TexEmit - Particle Emission from Texture

Beta - January 2017

Index

TexEmit - Beta Overview

'TexEmit' in short is a tool that allows users to emit particles from locations on a mesh as defined by a texture using the inbuilt Unity particle system. This tool has been designed to allow you greater freedom in choosing how you would like to emit particles in your next game project.

To use TexEmit all that is required is a texture that serves as a map of locations from which to emit particles on your existing UV unwrapped mesh.

NOTE: This tool is in Beta and as such, users may experience some unexpected results when pushing it beyond our capacity to test at the present time. Please report any issues via email and we will address them ASAP. All core functions have been tested and are working as expected. Additional features will be added shortly in forthcoming updates.

Suggested use cases

- Great for static environment meshes that would benefit from emitting particles from texture details (cracks, lights, crevasses, emissive texture locations, monitor panels, etc…)
- Can be used to help bring attention to skinned meshes that will benefit from emission at particular locations as defined by a texture

What it can do

- Emit particles on a UV unwrapped mesh (skinned or static) by generating emission points from a reference texture

What it can’t do

- ‘TexEmit’ cannot emit from meshes with a triplanar shader. It currently only works on meshes with standard UV mapped textures. This is because we generate emission locations based on UV coordinates and as such if your mesh is not unwrapped, there are no coordinates for the tool to generate emission points from.
- User cannot currently generate new emission points on a prefab containing the ‘EmissiveParticleMesh’ script in the inspector. However, if the object is is in the hierarchy/ scene, the user can generate new points and apply the generated points to the prefabbed object.
- TexEmit cannot presently emit more than one particle system per game object with the ‘EmissiveParticleMesh’ script attached.

How to use TexEmit - First time users

Set Up

Thank you for purchasing TexEmit! We hope you get great use out of this tool we have crafted. Please feel free to create your own mesh and UV’s to follow along with the set up demo, but please also be aware we have included assets to demonstrate how the tool works. This tutorial uses the assets included in the package, so you are able to follow on exactly as is described below.

Step 1 - Import package, locate assets and set up mesh/ textures

Included in the package are a number of textures and a test mesh. Once you have imported ‘TexEmit’ you will find the folder titled ‘TexEmit’ below

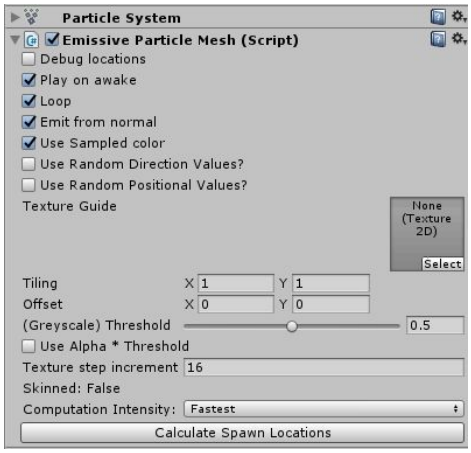
the 2Ginge folder (if you have any other 2Ginge tools or asset packs, they will also appear in this folder).

Open the meshes folder where you will find the ‘DemoSphere’ mesh and drag it into your scene. Within this folder you will also find a material titled ‘DemoSphere’, apply this to the “DemoSphere’ mesh, or feel free to create your own.

Once you have assigned a material to the ‘DemoSphere’ mesh, navigate to the ‘Textures’ folder, in which you will find a number of demonstration textures we have included. Again, feel free to create your own and ignore this step. For the purposes of this demo we will use the ‘LavaSphere’ texture. Drag the texture into the ‘Albedo’ slot while the ‘DemoSphere’ material is selected.

Next, drag the ‘LavaSphereEmission’ texture into the Emission channel and play with the emission colour until you are happy with the result.

Step 2 - Attach the ‘EmissiveParticleMesh’ script

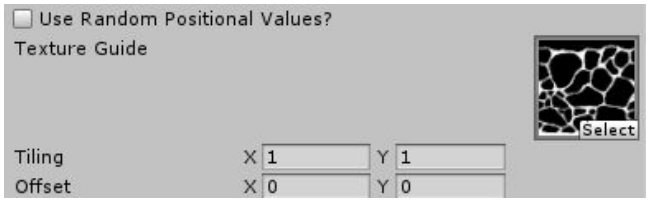


Now you have your mesh and textures ready, we can attach the ‘EmissiveParticleMesh’ script to the object you would like to emit the particles from.

