Hayate ~~and Nuke~~ particle system Reference – Particle turbulence simulation

Document Version: 1.2

Hayate Version: 1.3

Nuke Version: - Has been removed

The Hayate and Nuke particle system are designed to work with both Unity pro and free and as an add-on to the existing Shuriken particle system. Since Unity 3D's Shuriken particle system does not support turbulence effects the use of additional code is required to achieve effects as shown below:

Hayate

The choice of name of this particle modifier has been inspired by Unity's particle system Shuriken which is a throwing knife used by Ninjas. Hence the name Hayate also derives from Japanese Ninja terminology and means "cold breeze". This is also in dependence on what it does: Simulate wind / turbulence.

How it works:

Hayate takes the particles created by the shuriken system and recalculates their positions based on turbulence. Because of the new calculations following Shuriken components can't be used with this add-on:

- Collision (fixed in 1.2)

I'll try to fix this as soon as possible.

If you would like to report a bug, request a feature or contact me please feel free to use these email addresses: m4xproud@gmail.com

Youtube-video: http://youtu.be/P2T52J3dKoU

How to use:

Since Hayate is an Add-on to the Shuriken system you simply have to add the Hayate component to a GameObject that also has a Shuriken particle system. That's all! Now you have to tweak the Values to match your imagination.

Variables:

Debug Lines: Uses Unity's Debug.DrawLines() API to draw lines. Good to use for debugging.

Animate Offset: This changes the offset of the Perlin noise field to simulate gusts of wind.

Offset Speed: To change the speed of the animation change this value as needed. You usually want to use values between 0.001 and 0.01.

Particles follow emitter: This makes the particles return to the emitter's center. The further they are away, the less they will be affected by this.

Follow strength: This tweaks the strength of the emitter. Use this value to tweak the distance at which particles will return to the emitter.

Amplitude: This tweaks the scale of the distortion field on 3 Axis. The higher these values are, the smaller the turbulence field is. If it's smaller it will make the particles jump.

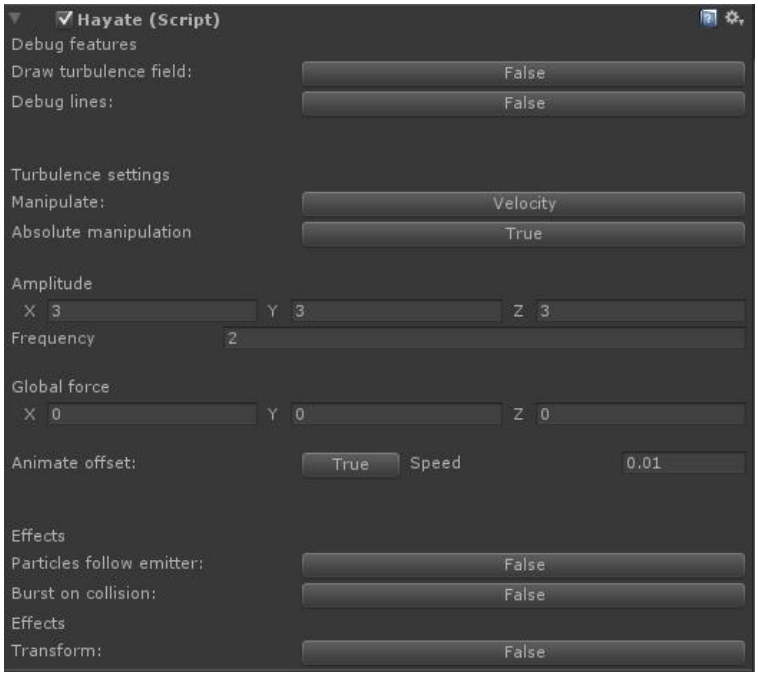
Global Force: This makes all particles move in the same direction specified here.

Frequency: Distortion tweaks the effect of the distortion field on the particles. Turn this up to make particles move further away. Be careful! The higher this value is, the faster the particles will go.

Offset: This is the offset as described above. If you like to use custom animations you can use this value to do so.

Burst On Collision: If your emitter has a collider attached to it, it will start to emit a certain amount of particles immediately.

