Flow Noise

Contact
Ken Perlin
New York University
www.mrl.nyu.edu

Fabrice Neyret
IMAGIS

Flow textures defined with shaders that use Perlin Noise can look great, but they don't "flow" right, because they lack the swirling and advection of real flow. We extend Perlin Noise so that shaders that use it can be animated over time to produce flow textures with a "swirling"quality. We also show how to visually approximate advected flow within shaders.

Rotating Gradients

The classic Perlin Noise function[1] can be described as a sum of overlapping pseudo-random "wiggle" functions. Each wiggle, centered at a different integer lattice point (i; j; k), consists of a product of a weight kernel K and a linear function (a; b; c)i;j;k. K smoothly drops off away from (i,j,k), reaching 0 in both value and gradient at unit distance. Each (a; b; c)i;j;k = a(x_i)+b(y_j)+ c(z _ j) that has a value of 0 at (i,j,k). The result of summing all these overlapping wiggle functions: noise has a characteristic random yet smooth appearance.

Our modification is to rotate all the linear vectors (a; b; c)i;j;k over time, which causes each wiggle function to rotate in place. Because all the (a; b; c) vectors were uncorrelated before the rotation, they will remain uncorrelated after the rotation, so at every moment the result will look like Perlin Noise. Yet over time, the result will impart a "swirling" quality to flow. When multiple scales of noise are summed together, we make the rotation proportional to spatial frequency (finer noise is rotated faster), which visually models real flow.

Pseudo-Advection

Beyond swirling, fluids also contain advection of small features by larger ones, such as ripples on waves. This effect tends to stretch persistent features (for example, foam), but not newly created ones (for example, billowing), to varying degrees, according to their rate of regeneration, or structure memory M.

Traditional Perlin Turbulence is an independent sum of scaled noise, where the scaled noise is bi(x) = noise(2ix)=2i, and the turbulence is tN(x) =PN i=0 bi(x). This can define a displacement texture color(x) = C(x + ItN(x)), where C is a color table and I controls amplitude. Our pseudo-advection displaces features at scale i + 1 and location x0 in the noise domain to x1 = x0 + k ti(x0), where k is the amplitude of the displacement (see below). For small displacements, this can be approximated by x1_k ti(x1), so displacement k is proportional to an amplitude I specified by the user. We can scale this by desired structure memory M, since passive structures are totally advected, when M = 1, while very active structures are instantaneously generated, thus unstretched, when M = 0. Our advected turbulence function is defined by modifying the scaled noise to: bi(x) = b(2i(x_IM ti_1(x)))=2i and using this to construct the sum tN(x) = PN i=0 bi(x).

Results

Many flow textures can be created; some can be viewed at mrl.nyu.edu/flownoise/

Reference

1. Ebert, D. et al. (1998). *Texture and modeling*. Morgan Kaufmann Publishers, July 1998.

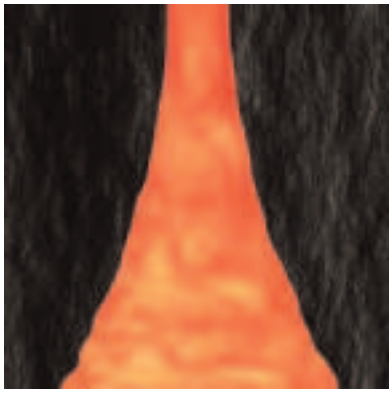

Figure 1. Lava flow.

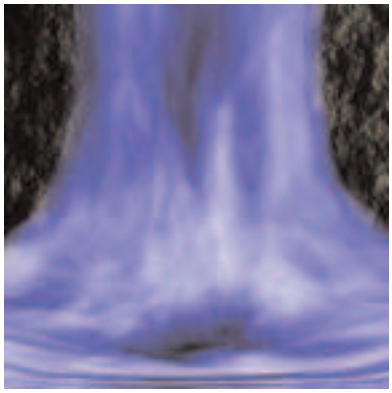

Figure 2. Waterfall.

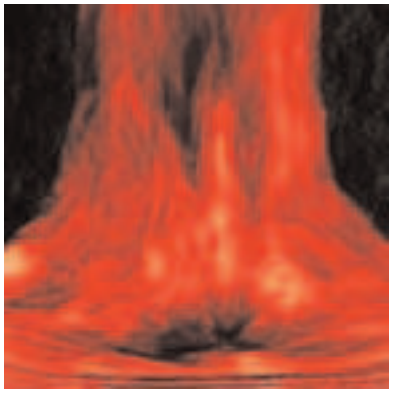

Figure 3. Waterfall with lava-flow texture.