Designer Worlds: Procedural Generation of Infinite Terrain from USGS Elevation Data

Ian Parberry

Technical Report LARC-2013-02

Laboratory for Recreational Computing
Department of Computer Science & Engineering
University of North Texas
Denton, Texas, USA

August, 2013

Designer Worlds: Procedural Generation of Infinite Terrain from USGS Elevation Data

Ian Parberry
Department of Computer Science & Engineering
University of North Texas
Denton, TX, 76203–5017
URL: http://larc.unt.edu/ian

***Abstract*—The standard way to generate random terrain for video games and other applications is to post-process the output of a fast noise generator such as Perlin noise. This requires game designers to be reasonably well-trained in mathematics. We describe how a well-known variant of Perlin noise called *value noise* can be used without post-processing to generate terrain with varying characteristics such as mountains, rolling hills, and plains. The design process uses elevation data from the USGS and can be used easily by a designer who is trained in geography rather than mathematics.**

***Index Terms*—Procedural content generation, terrain generation, height map, Perlin noise, value noise, geospatial analysis.**

I. Introduction

Games such as Minecraft [1] generate the visible portion of a potentially infinite terrain in real time. Real time means potentially being able to calculate heights for around 10^5 points in a 1/60th of a second frame of animation. As noted in Smelik et al. [2], terrain generation is often based on fractal noise generators such as Perlin noise [3], [4] (for more details see, for example, Ebert et al. [5]).

Terrain generation is an example of what is known in the game industry as *procedural content generation*. Procedural content generation needs to have three important properties. Firstly it needs to be fast, meaning that it has to use only a fraction of the computing power on a current-generation computer. Secondly, it needs to be both random and structured so that it creates content that is varied and interesting. Thirdly, it needs to be controllable in a natural and intuitive way.

While Perlin noise is fast, it lacks somewhat in that it tends to create terrain that is uniform and boring, often requiring significant post-processing to add interesting features. It is also not intuitive to control for a design professional who is not mathematically trained. We describe how to use a version of Perlin noise called *value noise* [6], [7], [8], [9] for the procedural generation of terrain data for use in a video game or terrain simulator. We further perform a spatial analysis of elevation data for the state of Utah from the United States Geological Survey [10], and show how the results can be easily integrated with value noise to generate random terrain that shares height characteristics with places in the real world.

The main part of this paper is divided into four sections. Section II contains an overview of value noise. and describes how to use it to generate random terrain. Section III describes a spatial analysis of GIS data. Section IV shows how to incorporate the GIS data from Section III into the value noise algorithm. More information about the subject of this paper, including high-resolution images of generated terrain, can be found in Parberry [11].

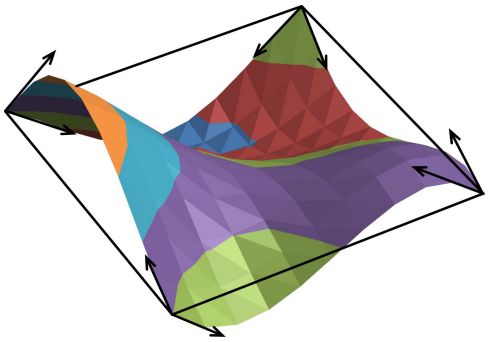

Fig. 1: Perlin noise interpolates and smooths between random gradients at grid points.

II. Procedural Terrain from Value Noise

Perlin noise was developed by Ken Perlin [3], [4] as a source of smooth random noise because generic random noise is too harsh for many applications such as procedural texture generation. Formally, the 2D Perlin noise function $h : \mathbb{R}^2 \rightarrow [-1, 1]$. The Perlin noise algorithm starts by computing random gradient vectors at integer grid points. To find a noise value y at $(x, z) \in \mathbb{R}^2$, it first finds the four closest integer points $(\lfloor x \rfloor, \lfloor z \rfloor)$, $(\lfloor x \rfloor + 1, \lfloor z \rfloor)$, $(\lfloor x \rfloor, \lfloor z \rfloor + 1)$, and $(\lfloor x \rfloor + 1, \lfloor z \rfloor + 1)$, where for all $x \in \mathbb{R}^+$, $\lfloor x \rfloor \in \mathbb{Z}^+$ is the smallest integer that does not exceed x. It then interpolates and smooths between those four gradients to get the noise value y as shown in Figure 1.

