COS 452
Assignment 4, Part 1
Tyler LaBerge

Perlin noise is a type of gradient noise algorithm that was developed by Ken Perlin in 1983. It is the foundation of many procedural texture and modelling algorithms and is used to produce realistic, natural looking computer graphics. For example things like clouds or fire can be modelled realistically with the Perlin noise algorithm.

The algorithm for Perlin noise put simply is as follows. First, given an input coordinate calculate the unit gradient vectors for each of the corners of the input coordinates surrounding unit square. Next, calculate the distance vectors from the given input coordinate to each of the surrounding corners on the unit square. Next, calculate the influence values by taking the dot product between each of the gradient vectors and the corresponding distance vector. Next, interpolate the values between each of the calculated dot products to get an averaged influence value. Finally, apply an ease function to make the interpolated value appear more natural. The final value will be between 0 and 1 and is the calculated noise value of the given coordinate using the Perlin noise algorithm. By applying this algorithm to many coordinates you will generate an array of Perlin noise.

The “noisiness” or randomness of the generated noise is determined by the frequency. A higher frequency will make the generated noise more random, and a lower frequency will make the generated noise less random. You can increase the frequency simply by multiplying each input coordinate to the perlin noise function by a scalar value. The higher the scalar multiple the higher the frequency of the generated noise. The wavelength is the inverse of frequency, and so by increasing the frequency you will decrease the wavelength, which means the waves of the outputted noise values will decrease in length.

The way the fireball example works is by sending each vertex coordinate in the fragment shader to a function called heightMap. This function calls the snoise function which calculates a value for the coordinate using the simplex noise algorithm. The heightMap function acts as a smoothing function by calling the simplex noise function multiple times using different frequencies on the input. Each frequency is double the previous one. So we have snoise(coord * 1), snoise(coord *2), snoise(coord*4) etc. up to 16. The heightMap function sums the return values together and returns that value back to the main function, thus the reason heightMap is a smoothing function. It is essentially layering simplex noise of different frequencies on top of one another to produce smooth noise. The color of each vertex is based on the noise of that vertex and is altered to fall within the yellow-red range. There are some other aspects in the fragment shader, such as lighting and so forth but the main aspect is just in generating the smooth noise and simplex noise using the heightMap function and snoise function, and then coloring the resulting value accordingly. Applying this to each vertex creates the fireball looking effect.

Sources:
http://www.redblobgames.com/articles/noise/introduction.html#frequency
http://flafla2.github.io/2014/08/09/perlinnoise.html
https://en.wikipedia.org/wiki/Perlin_noise