Brett Layman

CSCI 441

4/26/2018

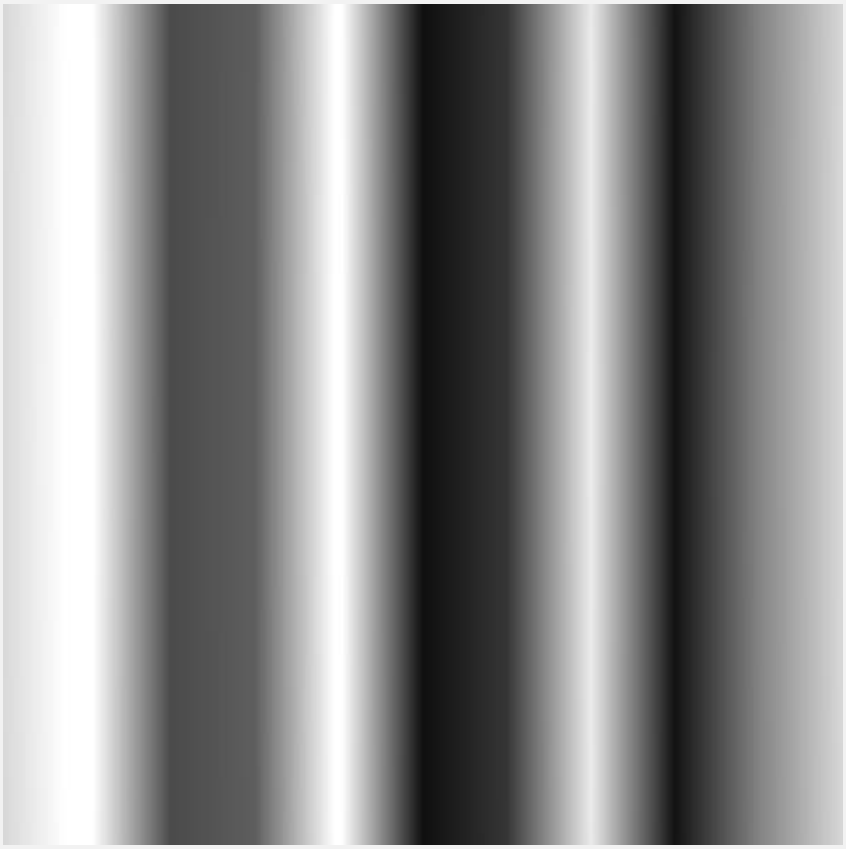
Program 3 Extra Feature Description

For my plus one feature, I chose to experiment with different types of procedural noise texture generation. Most of what I learned about these techniques came from the tutorials on this site:

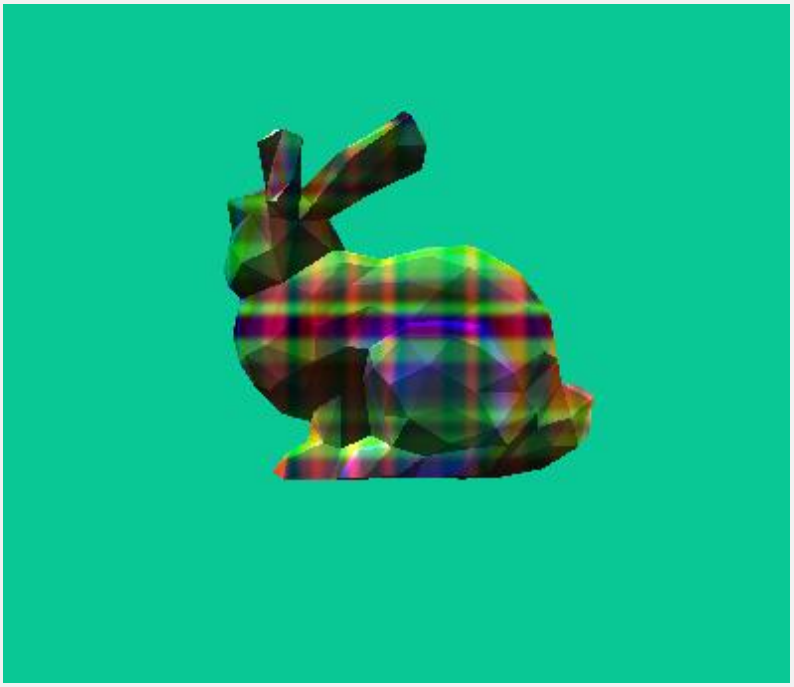
https://www.scratchapixel.com/lessons/procedural-generation-virtual-worlds/procedural-patterns-noise-part-1/introduction

My code is described in the README file, so look there for implementation details. Here are the techniques I implemented:

One Dimensional Linear Interpolation of Random Numbers



This technique generates an array of random values with values between 0 and 1, and uses linear interpolation and cosine smoothing to generate the gradients between values. Integer inputs to the sampling function correspond to the generated random numbers, while the decimal values in between integers are where interpolation occurs. In this example, I used the x value of my ray intersection position with a cube (here showing the flat front surface) as input for sampling the random numbers. I scaled my inputs by a factor of 7 to achieve this result.