The algorithm then adds noise values at various frequencies and amplitudes in a process called $1/f$ *noise* or *turbulence*. Noise at a single frequency is called an *octave*. The amplitude is multiplied by the *persistence* (usually 0.5) from one octave to the next. The frequency is multiplied by the *lacunarity* (usually 2.0) from one octave to the next.

For the purposes of terrain generation, $y = h(x, z)$ is multiplied by a scale value and used as the height of the terrain at horizontal point (x, z) for each point (x, z) in the

Fig. 2: Terrain generated from Perlin noise.

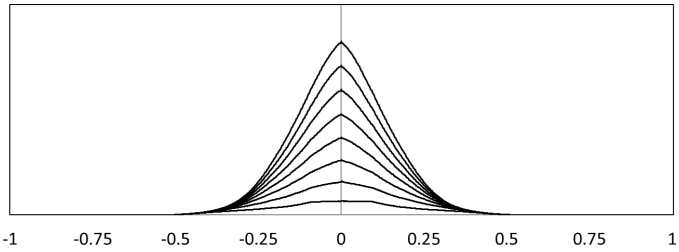

Fig. 3: Perlin noise height distribution with persistence 0.5, lacunarity 2.0, and (from bottom to top) 1–8 octaves.

2D Cartesian plane. Figure 2 shows an example of terrain generated[1] from Perlin noise using 16 octaves, persistence 0.5, and lacunarity 2.0. Notice how the terrain in Figure 2 looks uniform and nondescript. Once thing that appears to be lacking is the presence of distinctive terrain features that can be used for navigation. Part of the problem is that the height distribution (see Figure 3) is highly symmetrical.

Perlin noise can be made into *value noise* [6], [7], [8], [9] by choosing a random value at each grid point and interpolating between those instead of between gradients as shown in Figure 5. Notice that unlike Perlin noise, value noise is not constrained to be zero at grid points. Value noise uses four fewer floating-point multiplications than Perlin noise for computing each height. This leads to a small speedup. Figure 6 shows the running time in milliseconds for both value noise and Perlin noise computing 2D noise values for 10^8 random points, with persistence 0.5, lacunarity 2.0 and 1–16 octaves on an Intel® Core™ i7-3930K CPU @ 3.2 GHz. Figure 7 shows that this is faster by 10–18%. In comparison, Simplex noise [4] is reputed to be about 10% faster than Perlin noise [3], although the advantage can only be seen in higher dimensions.

When height values are chosen from a uniform distribution, the rendered terrain looks similar to that depicted in Figure 4.

[1]All terrain images in this paper were rendered using Terragen 3 [12] from a DEM file.

Fig. 4: Terrain generated from value noise with a uniform probability distribution.

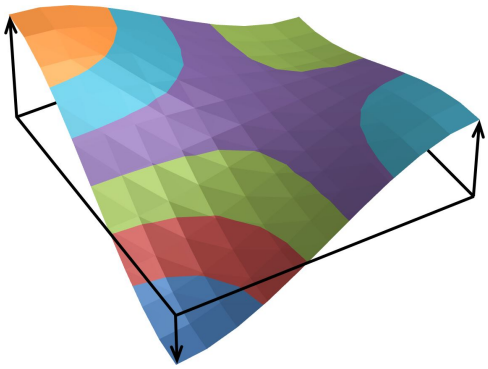

Fig. 5: Value noise interpolates and smooths between random heights at grid points.