Navigate to the ‘Scripts’ folder and drag the ‘EmissiveParticleMesh’ script onto your object. Now with your object selected, in the inspector you should find the ‘EmissiveParticleMesh’ and a Unity Particle System components present.

Step 3 - Component settings

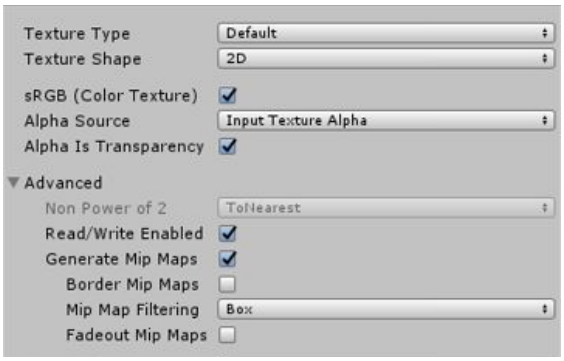
There are a number of options provided to the user under the ‘EmissiveParticleMesh’ component. Now there is a particle system attached to your object (the default Unity system), it will be emitting from the object center by default. This is because we do not have a reference texture for the script to generate emission points from just yet.



The texture guide texture put simply is a mask. This mask tells the script where to generate emission points form on your UV mapped mesh. When creating your own textures, bear in mind that white areas will be read as locations to generate emission points along and black areas will be ignored. It is important to note also that at present, the number of pixels in the reference texture directly influences the number of points on the texture that the script can generate emission points from. If the texture is 512x512 and the tiling is set to 0.5x0.5, then the number of pixels available to

reference from becomes 256x256 and less points can be generated than the original 512x512 texture as a result.

After importing a texture to use as your ‘texture guide’, ensure you check the ‘read write enabled’ selection box in the import settings visible when the texture is selected in the project view. All reference textures must have this enabled for the tool to work correctly.



In the textures folder you will find the ‘LavaSphereEmission’ texture. Drag this texture into the texture guide slot present on the Emissive Particle Mesh component. Now that the texture guide has been added, you can generate emission points using the ‘Calculate Spawn Locations’ button below. However, before you do this, you may want to adjust some settings.

Debug locations:
View debug indicators of generated location points, (these will also display the color of the spawned particle).

Play on awake:
Start the emission when the scene is run.

Loop:
Loops the particle system. Please use this loop rather than the loop setting in the particle system itself. We currently override the loop setting in the particle system.

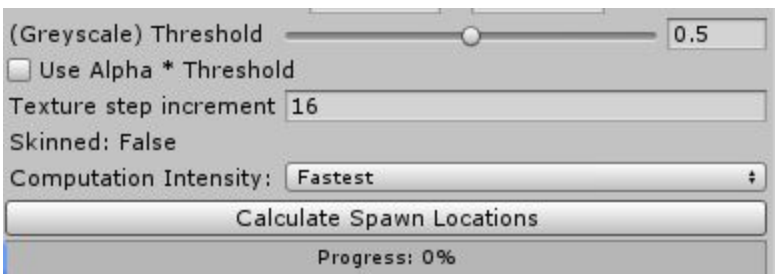
Emit from normal:
Samples the direction of the object's normals and emits towards the direction the normal faces.

Use sampled colour:
Sample the texture at the emission point and emit a particle of that colour from that location. In other words, emit a particle that is the colour of the texture at the location it emits from.

Use random direction values:
Randomly offsets the normals direction to ensure a more natural spread when emitting particles.

Use random position values:
Randomly offsets the emission point position slightly to give a more natural feeling spread.

Greyscale Threshold:
The ‘Greyscale Threshold’ determines the value between 0-1



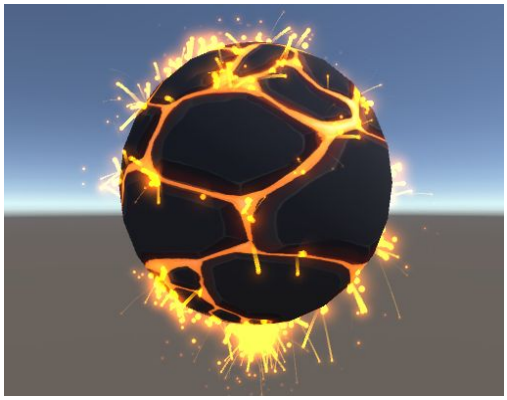
at which a point's color intensity (pixel value) must exceed to become a spawn point. Set the greyscale threshold to 0.05 for this particular example.