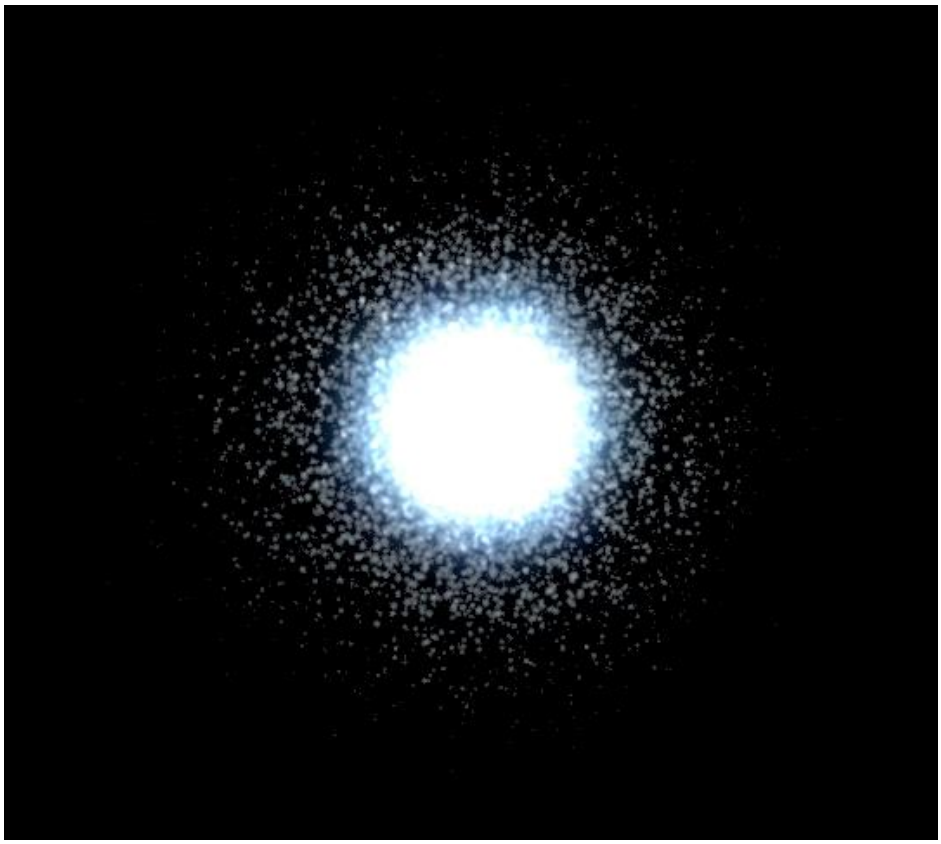
Burst Num: This is the amount of particles that will be spawned on collision.



For a demonstration please have a look at this video or check out the demo.