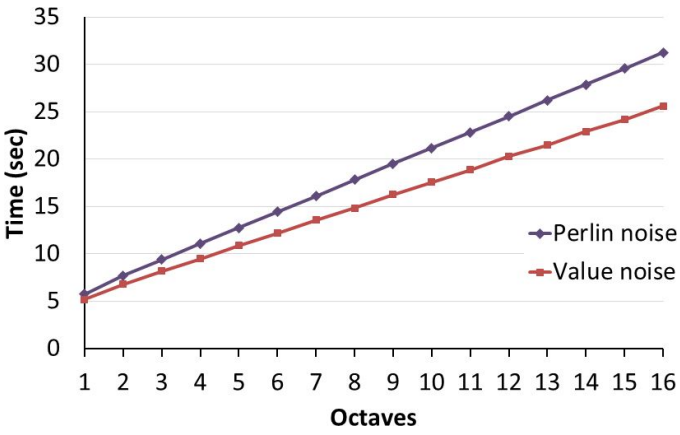

Fig. 6: Running time for Perlin noise and value noise measured in milliseconds for computing 2D noise values for 10^8 random points, with persistence 0.5, lacunarity 2.0 and 1–16 octaves.

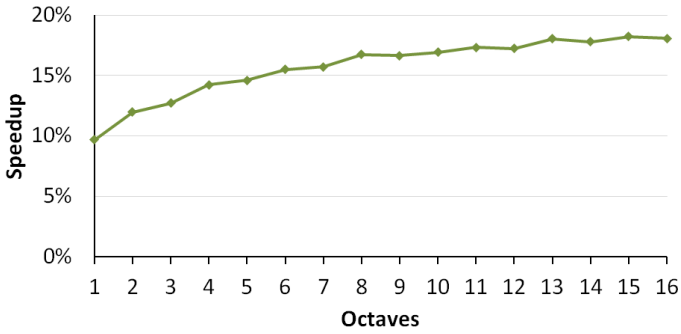

Fig. 7: Percentage speedup for value noise computing 2D noise values for 10^8 random points, with persistence 0.5, lacunarity 2.0 and 1–16 octaves.

Distribution	Height at `i`
Midpoint	See Shankel [13]
Linear	`(float)i/BM`
Cosine	`cos(PI * (float)i/BM)`
Normal	`exp(-(i-B/2)*(i-B/2)/100.0f)`

TABLE I: Some probability distributions and the corresponding height functions.

Instead of choosing height values from a uniform distribution, suppose we use some probability distribution function such as those described in Table I. Figures 8 and 9 show examples of terrain generated using value noise with various probability distributions. It can clearly be seen that the choice of probability distribution expresses itself in terrain with varying amounts of mountains, hills, and plains. The height distributions of the generated terrain are given in Figure 12. Compare to Figure 3.

III. Analysis of Geospatial Data

Spatial dependency is the term used by geographers for the observation that the elevation of terrain at any point $(x, z) \in \mathbb{R}^2$ tends, in general, to be correlated to the elevation

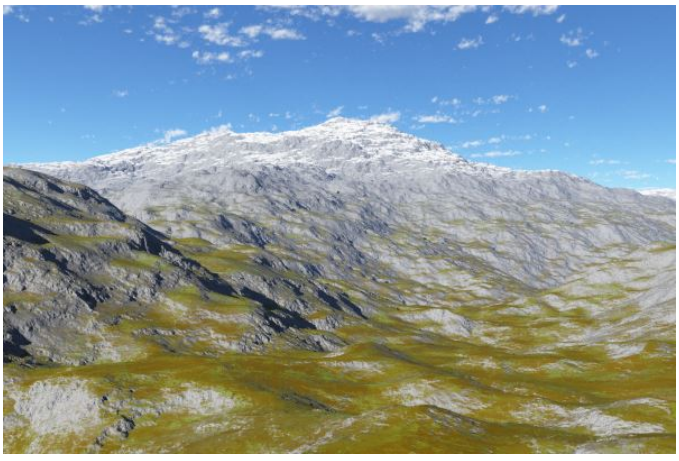

Fig. 8: Terrain generated from value noise with a midpoint displacement probability distribution.

