WAWI ABDEL KADER
popcornfx.com

Bio
Wawi Abdel Kader is a VFX/tech artist at PopcornFX.

POPCORNFX

Create interactive fire and ice particles

PopcornFX is a dynamic, cross-engine, real-time FX particle solution middleware that can also be used for certain effects in offline rendering. PopcornFX is script based, which gives us the opportunity to change the behaviour of our particles as we please, not to mention its amazing optimisation features that come in addition to shape sampling, including spatial layers, distance fields, scene intersects, custom events and so on. We'll be making two separate effects: a fire emitter or a fire ball, and an ice emitter or an ice ball. The interesting part is that when the effects are within a certain radius, the fire ball will lose energy, become less intense and might even die.

So why is fire defeated by ice in this tutorial? Because winter is here!

01 Atlas builder for optimisation Since we'll be making two different effects, a fire and an ice ball, we'll need different textures. We'll use the atlas builder to put together textures that we need for both effects. This atlas will enable us to choose the particle texture based on an ID, without having to use different texture files. This way, if we use the same material in both effects along with the same texture, the rendering will batch them together, giving us less draw calls (more optimisation).

02 Select the appropriate material Choose the texture and the atlas definition, then set the material to AlphaBlend_Additive_Soft, a material that can be used as additive and as alpha blend (or multiply) depending on the alpha channel. So if the alpha in the Color that we set in PopcornFX is 1, then it's set to Alpha Blend/Multiply, and if it's 0 it's fully additive.

03 Set particles/initial values at spawn We'll make the emitter spawn infinitely and set the Spawn Count to 25. The spawner script is used to set the initial value when the particles spawn. We'll add a variation to the Size, Velocity and Life by multiplying them by a random value, and set the

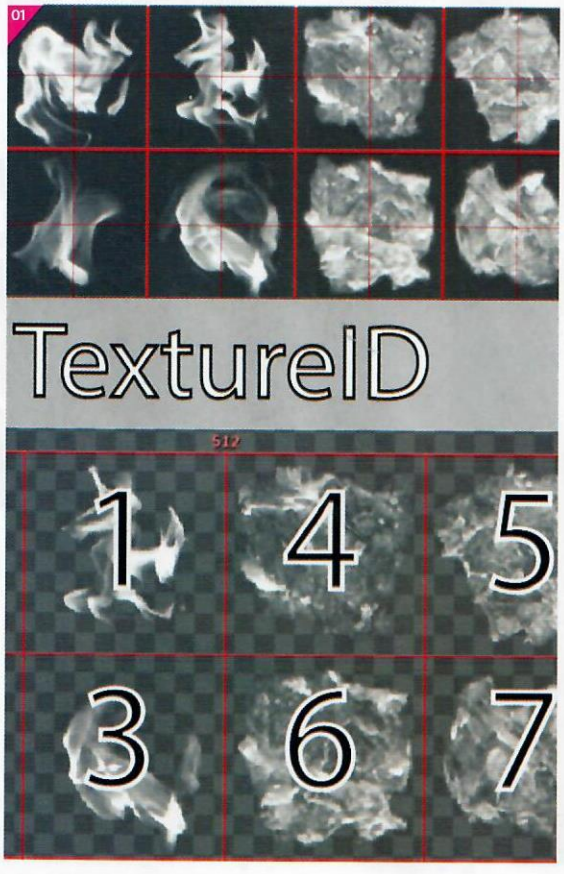
YOUR FREE DOWNLOADS
from filesilo.co.uk/3dartist
• Tutorial screenshots

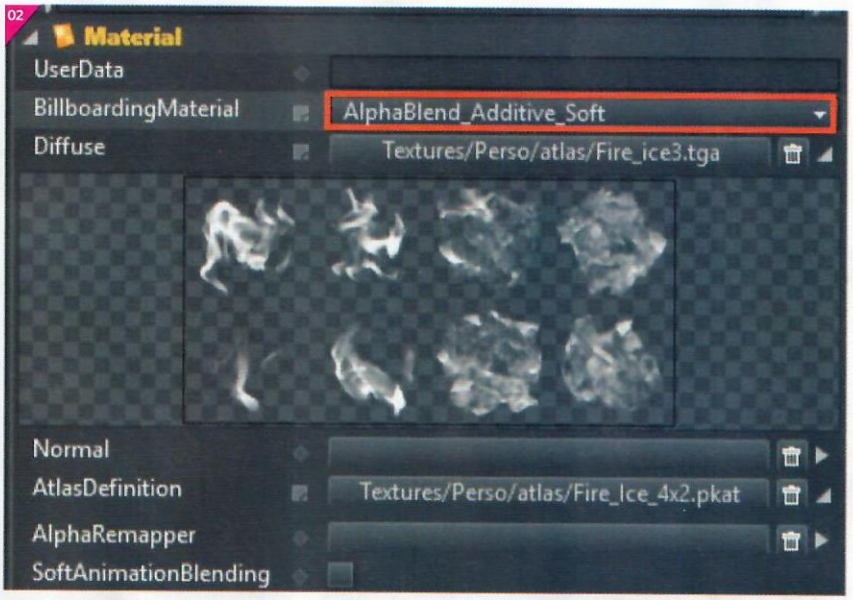
TextureID to a random value between 0 and 4. The randsel(-1,1) randomly selects either the first value or the second one; if it's negative it's going to mirror the texture.

