “targetObject” property in the script. Whatever your preference, the script should be initially disabled, only after dozens milliseconds after the execution of the scene, you set enabled of the script to execute the effect.

If your mesh object is being animated, you should stop the animation about 0.05 seconds before executing this script, otherwise the particles will not be emitter from the correct position but slightly offset.

Execution Interval property allows you to separate the dispersing effect by a constant amount. The execution order is the same as the Target Object array.

Example: Disintegrate Test Character 2

Test Character 2 (TC2) is included in this package under the Models folder for demonstration and instruction. As our legal customer, you can use the model, textures and animations just as the rest of the asset in this package, just to clarify that TC2 is not the selling point of this asset. The model was done in Blender, it is rigged, skinned, animated and divided into 2 parts: Main body and eyeballs. Actual character models usually are divided further more for clothes and hairs, but the usage of Disperse Pixels script is identical. First we drag the TC2 model to any scene then add the Disperse Pixels script to any object in the scene, it can be the model’s root Game Object. Disable the script because why would you disperse a newborn object?

The model has 2 parts which need to be disintegrated, so type “2” to the size of Target Object and then assign both mesh objects (which contains the Skinned Mesh Renderer component) to the Target Object array property in our script.

Assign the main camera in the scene to Main Camera property and “Camera disperse” from Prefabs folder for Disperse Camera property which is for capturing the screenshot.

The Animator property is for temporarily disabling the animation of the whole model whose Animator component is usually in the root model object, so assign the root Game Object to this property, or ignore it if you do not want to stop animation in case of not making the model disappear at all for effects like blink teleportation. However, if the model still temporarily disappears before emerging into the scene, it is still worth pausing the animation. Animator is resumed when this script is disabled or destroyed.

For Disperse Particle property, assign any particle prefab in Prefabs folder of your choice.

To improve performance, we set higher Downsampling factor, 4, which resizes the width and height of the screen shot. Higher factor will reduce tinting accuracy. If the particle prefab is Cloudy type, it is not too noticeable with poorer sampling. You may use 2 or even 1 for Sandy type if performance allows.

The size or length of Position Offset and Max Particles properties must match that of Target Object, so type “2” to the Size fields of both. Position Offset values can retain as is, or for prefabs like Disperse Particle Cloudy Noise Ghostly, give the main body part (0,1.4,0) to position the sub emitter “glass distortion barrel” properly.

Max Particles count is a tricky one if your model is heavily divided, also different type of particle texture requires different amount of particles to fill in the gaps initially. You need to check the mesh area of each part to assess how many particles are needed to be emitted from that mesh. For Sandy type, usually the smaller parts like eyes need about 20 particles; larger parts like skin, shirt, leg and tails would need near 1000. Once you are satisfied with the ratio, you can only adjust the Max Particles Multiplier which is especially helpful when you change the Disperse Particle prefab. For example, when you use any Cloudy type prefab, you may also change the Max Particles Multiplier from 1 to 0.5 instead of modifying all values in the Max Particles array one by one manually. In fact, because the eye sockets of the main body is already painted as a fail-safe, we can set zero particle for eyeballs, this will prevent the Particle Prefab from being instantiated for the eyeballs.

Finally we need to decide how the model disappears. All parts of Test Character 2 use the built-in Standard Shader. If you use custom shader, try to incorporate Cut Out or Fade render modes like the built-in ones. Luckily if you have no energy to bother with rewriting shader, we have a shortcut to hide the object instantly by disabling the Renderer component, so we choose “Instant Hide” in Fade Out Mode. Fade Out Delay again is based on the type of Disperse Particle prefab we have chosen. “Cloudy slow” would need about 0.3 seconds, while “sandy explode” or “cloudy noise ghostly” will be fine with 0.1 seconds. You may fine-tune the delay by pausing the playing scene, enable the script, and proceed each frame by slowly mashing the “Step Forward” button for inspection.

Congratulations! You have successfully disintegrated Test Character 2 yourself!

If you want to play sound effect, assign the “Audio Source disperse” prefab to Audio Source property and select the Audio Clip which matches the Disperse Particle effect. The source will be parented to the selected Target Object.

Create new Disperse Particle prefab

You can modify from our presets in the Prefabs folder, preferably duplicate (Ctrl+D) the source in the project folder and modify on the copy only. If you create your own from scratch, there are few rules apply for this script specifically.

In the Emission module, remove the rate over time/distance and add a burst of 10,000 particles. This is going to be capped by the Max Particles property so we want to have a high enough emission to begin with. Just make sure the Duration in the Main module is 0.1, then Looping and Play On Awake are both unchecked. You may change the Max Particles in Main module just for preview and denoting the recommended amount of emission, even though this is to be overridden by the same property of the script.

Curves of Velocity Multiplier (Velocity over Lifetime module) and either Size over Lifetime module or alpha of Color over Lifetime module need to be given near zero and zero value respectively in the beginning (time: zero). The velocity multiplier is for halting the movement so the particles stay at their initial position to be tinted by the screenshot.Giving Velocity Multiplier exactly zero value will result in infinite error, hence the near zero value, e.g. 0.01.