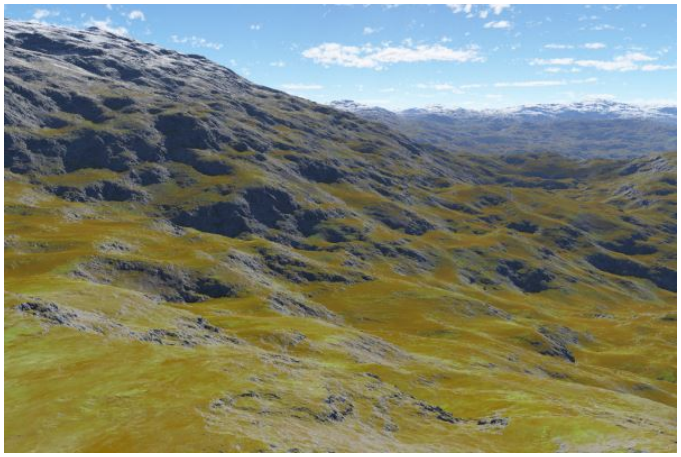

Fig. 9: Terrain generated from value noise with a linear probability distribution.

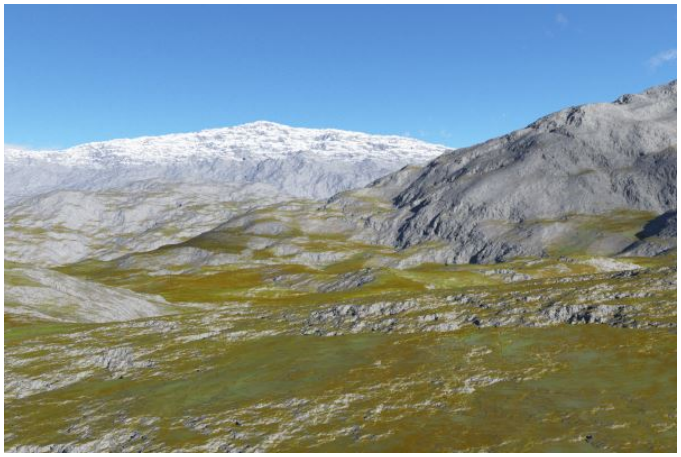

Fig. 10: Terrain generated from value noise with a cosine probability distribution.

Fig. 11: Terrain generated from value noise with a normal probability distribution.

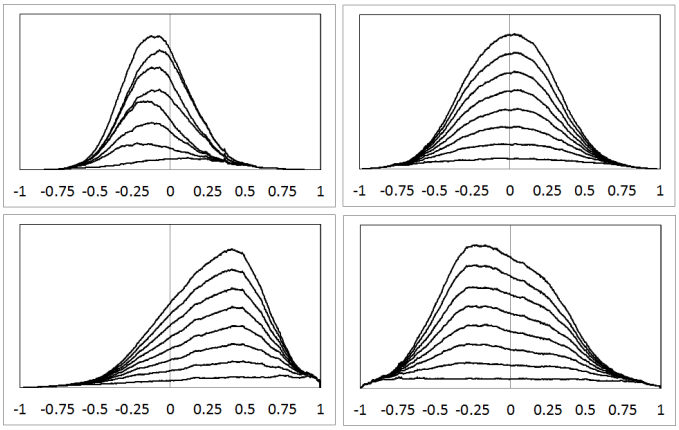

Fig. 12: Value noise height distributions from height distributions (clockwise from top left) midpoint displacement, linear, cosine, and normal, with persistence 0.5, lacunarity 2.0, and (from bottom to top) 1–8 octaves.

of the terrain at points $(x + \delta_x, z + \delta_z)$ for the appropriate small values of $\delta_x, \delta_z \in \mathbb{R}$. The commonly accepted wisdom is that terrain elevation over a geographically contiguous area is usually somewhat, but almost never exactly normally distributed, and that the variations from a perfect normal distribution are often things that make the terrain "interesting" (see, for example, Berry [14], [15]).