Use Alpha * (Multiply) Threshold:
The alpha multiply threshold comes into play by multiplying the pixel value. Regardless of the intensity or greyscale value of the pixel sampled if the alpha = 0, the pixel value = 0. You can use alpha to mask areas you do not wish to generate points along to ensure a hard cut off where gradient or black zone might not do. In the case of this reference texture, we can leave the box unchecked.

Texture Step Increment:
The texture step increment number determines how many pixels we skip (also known as stride) during each sample of the reference texture until the next check. For this example, let's set the increment number to 16.

Computation Intensity:
Computation intensity settings allow you to select the best performance level for your machine while generating spawn locations. Heavy will process the calculation quickly, but will dedicate more resources to the task and light will dedicate less resources to the task - taking longer - but working in a kinder manner to lower end PC's. An added benefit of selecting a slower calculation is that you can visualise the generation of the points with debugging turned on. This way you can see what you're going to get before it's done and save time if the result is not what you're after.

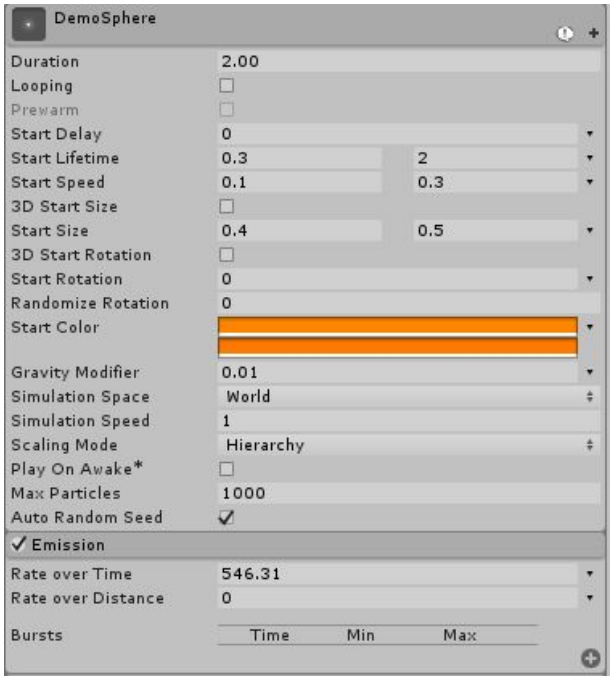
That's it! Now you've got your settings filled out you are ready to generate your emission points. Simply press the 'Calculate Spawn Locations' button and your emission points will generate. If you would like to visualise the spawned locations, turn 'Debug locations' on at the top of the "Emissive Particle Mesh' component. The object being calculated for must be selected while the calculation is taking place, otherwise the calculation will pause until the object is selected once more.

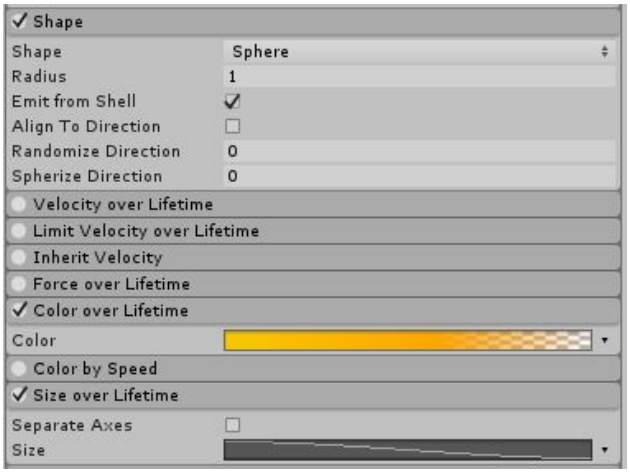


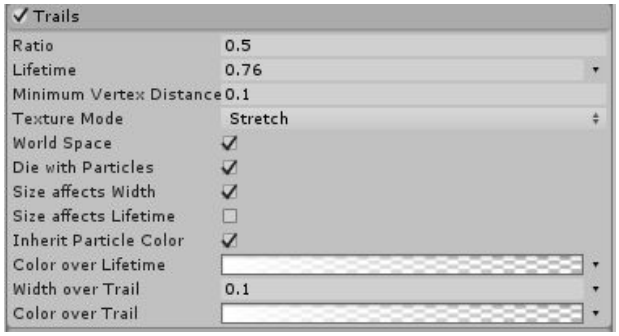
A quick note that in edit mode the particle system will not emit from the generated points. Only in play mode do the emission points emit the particles. The ability to visualise the emission from the generated points will be added post beta.

