Designer Worlds: Procedural Generation of Infinite Terrain from Real-World Elevation Data

Ian Parberry
University of North Texas

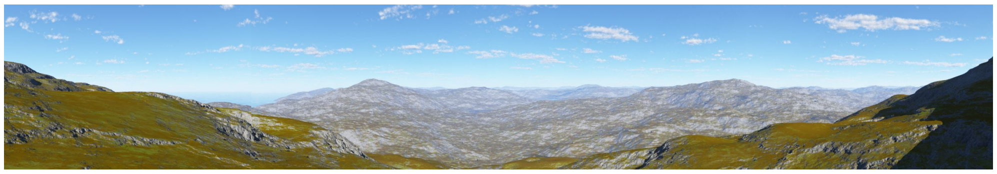

Figure 1. A 180° panorama of terrain generated using elevation data from Utah.

Abstract

The standard way to procedurally generate random terrain for video games and other applications is to post-process the output of a fast noise generator such as Perlin noise. Tuning the post-processing to achieve particular types of terrain requires game designers to be reasonably well-trained in mathematics. A well-known variant of Perlin noise called *value noise* is used in a process accessible to designers trained in geography to generate geotypical terrain based on elevation statistics drawn from widely available sources such as the United States Geographical Service. A step-by-step process for downloading and creating terrain from real-world USGS elevation data is described, and an implementation in C++ is given.

1. Introduction

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. First, it needs to be fast, meaning that it has to use only a fraction of the computing power on a current-generation computer. Second, it needs to be both random and structured so that it creates content that is varied and interesting. Third, it needs to be controllable in a natural and intuitive way.

As noted in the excellent survey paper by Smelik et al. [2009], procedural terrain generation is often based on fractal noise generators such as Perlin noise [Perlin 1985; Perlin 2002] (for more details see, for example, the book by Ebert et al. [2003]). 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 inclined.

We describe how to use a variant of Perlin noise called *value noise* for the procedural generation of terrain data for use in a video game or terrain simulator. We then perform a spatial analysis of elevation data for the state of Utah from the United States Geological Survey and show how the results of such an analysis can be easily integrated with value noise to generate procedural terrain that shares height characteristics with real terrain.

The remainder of this paper is divided into three sections. Section 2 contains an overview of value noise and describes how to use it to generate random terrain. Section 3 describes the results of a spatial analysis of GIS data from some interesting terrain. Section 4 shows how to use the GIS data from Section 3 in the value noise algorithm.

2. Procedural Terrain from Value Noise

Perlin noise was developed by Ken Perlin [1985; 2002] 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 inte-

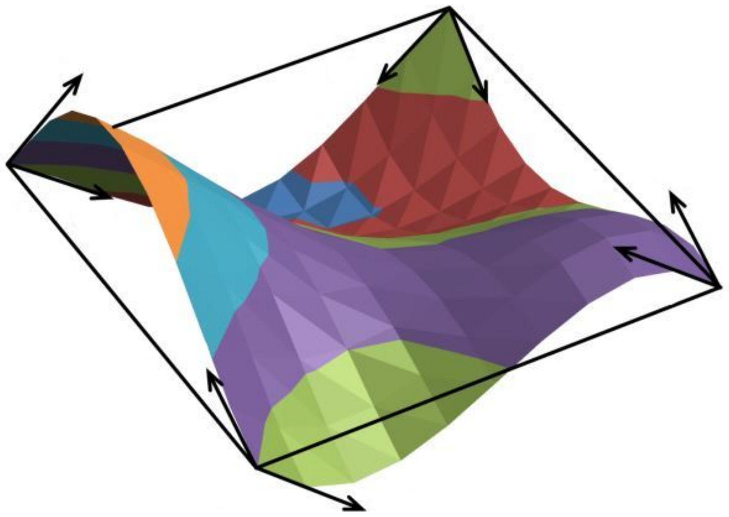

Figure 2. Perlin noise interpolates and smooths between random gradients at grid points.