Common deviations from a normal distribution include *noise*, which can be but is not necessarily from measurement error, *spikes*, which can sometimes be correlated with localized geographic features such as mesas, escarpments or whoodoos, *skewness*, which is a measure of asymmetry about the mean, and *kurtosis*, which measures the second derivative at the highest point(s) of the distribution, that is, whether the bell-curve is pointed or flat.

A casual examination of USGS elevation data reveals that on a local scale (say tens of km^2) elevation often appears to have an underlying normal distribution, but on a larger scale (say hundreds of km^2), the sources of abnormality tend to predominate. This is consistent with *Tobler's First Law of Geography*, which states that "All things are related, but nearby things are more related than distant things" (Tobler [16]).

For example, Figure 13 (taken from the UAGRC [17], which serves elevation data for the state of Utah from the United States Geological Survey [10] correlated with digital satellite images) shows a map of approximately 400km^2 and its corresponding height distribution. The distribution is clearly only vaguely bell-shaped, and the dominant feature is a large spike on the left of the distribution. Figure 14 shows the height distributions for all of the 25km^2 size subgrids from Figure 13, which similarly show various amounts of normalness, noise, spikes, skewness, and kurtosis.

IV. A New Approach to Terrain Design

A designer wishing to generate a a particular type of terrain need only carry out the following procedure.

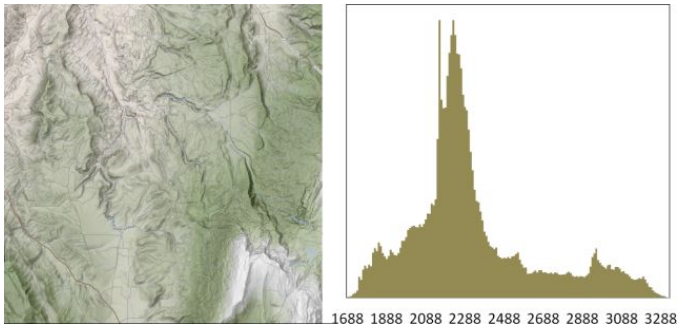

Fig. 13: Map of a 400km^2 region of Utah from the UAGRC (left) with its height distribution (right).

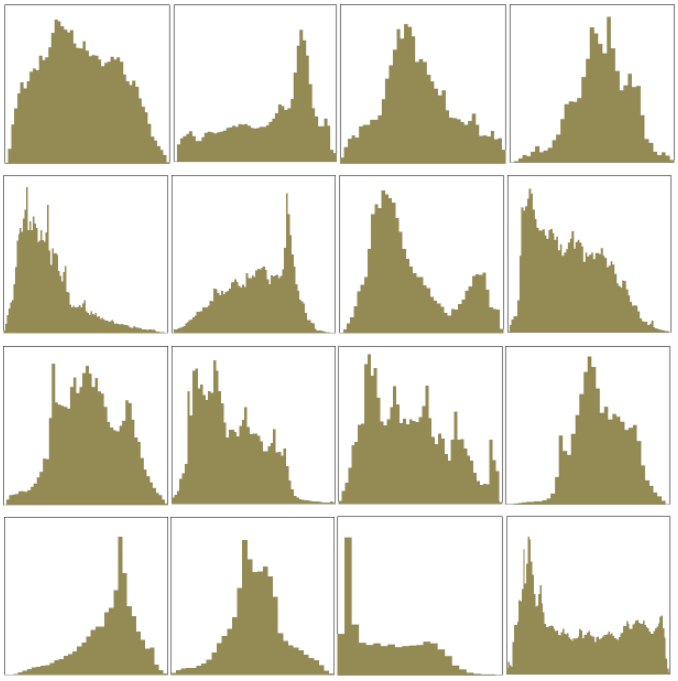

Fig. 14: Height distributions of the sixteen 25km^2 subgrids of the area shown in Figure 13.