Performance tips

- Assign your main camera in the scene to the script manually. While the script will use Camera.main if none is assigned, the issue is that this property is a shortcut for Object.FindObjectWithTag("MainCamera"). Not as terrible as GameObject.Find, but why the laziness?
- Disable "Cast Shadow" in Renderer module of particle system or change the particle material to "particle unlit". It is done this way for mobile variants anyway.
- Pixelate and Sandy effects use opaque material because it is cheaper than transparent material. Transparent material also has the issue of lacking depth writing in most built-in shader and resulting z-fighting. Alpha depth test still generally causes performance issue on mobile.
- Disable and re-enable the script instead of destroy-instantiating the script prefab again if the target object is not to be destroyed but keep being active in the same scene such as teleport/blink or a dead mob object to be respawned constantly. Destroy occasionally falls victim of memory leak bug, when you instantiate the prefab of a destroyed object, the memory usage stacks. Certainly you would file a bug ticket to Unity and expect it to be fixed very soon, but mostly you would like to avoid it altogether.
- Killing and respawning the same object can be simply done by disabling the whole object. When it is time to respawn the target object, reposition and enable it again. Just remember to carefully time the action of disabling the whole object and the script, as Disperse Particle clone is parented to the target object, disabling the target object will kill all existing particles. Also when you disable the script and Fade Out Mode is anything but "none", the disabled Renderer component will be enabled again to visualize the object again. So you may disable the target object after 4 seconds (as the durations of all Disperse Particle effects are no more than that), and then disable the script after that to prevent the object from blinking.
- Change the particle collision mode from "World" to "Plane" and assign a game object for reference of the central location of the plane.
- You can fiddle with the particle system prefabs for more different behaviors. We choose not to cover some less common use cases to avoid unnecessarily derive variants which only confuse our customers and tax the performance. One possible customization you may be interested in if the "out" disperse effect is used on a running target is that you can enable Inherit Velocity module of particle system and give a positive multiplier value between 0 and 1 so the particles will move in the same direction as the target.
- Unity only begin supports GPU instancing for mesh render mode particles from 2018.1. Billboard particle is handled by dynamic draw call batching which still give users leeway of high emission as long as screen fillrate and transparent overdraw are kept properly low, so do not worry too much for the emission rate and always use profiler to investigate the true culprit of performance dip.

Known issues

- From Unity 5.4 to 2017.1, Stop and Simulate cause GetComponent allocs which reduce performance. This is fixed from 2017.2 onward.
- Tinting particles emitted from the backside by frontside already detailed in the previous section.
- If the mesh target is partially off-screen, particles emitted from those off-screen faces will pick the nearest visible pixel. Particle System also culls mesh faces too far away from the camera frustum, so those faces will not emit any particle at all.
- Disperse Particle effect is oversized if the mesh object scale is too high such as 100, this is usually the case when importing FBX from Blender. Uncheck the icon next to it which reads "scale all data according to current Blender size…" when exporting data in Blender.
- Some skinned mesh models from various external source also suffer this issue. To fix that, search all child objects containing the Skinned Mesh Renderer and change their transform-scale values back to 1. If you still do not see any particle, you may increase the Start Size of the assigned Disperse Particle prefab, as well as the Max Particle Size in the Renderer module of Particle System.
- The Armature object which controls the animation of the skinned mesh objects should use scale 1 as well, otherwise not only the size and speed of the particles are improperly scaled, the particle collision will look completely wrong. As a workaround to the buggy collision behavior, you need to divide the Radius Scale in Collision module by the same scale value you use to multiply the start size of the Disperse Particle prefab. Anyway, expect Unity to patch this collision radius bug soon.
- Disperse Particle effects such as Cloudy Ghostly which use glass distortion (grabpass) shader and ribbonized particle trails will create performance spike when they are initialized for the first time. To avoid sudden drop of frame rate when you actually use such effect, pre run the effect on Awake. The warm-up can be done off-camera. The demo scene has done that, contained in the "pre warm objects" for your reference.
- Ribbonized Particle Trails used in Cloudy Ghostly do not inherit color correctly [964377]. This issue is only fixed from Unity 2017.3.0p2.
- If Disperse Pixels script is executed when Time Scale is at zero, the script will fail to tint the particles even if the time scale is no longer zero. The engine console will also issue a warning message "*Internal: JobTempAlloc has allocations that are more than 4 frames old - this is not allowed and likely a leak*". You should either avoid executing the script during zero time scale, or cap the minimal time scale to like 0.01 if it is unintentional to have zero time scale while creating slow motion effect. This is partially fixed in v1.12.

Terms of Use

Disclaimer: This section only summarizes a portion of Unity's Asset Store Terms of Service and End User License Agreement (EULA). It does not override the agreement detailed on Unity's website if there is contradiction between the two.

This asset published by Moonflower Carnivore (the licensor) is intended to be only distributed publicly in your (the end user's) electronic game and interactive media, in the form of an executable which cannot be modified in any editor again, including but not limited to Unity editor.

You are free to use and modify this asset if you obtain this asset legally via Unity Asset Store firsthand and distribute your executable containing this asset legally.

You are not allowed to publicly reproduce, distribute, sublicense, rent, lease or lend this asset, either untouched or retouched, or any resource included in this asset for commercial or non-commercial purposes.

If you obtain this asset secondhand, not via Unity Asset Store firsthand, you are not authorized to use this asset, unless you obtain this asset legally via Unity Asset Store.

Per EULA, sharing this asset to other computers is only allowed:
(...) *provided that these computers are either all (i) physically located at a single physical location ("Site") belonging to END-USER, or (ii) laptops belonging to END-USER which have been made available by END-USER to its employees that are employed at the same Site provided all such computers have appropriately licensed Unity software installed.*

If you really want to include any single content of our assets in your own asset to be sold commercially, you must email us to negotiate a separate contract for permission.

If you like this asset, please rate it or leave your comment in our Asset Store item page. However, if you need technical support, you should email us directly. You may also be interested in other assets created by us, so check out our publisher page on Asset Store. Some assets are even offered for free, so don't delay.

Moonflower Carnivore
2015-2017

moonflowercarnivore@gmail.com