

Pixel Shader Code

(abbreviated)

```
//Declare pixel shader version
ps.2.0

//g f1(electrical potential)      b      f2(ion concentration)

texld    r0, t0, s0      //current cell
texld    r1, t1, s1      //4 texture coord. to sample 8 neighbors
texld    r2, t2, s1      //by means of bilinear filtering
texld    r3, t3, s1
texld    r4, t4, s1
texld    r6, t6, s6      //random number texture

mul      r8, r0, c4      //rescale center from [0.0, 1.0] into r8
add      r8, r8, c5

add      r1, r1, r2
add      r1, r1, r3
add      r1, r1, r4      //r1 = sum straight/diagonal neighbors

mul      r1, c2, r1      //scale straight/diagonal neighbors

mad      r0, c1, r0, r1 //update center (averaged due to diff.)

mul      r0, r0, c4      //rescale updated center from [0.0, 1.0]
add      r0, r0, c5      //finished second part of equation here

//f1
sub      r4.g, c0.r, r8.g      //(a-f1)
sub      r7.g, r8.g, c3.r      //(f1-1)
mul      r4.g, r4.g, r7.g      //(a-f1)*(f1-1)
mad      r4.g, r4.g, r8.g, -r8.b  //(a-f1)*(f1-1)*f1-f2

//f2
mad      r4.b, c0.g, r8.g, -r8.b  //(b*f1-f2)
mul      r4.b, c0.b, r4.b      //e*(b*f1-f2)
//finished first part of equation here

add      r0, r0, r4      //add first part of equation

mul      r0, r0, c6      //rescale to [0.0, 1.0]
add      r0, r0, c7

mad      r6, r6, c3.g, c3.b // [0.0, 1.0] => [-0.5/256, 0.5/256]
add      r0, r6, r0      //add for stochastic rounding

mov      oC0, r0      //output: new cell state; automatic rounding
```