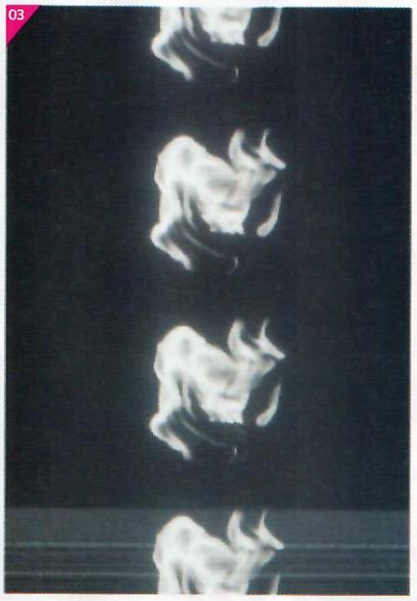
04 Add rotation and rotation speed We'll create a rotation evolver that creates two fields, Rotation and ScalarRotationSpeed, where Rotation controls the initial rotation and ScalarRotation controls the rotation speed in radians per second. We'll set the Rotation to rand(-1,1)*pi to have the initial rotation varying from -180 to 180 degrees, then the ScalarRotationSpeed to rand(-1,1)*pi*0.2 to have a variation of the Rotation Speed.

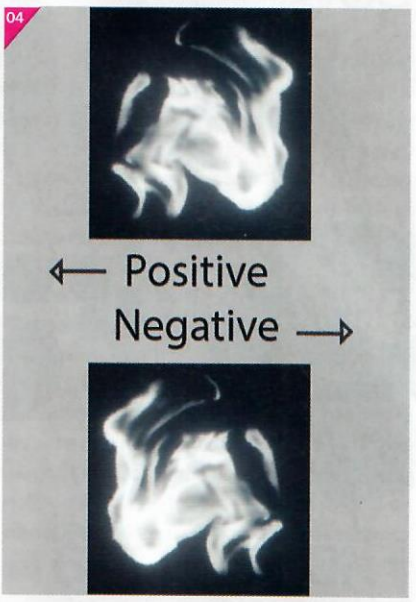
05 Add localspace and variation over time We need the particles to move with the emitter, so we'll add a localspace evolver. In most effects we need to change the values over the lifetime of the particle, so to do that we can use the following method to obtain a more dynamic effect. We start by creating an evolver field that lets us modify a field (let's name it ValueA, it can also be Size or Color) through a curve depending on the Life Ratio of the particle. And if we want the initial ValueA to vary, we can create a new field (let's name it ValueACoeff) in the spawner script and multiply it by the desired random values. Then multiply ValueACoeff by ValueA in the evolver script.

> PopcornFX is script based, which gives us the opportunity to change the behaviour of our particles as we please







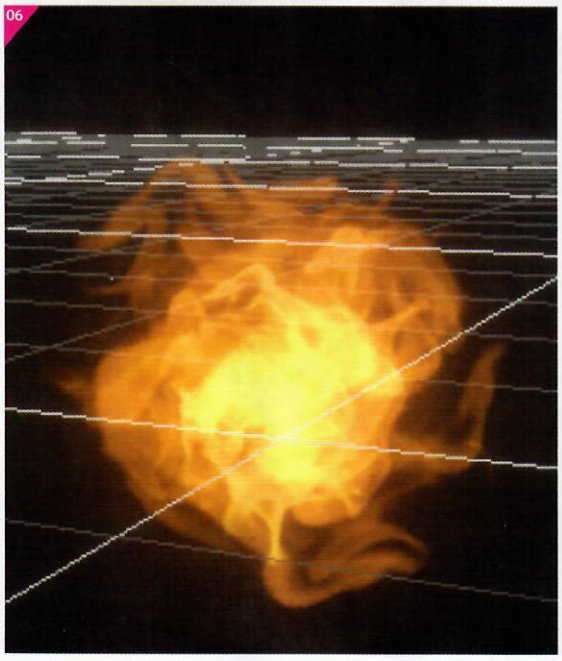


06 Make variations to size and colour We'll now replace Size in the spawner script by SizeCoeff (D) and add it to the fields, and in the evolver script write Size = Size*SizeCoeff; (E), which will take the Size value in the curve of the evolver field and multiply it with SizeCoeff. We can do the same with the Color to increase the colour intensity of the effect, or just have it changed in the Curve.

07 Spatial Layers and spatial insertion Spatial Layers are a PopcornFX special feature that lets particles interact with each other, passing on info like Color, Position, Velocity or any other custom field. We'll create the same Spatial Layer in the ice effect and the fire effect, name it 'FireVsIce' and set it to Global so that it is shared between the effects. Since the fire emitter effect will be the one that changes its behaviour, it means that these particles should be looking for the ice particles, so we're going to put the spatial insertion in the IceParticles.

08 Function of the closest We'll add the function spatialLayers.LayerName.FieldName.closest(float3 centre, float radius), where LayerName is FireVsIce, the FieldName is Position and then the centre should be Position. As for radius, it's going to be an attribute that we'll create, which is a variable exposed to the engine that can be modified through it. Next we add a condition to see if the particles are nearby. If it's not the case, nothing will change. If it is, we'll multiply their Size by the remapped value of the distance between them. This will reduce the Size of the particle.

09 Put the effects in the same scene The remapped Value is used to smooth the size variation, instead of it being abrupt. In this example we remapped the distance to a value between 0 and 1, to change the behaviour of the effect. We can add the Ice effect in the 3D Layers (World) of the fire effect, and then we can move the effects and see how they react to each other. You can also have fun with the variations. There are a lot of possibilities in PopcornFX that we can take advantage of, like mesh sampling a skull among many other features.




```
08 ion void  Eval()
float3 Spatial_Pos = spatialLayers.FireVsIce.Position.closest(Position,SpatialRadius);
float isClose= select (0,1,Spatial_Pos.x!=infinity);

float remappedVal =select (1,remap(length(Spatial_Pos-Position),0,SpatialRadius,0,1),isClose=
Size = Size*SizeCoeff*remappedVal;
```