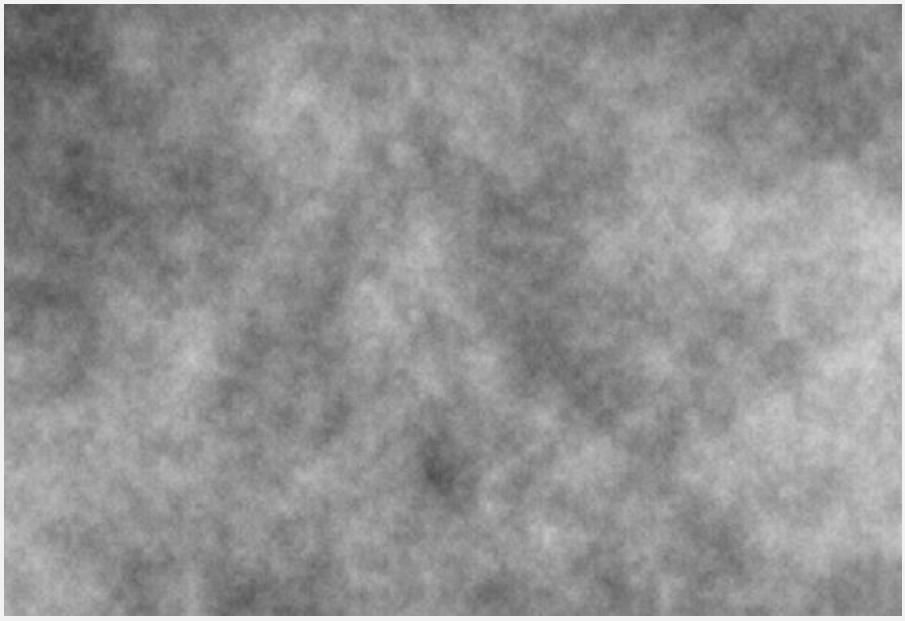


In this case, I used one dimensional linear interpolation of random values with different colors for different axes. Red is varied along the x axis, green along the y, and blue along the z axis. This is combined with Phong shading to get the above result.

Two Dimensional Linear Interpolation of Random Numbers

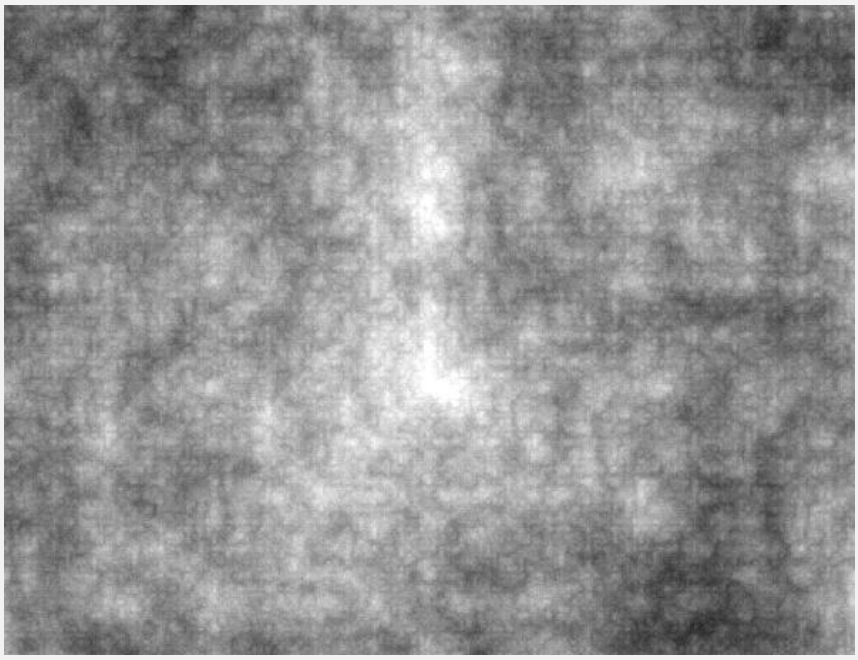
Like the one dimensional technique, this also uses linear interpolation and cosine smoothing. However, it interpolates values in a 2D grid across the x and y axes. This is done by interpolating between the 4 corner integer values that surround the ray intersection point (using the x and y coordinates).