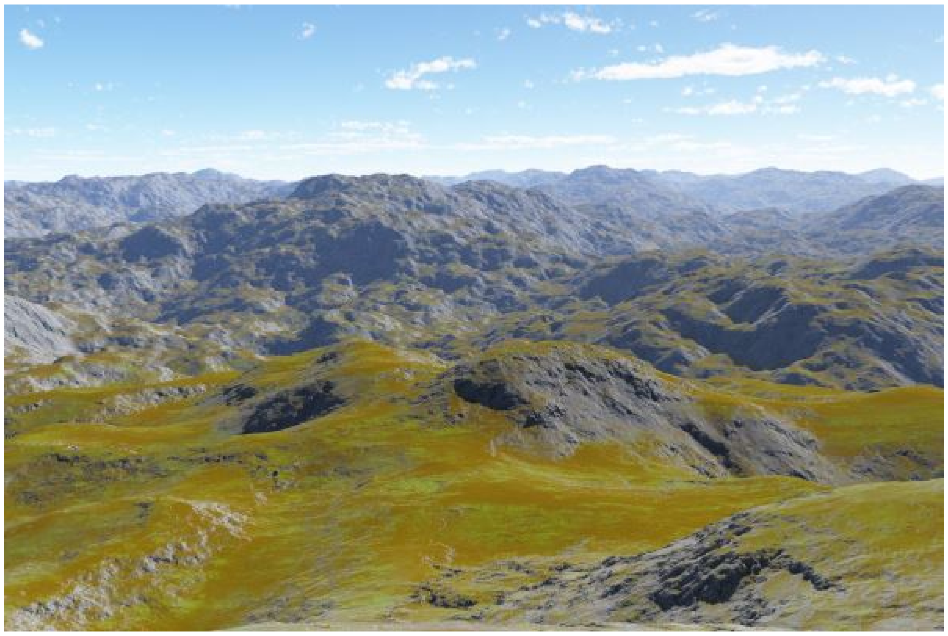

Figure 3. Terrain generated from Perlin noise.

ger 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 2.

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 2D Cartesian plane. Figure 3 shows an example of terrain generated from Perlin noise using eight octaves, persistence 0.5, and lacunarity 2.0. (All terrain images in this paper were rendered using Terragen[1] from a DEM file.) Notice how the terrain in Figure 3 looks uniform and nondescript; there is an absence of distinctive terrain features that can be used for navigation. Part of the problem is the height distribution of Perlin noise (see Figure 4).

The well-known value noise algorithm differs from the Perlin noise algorithm in that it chooses a random *height* at each grid point and interpolates between those instead of between gradients, as shown in Figure 5. Unlike Perlin Noise, value noise is not constrained to be zero at grid points. Terrain generated with Perlin noise usually

[1]Available for free download from `http://planetside.co.uk/products/terragen3`.

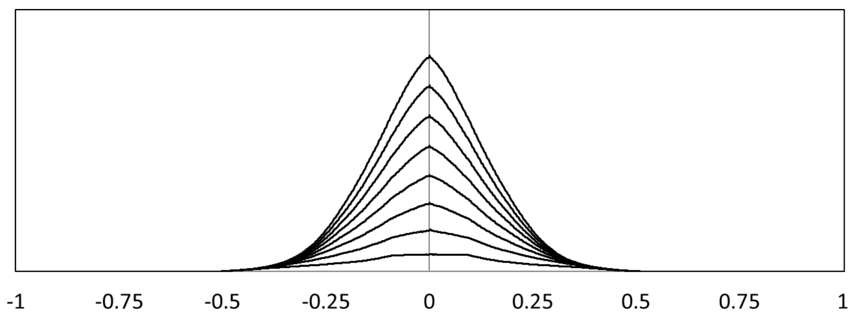

Figure 4. Perlin noise height distribution with persistence 0.5, lacunarity 2.0, and (from bottom to top) 1–8 octaves.

consists of ranges of mountains or hills separated by valleys. Hitting zero elevation means that there are no large valleys. Smaller valleys don't join, they are blocked by intermediate hills. Figure 6 shows some terrain generated from value noise which, like the terrain generated from Perlin noise shown in Figure 3, looks uniform and nondescript.

Value noise uses four fewer floating-point multiplications than Perlin noise per point per octave. This leads to a small speedup. Figure 7 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^{TM} i7-3930K CPU @ 3.2 GHz. Figure 8 shows that this is faster by 10–18%. In comparison, simplex noise [Perlin 2002] is reputed to be about 10% faster than Perlin noise [Perlin 1985], although the advantage can only be seen in higher dimensions.

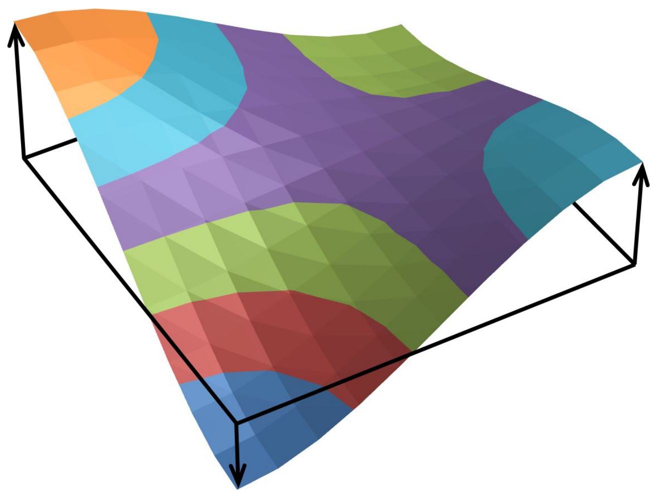

Figure 5. Value noise interpolates and smooths between random heights at grid points.