1) Identify a geographical location whose height distribution suits the needs of the design. For example, the designer might pick Utah for mountainous terrain.
2) Download elevation data for the area chosen in Step 1 and extract a reasonably part of it that has interesting characteristics. For example, Figure 15 (left) shows 1.34 km^2 of real terrain from Utah on the left, with the rendered version on the right for comparison with later procedurally generated results. Suppose that the data has elevation values for p points.
3) Extract a probability distribution from the elevation data obtained in Step 2. For example, Figure 15 (right) shows the probability distribution for the terrain shown in Figure 15 (left) in bands of 10m elevation.
4) Scale the elevations from Step 3 to the range $[-1, 1]$.

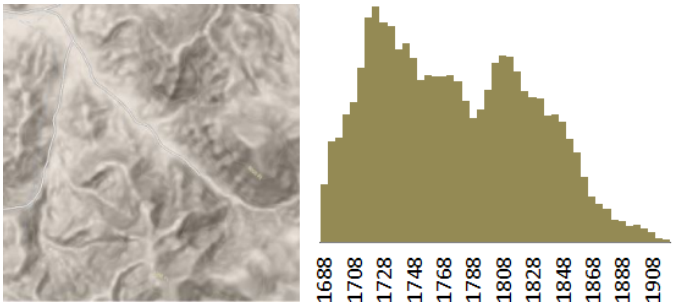

Fig. 15: Map of an interesting region of Utah from the UAGRC (left) with its height distribution (right).

From	To	Count
-1.00	-0.91	7
-0.91	-0.83	10
-0.83	-0.74	14
-0.74	-0.65	21
-0.65	-0.57	19
-0.57	-0.48	17
-0.48	-0.39	15
-0.39	-0.30	15
-0.30	-0.22	15
-0.22	-0.13	13
-0.13	-0.04	11
-0.04	0.04	15

From	To	Count
0.04	0.13	17
0.13	0.22	14
0.22	0.30	13
0.30	0.39	11
0.39	0.48	10
0.48	0.57	7
0.57	0.65	4
0.65	0.74	3
0.74	0.83	2
0.83	0.91	1
0.91	1.00	1

TABLE II: Probability distribution table corresponding to the terrain in Figure 15. The entries in the third column are the number of values randomly chosen from the range indicated by the entries in the first two columns. Note that the entries in the third column sum to 256.

Choose a small number (say $s/10$) of scaled bands for the probability distribution table. For each band $[\alpha, \beta)$, let ℓ be the number of points whose scaled height h lies in the range $\alpha \leq h < \beta$ (if $\beta = 1$ then let the last inequality be "$\leq$" instead of "$<$"). Let $n = \text{round}(s\ell/p)$ where for all $x \in \mathbb{R}$, $\text{round}(x) \in \mathbb{Z}$ is the closest integer to x. Then n is the number of values in the probability distribution table to be chosen from the range $[\alpha, \beta)$. For example, each row of Table II shows the corresponding values of α, β, and n. The resulting probability distribution table will be a quantized version of the original distribution.

5) Generate terrain using the table from Step 4.
6) Check samples of the terrain from Step 5 for suitability.

Steps 1 and 6 are the only ones that require the creativity of a design professional. Steps 2–5 can be automated or carried out by an assistant. For example, Figure 16 shows some terrain generated directly from GIS data for the area of Utah shown in Figure 15, and Figure 17 shows some random terrain generated using value noise with a displacement probability distribution obtained from GIS data for the same area of Utah.

V. Conclusion

We have described how value noise can be used in terrain generation to reduce the need for time-consuming post-processing to achieve variations in terrain. Our method allows intuitive designer control of the variations the generated terrain using probability distributions. Interesting probability distributions can be constructed by hand, computed with standard math functions, or created from spatial analysis of GIS data.