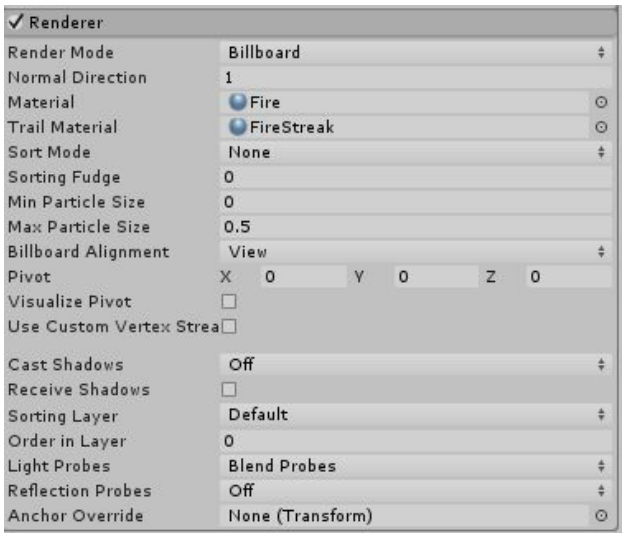
Particle System Settings:

The particle system settings determine how the particles emitted will look, just like a standard particle system would do. That said, below you will find the settings used in the demo scene for the lava ball. Simply copy these settings to achieve the same effect.









Step 4 - Runtime Functions

Loop(bool value):
Takes a value and sets the looping variable to that value, this either sets looping on or off in the script, however it will not begin playing.

Play():
Starts the emission.

Stop():
Stops the emission.

Best Practices

- Make sure the ‘editor’ folder is present in your project otherwise the ‘EmissiveParticleMesh’ script will not work
- Keep looping turned off within the unity particle system, instead use the looping checkbox located on the ‘EmissiveParticleMesh’ script
- Do not have ‘play on awake’ turned on within the particle system itself, instead select play on awake in the checkbox found at the top of the ‘EmissiveParticleMesh’ script
- Emission texture size will be best determined by your individual use case. For higher fidelity emission ‘more emission points’ a larger texture will be better than a small one. However, we also allow you to set your fidelity using the ‘texture step increment’ input. For example, by setting the texture step increment number higher, the less fidelity you will get when generating emission points.
- While higher res emission textures will give more fidelity while generating emission points, we recommend keeping the texture sizes down. As with all textures, this will help you optimise your project/ build size later.
- Power of 2 textures are suggested, but not necessary (for best performance use standard texture sizes; 512x512, 256x256, etc...)

Known Issues/ Bugs

- On skinned meshes, it is suggested that you turn debugging off when previewing animations. Your system will collect large amount of garbage and significantly increase computation intensity while attempting to update the debug locations each frame and likely crash on medium to lower end systems.
- Emission points will not generate unless object being generated for is selected (not particularly an issue or bug, bug something to keep in mind)

- Rate over distance emission type is disabled in this version (will not work for skinned/ animated meshes), will be adding post beta
- Cannot generate new emission points on a prefab in the inspector. However, users can edit a prefab that has been dragged into the scene in editor and apply changes to the prefab instead.

Future Improvements/ Features

- The ability to more accurately generate emission points from smaller textures
- The ability to more accurately generate emission points from textures tiled at a rate less than 1x1
- The ability to add multiple reference textures to a single object, allowing the user to emit more than one particle system from the one object. Essentially, one particle system per reference texture
- A cancel/ pause button for the ‘calculate spawn locations’ progress bar
- The ability to preview the emission accurately in edit mode (the same you will see as play mode)
- Skinned meshes will only show initial debug of generated locations to prevent system slowdowns/ crashes
- When using the emission type ‘rate over distance’, take into account distance moved during an animation at each emission point and emit particles accordingly
- Real-Time and GPU integration

Additional Help/ Contact

Feel free to contact us with any issues you may be having via any channel. We are always happy to support our customers and will address bug fixes as soon as possible. Please do not hesitate to contact us with feature requests either! We’d love to continue to make TexEmit better wherever possible.

Email: contact@2ginge.com
Website: www.2ginge.com
Twitter: @TwoGinge | @PezzSp | @JairMcBain

If you’d like to hear about our other projects and tools, please find our newsletter signup form at www.2ginge.com or check out our Unity asset store developer profile.