DPA 819 Physically Based Effects/Wisp algorithm

From dpawiki
< DPA 819 Physically Based Effects

Algorithm for Generating Wisps

Pseudo code for the entire algorithm

This algorithm uses a uniform random number generator and two separate copies of fractal summed Perlin noise (FSPN). The random number generator has an input parameter - the seed. Each of the FSPN has a collection of parameters that you need to set independently in order to get the full range of visual options from the wisps.

```
for( loop over particles )
{
  // Initialize position of first dot
    x = 2.0*prn1.eval() - 1;
    y = 2.0*prn1.eval() - 1;
    z = 2.0*prn1.eval() - 1;
   for( loop over dots for this particle )
   {
   // correlated walk for the random position
   // First: random displacement step
   vx = 2.0*prn1.eval() - 1;
   vy = 2.0*prn1.eval() - 1;
   vz = 2.0*prn1.eval() - 1;
   // Second: update position with a correlation function
   x = corr * x + (1-corr) * vx
   y = corr * y + (1-corr) * vy
   z = corr * z + (1-corr) * vz

   // move position to unit sphere
   radius = sqrt( x*x + y*y + z*z );
   xsphere = x/radius;
   ysphere = y/radius;
   zsphere = z/radius;

   // displace radially from sphere using fractal sum
   radial_disp = pow(fabs(PerlinNoise1.eval( Vector(x, y, z) )), clump ); // Perlin noise using wisp-named
   xsphere *= radial_disp;
   ysphere *= radial_disp;
   zsphere *= radial_disp;

   // map to guide particle coordinate
   xdot = xparticle + xsphere * pscale;
   ydot = yparticle + ysphere * pscale;
   zdot = zparticle + zsphere * pscale;

   // displace again with 3D fractal sum noise
   // generates three noise values by displacing position of evaluation
   dx = 0.1;
   xfsn = PerlinNoise2.eval( Vector(xsphere, ysphere, zsphere) ) * wisp_displacement;
   yfsn = PerlinNoise2.eval( Vector(xsphere + dx, ysphere + dx, zsphere + dx) ) * wisp_displacement;
   zfsn = PerlinNoise2.eval( Vector(xsphere - dx, ysphere - dx, zsphere - dx) ) * wisp_displacement;
   xdot += xfsn;
   ydot += yfsn;
```

```
      zdot += zfsn;

      // Now ready to bake a dot into the volume at (xdot, ydot, zdot)
      BakeDot( xdot, ydot, zdot, particle_density, particle_color );
   }
}
```