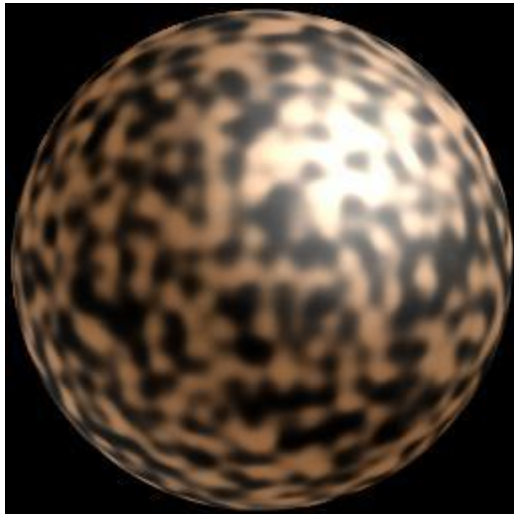
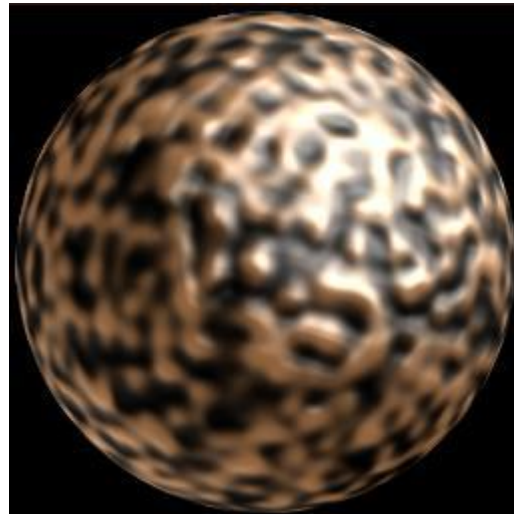


# Improved Noise reference implementation

Ken Perlin



Smooth ball demo



Bumpy ball demo

This code implements the algorithm I describe in a corresponding [SIGGRAPH 2002 paper](#).

// JAVA REFERENCE IMPLEMENTATION OF IMPROVED NOISE - COPYRIGHT 2002 KEN PERLIN.

```
public final class ImprovedNoise {
    static public double noise(double x, double y, double z) {
        int X = (int)Math.floor(x) & 255,          // FIND UNIT CUBE THAT
        Y = (int)Math.floor(y) & 255,              // CONTAINS POINT.
        Z = (int)Math.floor(z) & 255;
        x -= Math.floor(x);                          // FIND RELATIVE X,Y,Z
        y -= Math.floor(y);                          // OF POINT IN CUBE.
        z -= Math.floor(z);
        double u = fade(x);                          // COMPUTE FADE CURVES
        v = fade(y);                                  // FOR EACH OF X,Y,Z.
        w = fade(z);
        int A = p[X ]+Y, AA = p[A]+Z, AB = p[A+1]+Z, // HASH COORDINATES OF
        B = p[X+1]+Y, BA = p[B]+Z, BB = p[B+1]+Z;    // THE 8 CUBE CORNERS,

        return lerp(w, lerp(v, lerp(u, grad(p[AA ], x , y , z ), // AND ADD
        grad(p[BA ], x-1, y , z )), // BLENDED
        lerp(u, grad(p[AB ], x , y-1, z ), // RESULTS
        grad(p[BB ], x-1, y-1, z )), // FROM 8
        lerp(v, lerp(u, grad(p[AA+1], x , y , z-1 ), // CORNERS
        grad(p[BA+1], x-1, y , z-1 )), // OF CUBE
        lerp(u, grad(p[AB+1], x , y-1, z-1 ),
        grad(p[BB+1], x-1, y-1, z-1 ))));
    }
    static double fade(double t) { return t * t * t * (t * (t * 6 - 15) + 10); }
    static double lerp(double t, double a, double b) { return a + t * (b - a); }
    static double grad(int hash, double x, double y, double z) {
        int h = hash & 15;                          // CONVERT LO 4 BITS OF HASH CODE
        double u = h<8 ? x : y,                      // INTO 12 GRADIENT DIRECTIONS.
        v = h<4 ? y : h==12 ? x : z;
        return ((h&1) == 0 ? u : -u) + ((h&2) == 0 ? v : -v);
    }
    static final int p[] = new int[512], permutation[] = { 151,160,137,91,90,15,
    131,13,201,95,96,53,194,233,7,225,140,36,103,30,69,142,8,99,37,240,21,10,23,
    190, 6,148,247,120,234,75,0,26,197,62,94,252,219,203,117,35,11,32,57,177,33,
    88,237,149,56,87,174,20,125,136,171,168, 68,175,74,165,71,134,139,48,27,166,
    77,146,158,231,83,111,229,122,60,211,133,230,220,105,92,41,55,46,245,40,244,
    102,143,54, 65,25,63,161, 1,216,80,73,209,76,132,187,208, 89,18,169,200,196,
    135,130,116,188,159,86,164,100,109,198,173,186, 3,64,52,217,226,250,124,123,
    5,202,38,147,118,126,255,82,85,212,207,206,59,227,47,16,58,17,182,189,28,42,
    223,183,170,213,119,248,152, 2,44,154,163, 70,221,153,101,155,167, 43,172,9,
    129,22,39,253, 19,98,108,110,79,113,224,232,178,185, 112,104,218,246,97,228,
    251,34,242,193,238,210,144,12,191,179,162,241, 81,51,145,235,249,14,239,107,
    49,192,214, 31,181,199,106,157,184, 84,204,176,115,121,50,45,127, 4,150,254,
    138,236,205,93,222,114,67,29,24,72,243,141,128,195,78,66,215,61,156,180
    };
    static { for (int i=0; i < 256 ; i++) p[256+i] = p[i] = permutation[i]; }
}
```