Remaining open problems include the identification of interesting height distributions from world-wide GIS data, and the design of an algorithm for artificially generating plausible height distributions that appear similar to those found in the real world, such as those shown in Figure 14.

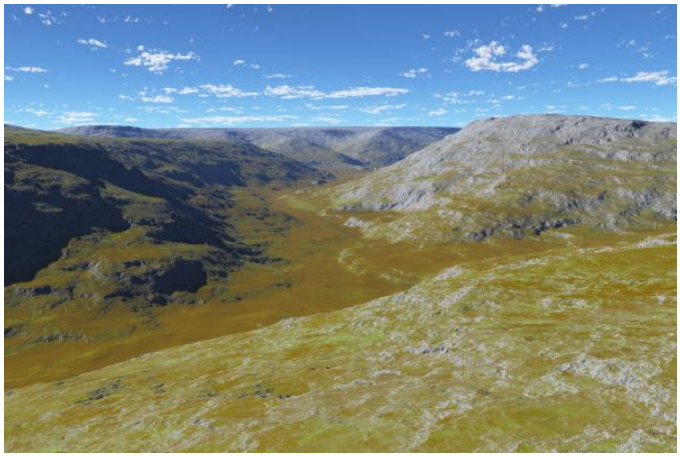

Fig. 16: Terrain generated directly from GIS data for the area of Utah shown in Figure 15.

Fig. 17: Terrain generated using value noise with a probability distribution from the area of Utah shown in Figure 15.

References

[1] M. A. Persson. (Viewed 2013) Minecraft. [Online]. Available: https://minecraft.net/

[2] R. M. Smelik, K. J. De Kraker, T. Tutenel, R. Bidarra, and S. A. Groenewegen, "A survey of procedural methods for terrain modelling," in *Proceedings of the CASA Workshop on 3D Advanced Media in Gaming and Simulation (3AMIGAS)*, 2009.

[3] K. Perlin, "An image synthesizer," in *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. ACM, 1985, pp. 287–296.

[4] ——, "Improving noise," in *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*. ACM, 2002, pp. 681–682.
[5] D. Ebert, S. Worley, F. Musgrave, D. Peachey, and K. Perlin, *Texturing & Modeling, a Procedural Approach*, 3rd ed. Elsevier, 2003.
[6] zooms...@hotmail.com. (2010) Value_noise: Explanation of value noise. [Online]. Available: https://code.google.com/p/fractalterraingeneration/wiki/Value_Noise
[7] A. van der Westhuizen. (2012) Value noise terrain generation. [Online]. Available: http://www.avanderw.co.za/value-noise-terrain-generation/
[8] aab-web. (2012) Noise part 1. [Online]. Available: http://scratchapixel.com/lessons/3d-advanced-lessons/noise-part-1/
[9] H. Elias. (Viewed 2013) Perlin noise. [Online]. Available: http://freespace.virgin.net/hugo.elias/models/m_perlin.htm
[10] USGS. (Viewed 2013) The US Geological Survey. [Online]. Available: http://www.usgs.gov/
[11] I. Parberry. (2013) Designer Worlds. [Online]. Available: http://larc.unt.edu/ian/research/valuenoise/
[12] M. Fairclough. (Viewed 2013) Terragen 3. [Online]. Available: http://planetside.co.uk/products/terragen3
[13] J. Shankel, "Fractal terrain generation — Midpoint displacement," in *Game Programming Gems*, M. DeLoura, Ed. Charles River Media, 2000, pp. 503–507.
[14] J. K. Berry, *Beyond Mapping III*. BASIS Press, 2013.
[15] ——, "Moving mapping to analysis of mapped data," *GeoWorld*, pp. 18–19, December 2004.
[16] W. Tobler, "A computer movie simulating urban growth in the Detroit region," *Economic Geography*, vol. 46, no. 2, pp. 234–240, 1970.
[17] UAGRC. (Viewed 2013) The Utah Automated Geographic Reference Center. [Online]. Available: http://gis.utah.gov/data/