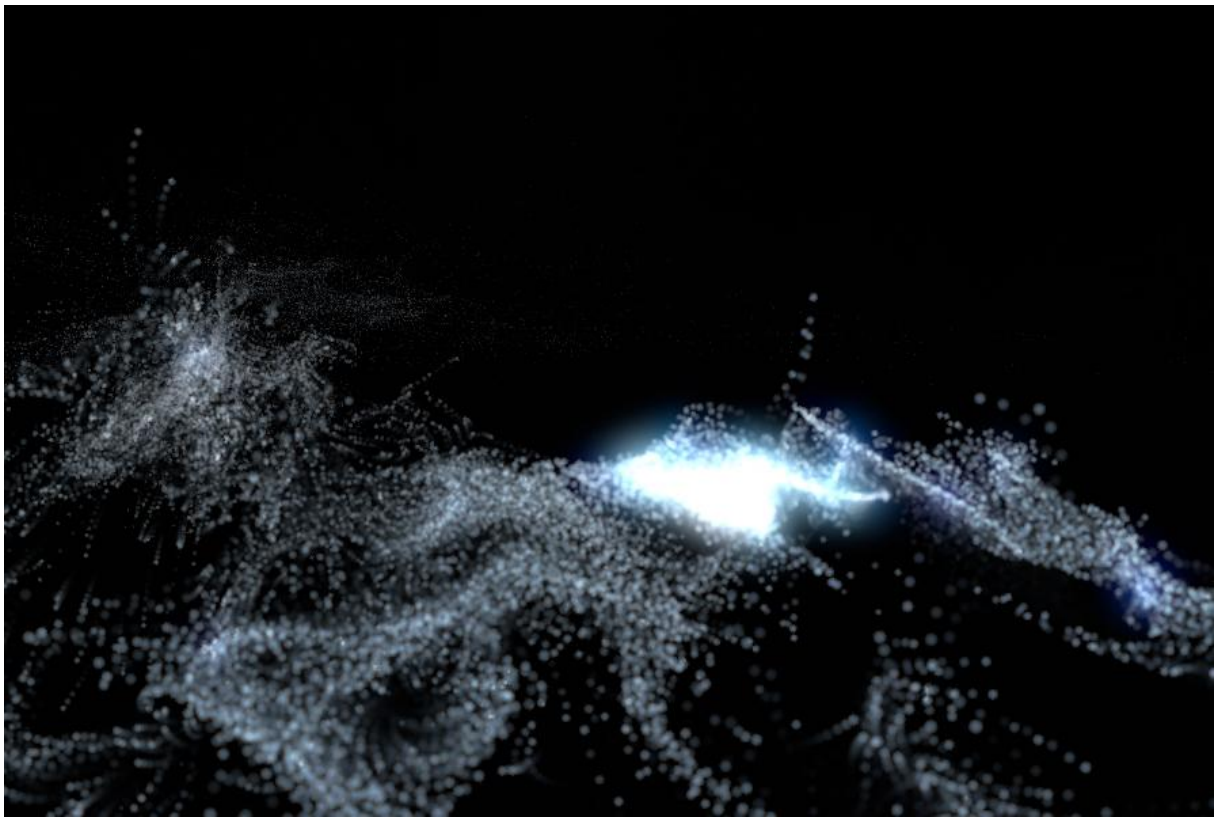
Sample images

Hayate deactivated – Normal Shuriken particle system



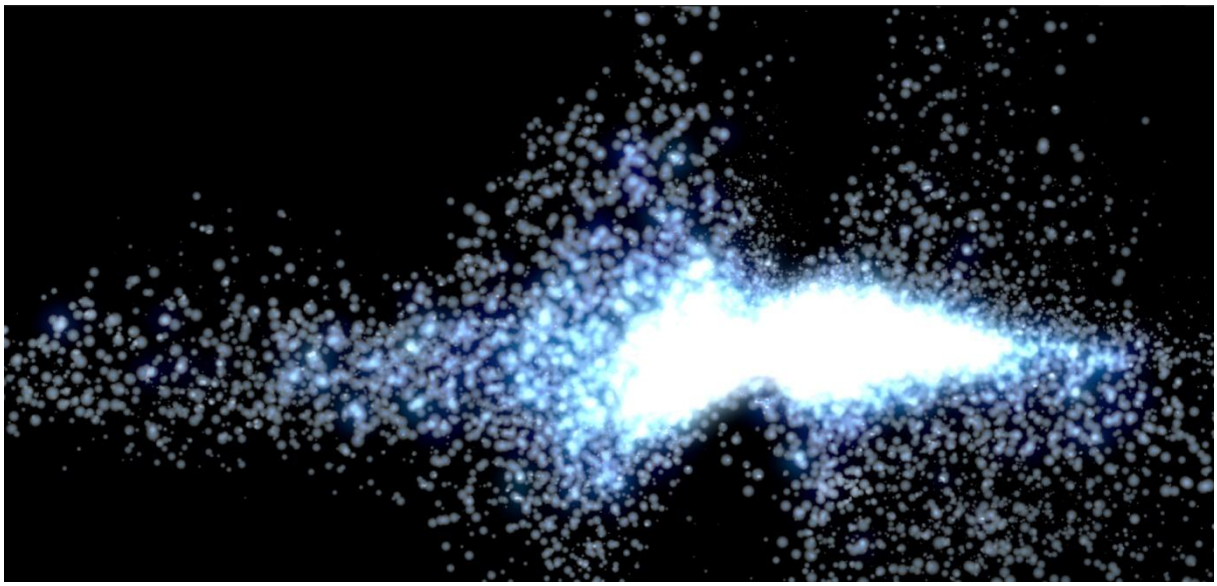
