Better Faster Noise with the GPU

*Geoff Wyvill
University of Otago

†Jeppe Revall Frisvad
Technical University of Denmark

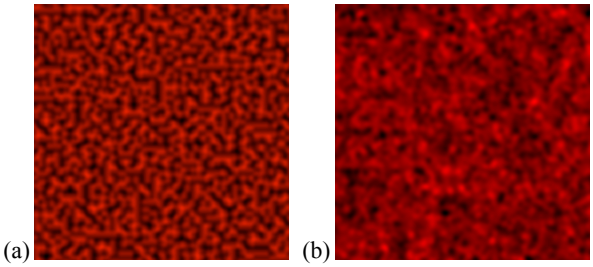
1 Background

Filtered noise [Perlin 1985] has, for twenty years, been a fundamental tool for creating functional texture and it has many other applications; for example, animating water waves or the motion of grass waving in the wind.

Perlin noise suffers from a number of defects and there have been many attempts to create better or faster noise but Perlin's 'Gradient Noise' has consistently proved to be the best compromise between speed and quality. Our objective was to create a better noise cheaply by use of the GPU.

2 Reference Noise

Filtered noise is a continuous, locally smooth function that is random in the sense that values more than a certain distance apart will be uncorrelated. We would also like it to be isotropic in a statistical sense so if we trace the function along a line its appearance doesn't depend on the direction of the line. Perlin's gradient noise has discernible axis aligned artifacts and the very non-random property that the value is zero at every point of the rectangular grid used to define the function. You can disguise these defects by adding noise values from non-aligned grids but then it is not so cheap.



(a) (b)

Figure 1: (a) Perlin noise [2002] (b) Reference noise

We started by creating an expensive noise of guaranteed quality, free from directional or grid-based artifacts. We convolve a set of randomly placed seed values with a Gaussian filter function. We call this reference noise and it represents a standard against which we compared other functions. As well as being expensive, the reference noise offers us no simple way to calculate it on-the-fly. We have to generate the whole field of pseudo-random seed values because we have no method of generating only the ones in the locality of our sample point.

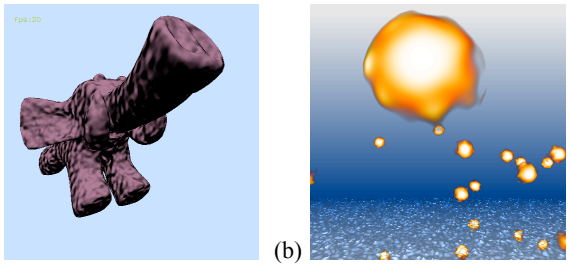
3 Implementation

We implemented various functions based on recursive space division and irregular grids to place pseudo-random seed in 'predictable' places but in terms of visual similarity to the reference, we have found nothing better than the Sparse Convolution Noise of J.P. Lewis [1989]. It does not have the directional or grid based artifacts familiar in Perlin noise and it is almost as good as our reference noise.

We create the Sparse Convolution Noise point by point on-the-fly in a fragment shader using a Spot Noise similar to van Wijk [1991]. We use alpha blending to combine the effect of spots but calculate the contribution from each kernel in the fragment shader using the spot as the centre of that kernel. We can also do this in three dimensions using a 3D real texture to supply the eyespace coordinates to the fragment shader. And it is fast enough to create solid noise based textures in real time.

4 Results

Figure 2a shows a bump-mapped elephant that uses a 3D solid noise texture. This can be rotated interactively and renders at about 20 frames/sec using an old Nvidia Quadro FX Go1400; 2b shows a frame from an animation of fireballs falling into the sea. The fire and sea motion are generated with the GPU noise.



(a) (b)

Figure 2: (a) Bump mapped elephant (b) Falling fireballs

We have also tested our code on an Nvidia 8800 GTX and the Elephant renders at 40 frames/sec. But we have not yet adapted the code to use the programmable parallel processors or special arithmetic of the 8800 series and we believe it can be made very much faster.

5 References

PERLIN, K. 1985. An image synthesizer. In *Computer Graphics (Proceedings of ACM SIGGRAPH '85),* 19,3, ACM, 287-296.

PERLIN, K. 2002. Improving noise. *ACM Transactions on Graphics,* 21, 3, 681-682.

LEWIS, J. P. 1989. Algorithms for solid noise synthesis. In *Computer Graphics (Proceedings of ACM SIGGRAPH '89),* 23, 3, ACM, 263-270.

VAN WIJK, J. J. 1991. Texture synthesis for data visualization. In *Computer Graphics (Proceedings of ACM SIGGRAPH '91),* 25, 4, ACM, 309-318.

*e-mail: geoff@otago.ac.nz
†e-mail: jrf@imm.dtu.dk