Fractal Layering of 2D Noise



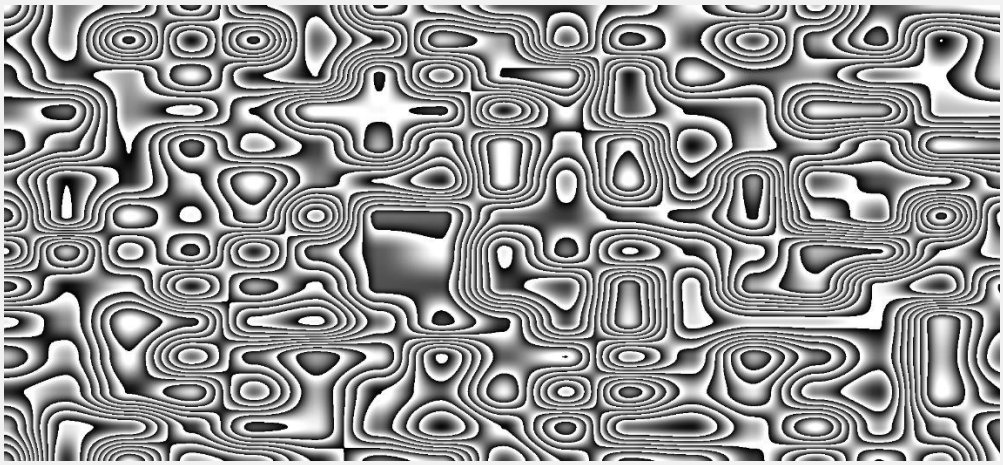
This technique uses the same 2D noise distribution as the previous technique, but instead of just sampling it once, it's called repeatedly to obtain and layer multiple samples. The results of each sample are divided by the number of samples, and summed to create a cumulative effect of all the samples. Importantly, each sample has a different scale, with the scale decreasing 2-fold each iteration (or by some other multiplier). I also tried decreasing the amplitude as the scale became smaller, which creates a softer looking result, like the one above, and required using a higher base amplitude.

Turbulence

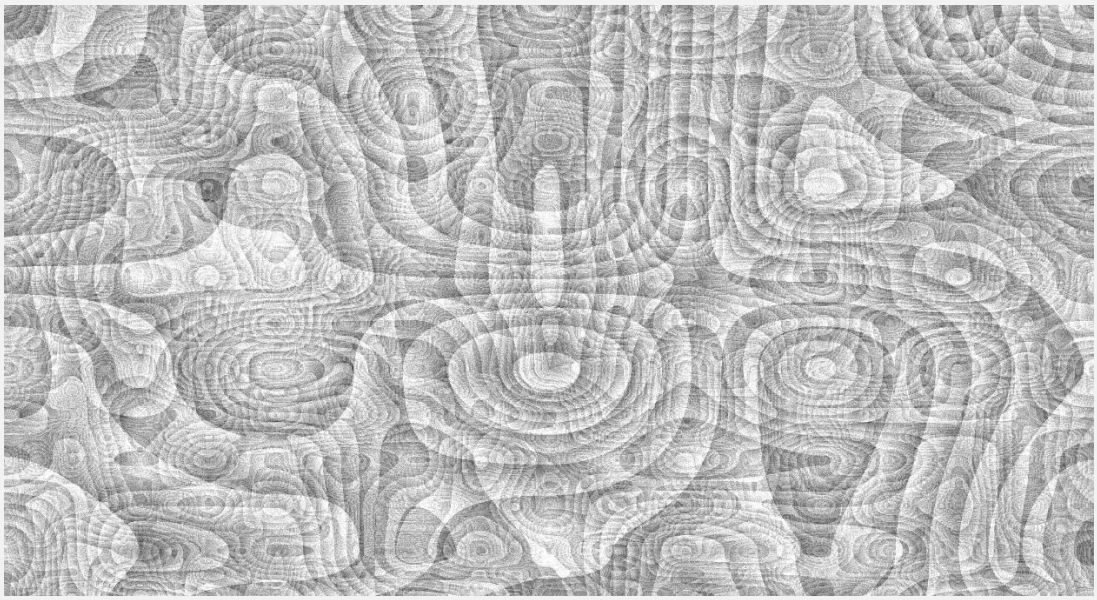


This effect is created by using the same approach as the previous one, using fractal layering, but with a twist. The noise function is modified by a few operations. First it is shifted downward so that some of the curve becomes negative, then it is doubled so that the positive portion is in the range [0,1], and finally the absolute value is taken. This creates dark snaking edges and an effect that looks a little like smoke.

Wood Grain



This is another technique that uses and builds on the 2D distribution of noise. In this case, the noise is transformed into topographic style bands. This is achieved by multiplying the noise function by a scalar, such as 10, and then using the fraction remaining after subtracting the integer portion of that number as the color value. In effect, whenever the function reaches a new integer, it resets, and a new band is started. The above image shows this technique applied to one layer of noise, while the image below shows multiple layers.



For more images, see my “SavedImages” zip file.