Hayate activated

Glow



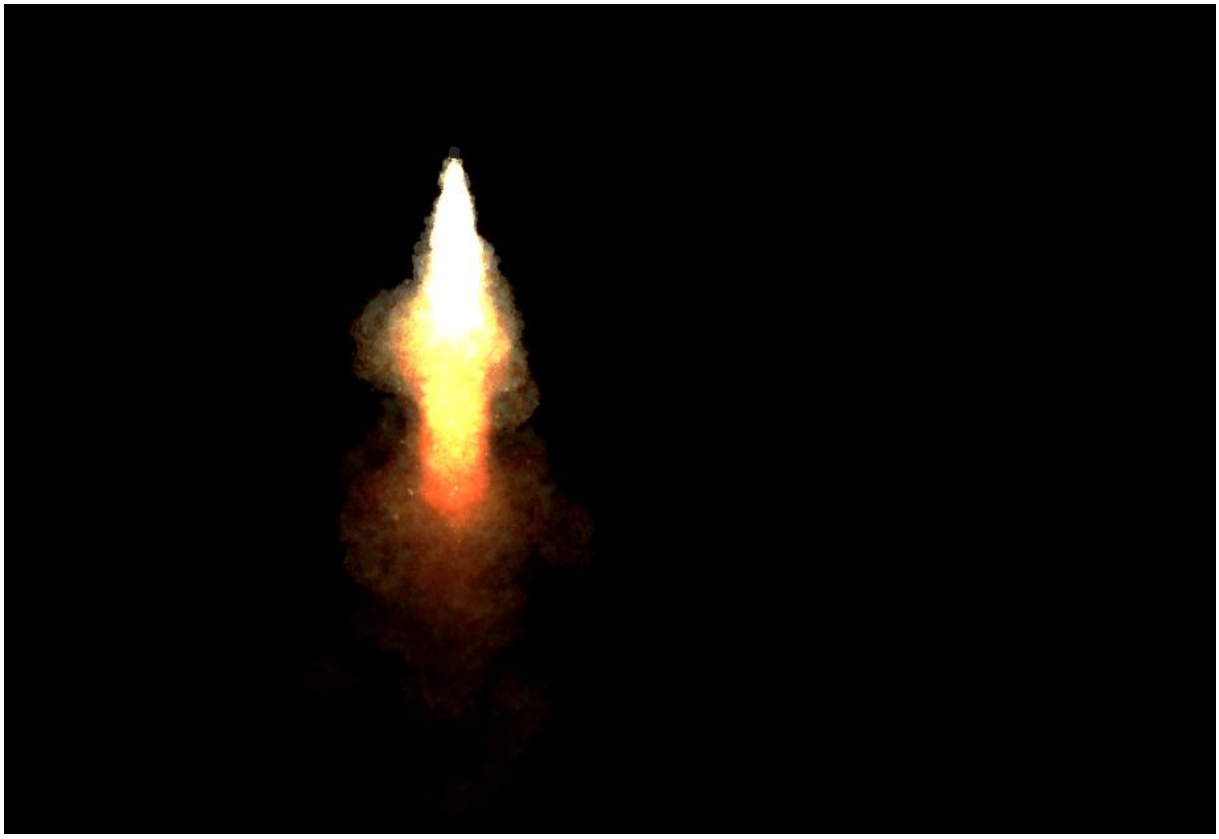
Motionblur + Glow

Glow

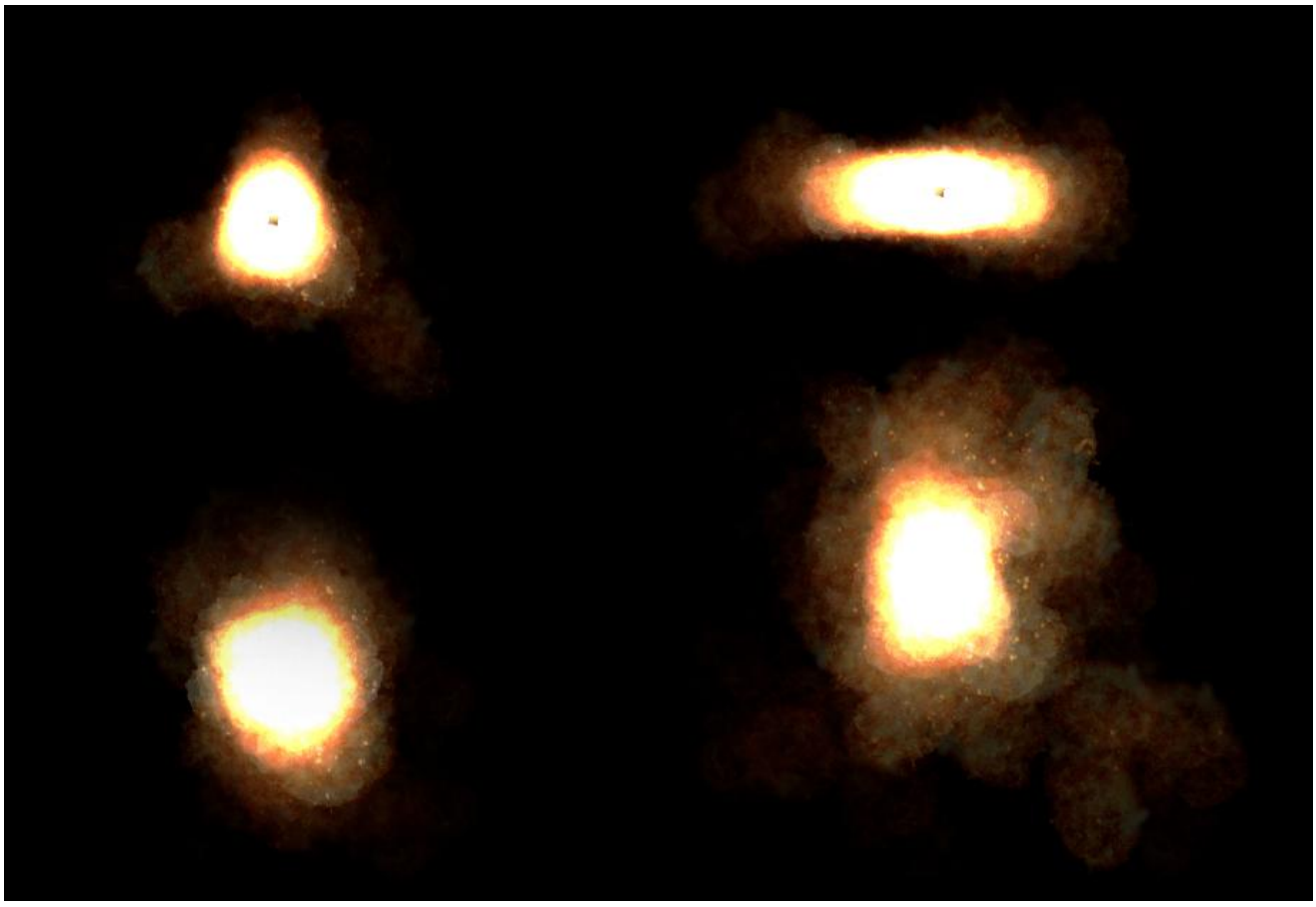


Glow

Each shot was taken at 20000 particles/s – lifetime: 1s @ 60+ FPS inside the unity editor.



The emitter is moving up. The turbulence field forces the particles into different directions

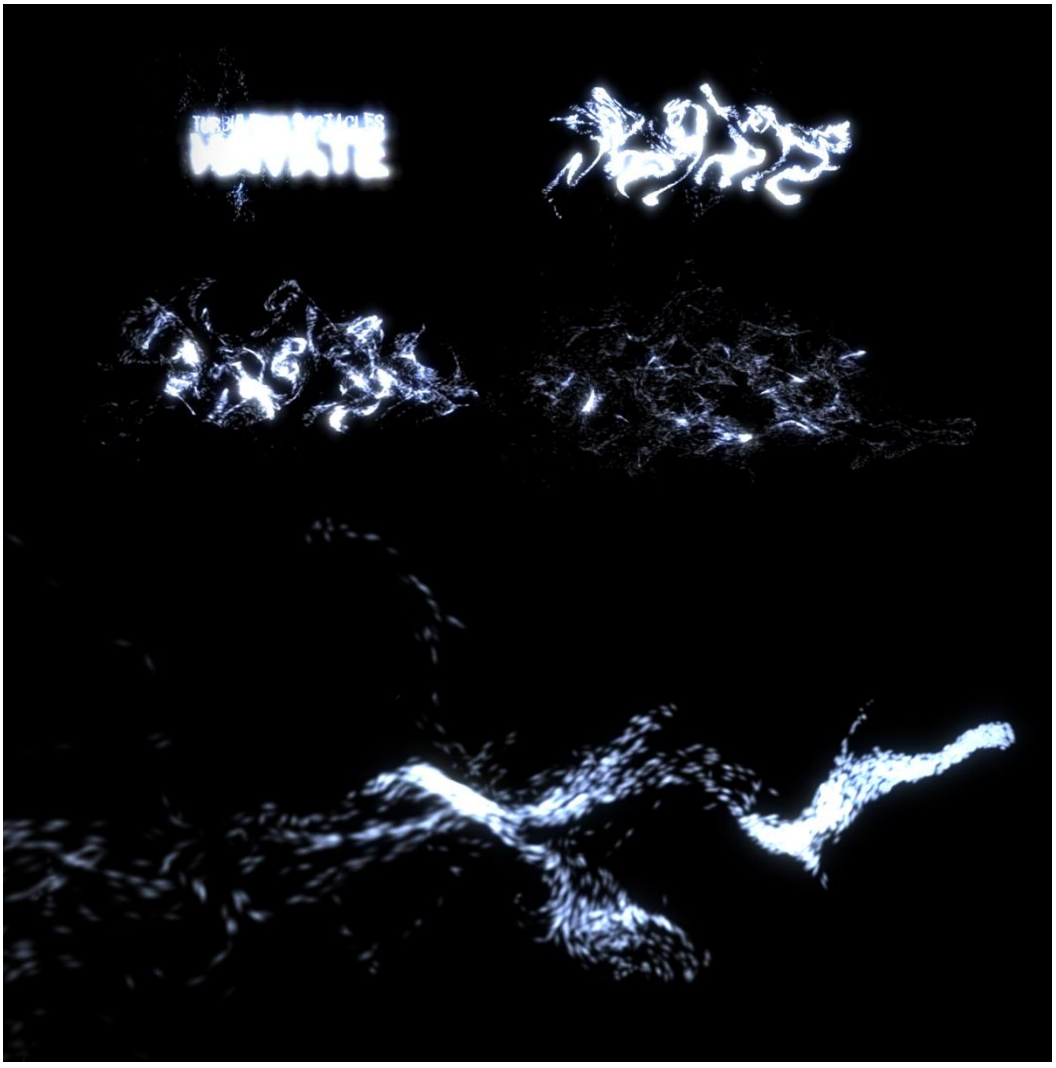


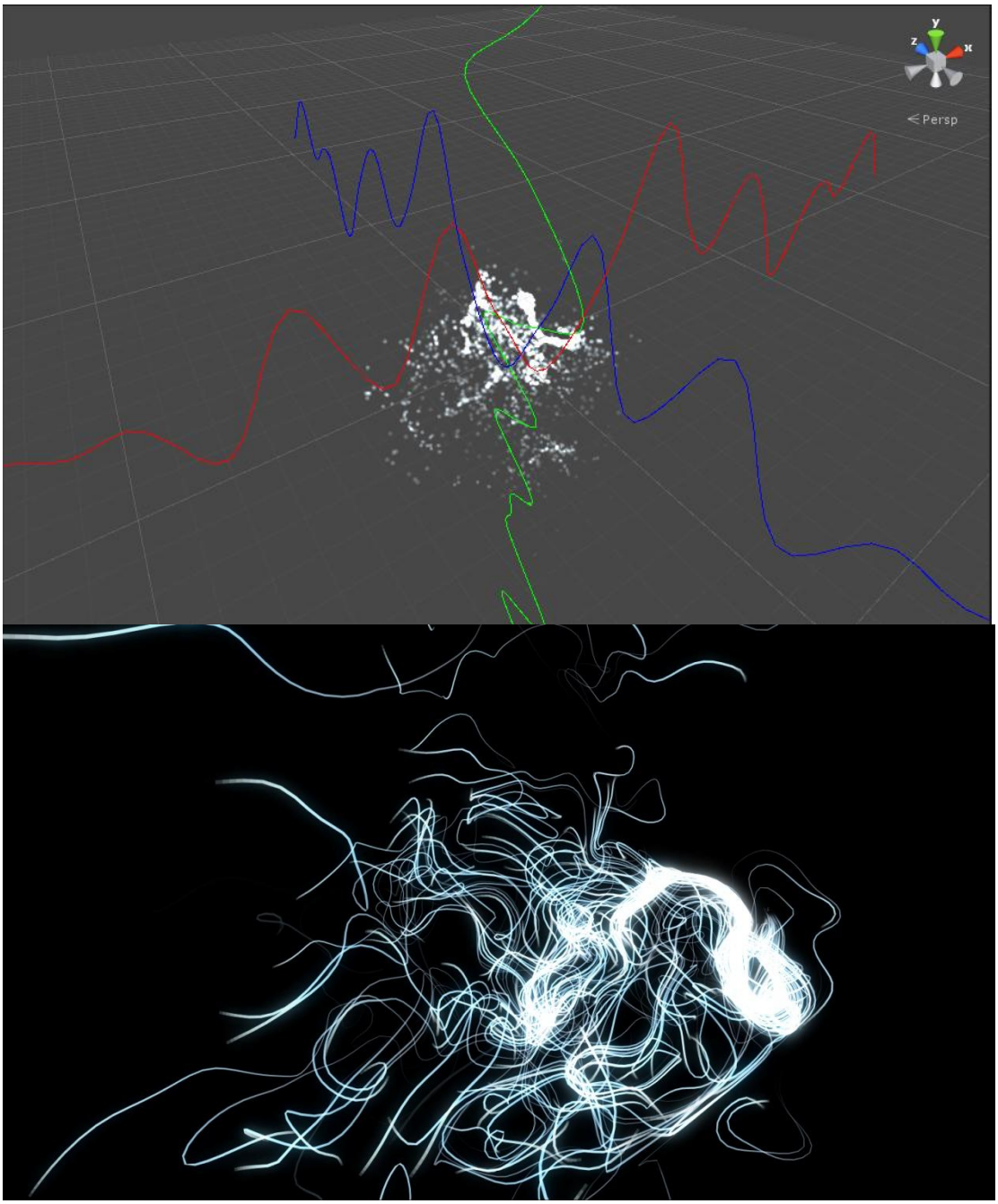
Animate more naturalistic fire only by animating the offset.

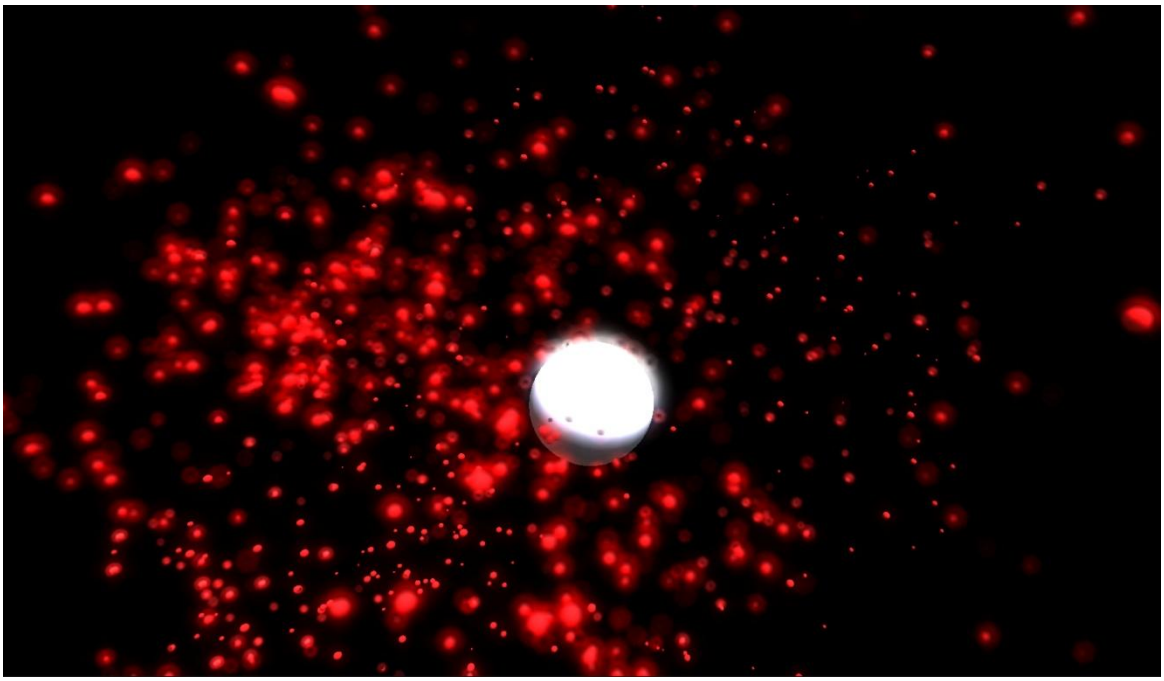
Update Notes:

Version 1.2

1. It is now possible to emit transform using Hayate. Just enable it in the editor and drag and drop a transform into the designated slot.
2. Hayate now is capable of visualizing the turbulence field on each axis (It is still necessary to start the game).
3. A custom editor has been added for easy usage.
4. Two new turbulence options are now available:
    a. Manipulate velocity relative
    b. Manipulate velocity absolute
5. Particle collision is now possible (Manipulate velocity relative).




y
z
x
Persp



Version 1.3

Version 1.3 brings a bunch of new features:

-Particle mesh target

-Partially support in editor mode

-Unity 3.5 support

-Autodestruction

-Random offset at start

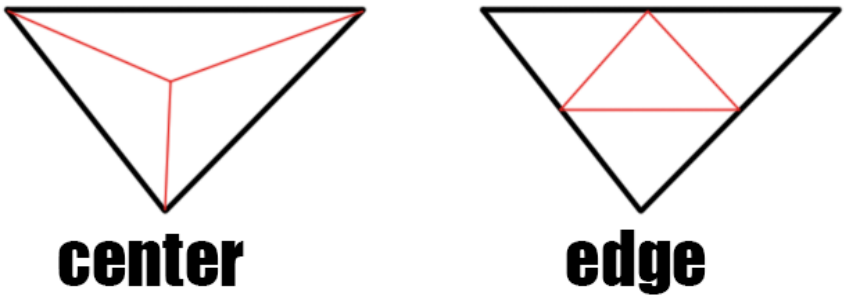
Mesh target

The biggest new feature is the new mesh target feature. This will allow you if enabled to animate particles towards the vertices of a mesh. To distribute the particles evenly or changing the density of a mesh a build in subdivision feature is also included. This will make every triangle as small as “smallest triangle”. Recommended workflow:

1. Enable mesh target in the menu.
2. Assign a Gameobject that includes a meshfilter. A mesh renderer is not required.
3. If you want to add more vertices to your mesh follow the next steps:
    a. Push the button “Retrieve mesh info”. This will tell you how big the smallest and the biggest triangle in your mesh are.
    b. Adjust the “smallest triangle” value accordingly. After the conversion every triangle will be smaller or euqual to this value.
    c. Choose a subdivision type (center or edge). This will affect the appearance of your mesh in different ways and one might look better than the other one. To find the right method for you will have to try a couple of times.
    d. Do you have a backup of your original file?

e. Push the button “Subdivide”. After the conversion a message will show up in the console and it will tell you how many particles are needed to fill the whole surface of your mesh.
f. Adjust the ‘max particles’ and ‘emission rate’ in Shuriken.

4. If you have a particle effect that fits your needs save both the mesh and the particle editor.



Note: Everytime you subdivide a mesh the reset button will only bring you back to the last mesh before subdivision. Changing the gameobject will cause the resetbutton to overwrite the mesh on this gameobject.

“Speed to mesh” is the speed at which each particle is interpolated towards its corresponding vertex. You can choose between two methods: “By distance” and “By time”.

By distance will accelerate the particles if they come closer to the mesh. This simulates some kind of gravity or magnetic field. It is possible that a particle will fly into the opposite direction and will never reach its destination if “speed to mesh” is too small. Once a particle reaches its destination it will most likely stay there.

By time will slow a particle down the closer it gets towards its destination. It is less likely that particles will not move towards its destination but if the speed is too slow it will take a lot of time. Increasing the speed will move them closer to their target. Turbulence will show the most effect in this type of interpolation.

(Linear interpolation will follow in the future)

Note:

The more vertices your mesh target has the more performance it will need. Use this feature wisely and rather make particles bigger than using more to fill a mesh without holes. See also Scene 7 in the demo.

Run in editor

Hayate now runs inside the editor. This might cause problems with the Transform particle feature because it will instantiate transforms. If you want to delete them simply push the “stop” button of your shuriken simulation tab (editor view).

Autodestruction

If this is enabled the particlesystem and its gameobeject will be destroyed if there are no more particles left and the animation does not loop.

Random offset at start

This will randomize the offset of the turbulence. Sometimes it is required to change / randomize the appearance of your animation especially if you instantiate a couple of them at the same time. This will prevent effects that look exactly the same.