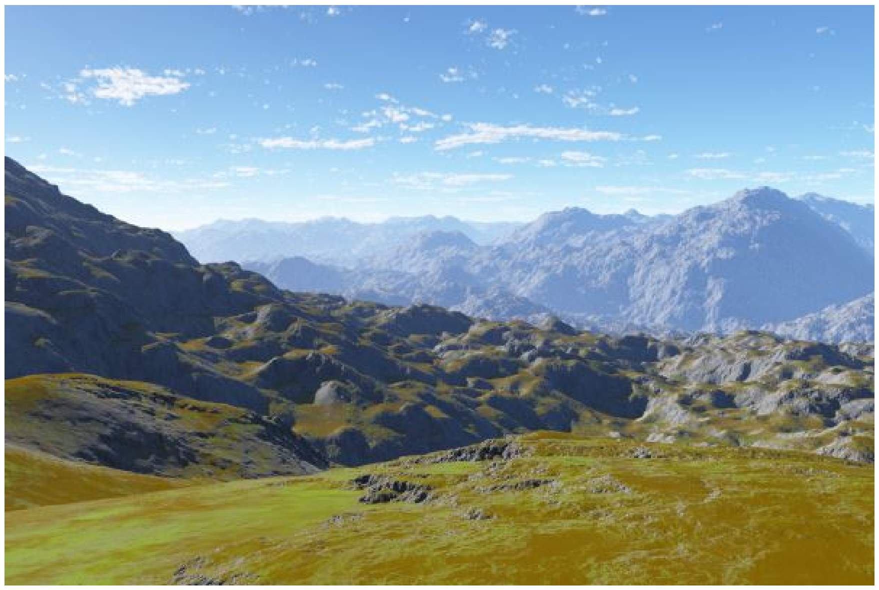

Figure 6. Terrain generated from value noise.

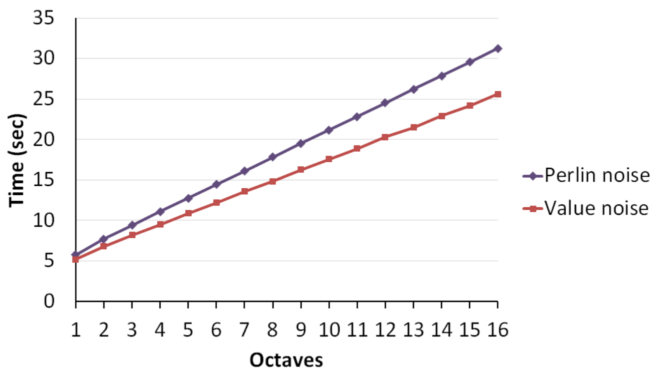

Figure 7. Running time for Perlin noise and value noise measured in milliseconds for computing 2D noise values for 10^8 random points, with persistence 0.5, lacunarity 2.0 and 1–16 octaves.

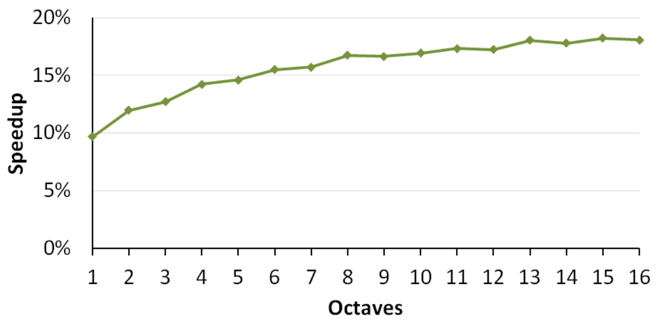

Figure 8. Percentage speedup for value noise computing 2D noise values for 10^8 random points, with persistence 0.5, lacunarity 2.0 and 1–16 octaves.

3. 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 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 [Berry 2004; Berry 2013] 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".

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 elevation data from the United States Geographical Service (USGS) reveals that on a local scale (say tens of square kilometers) elevation often appears to have an underlying normal distribution, but on a larger scale (say hundreds of square kilometers), the sources of abnormality tend to predominate. This is consistent with *Tobler's First Law of Geography* [Tobler 1970], which states that "All things are related, but nearby things are more related than distant things".

For example, Figure 9 (taken from the Utah Automated Geographic Reference Center, which serves elevation data[2] for the state of Utah from the United States

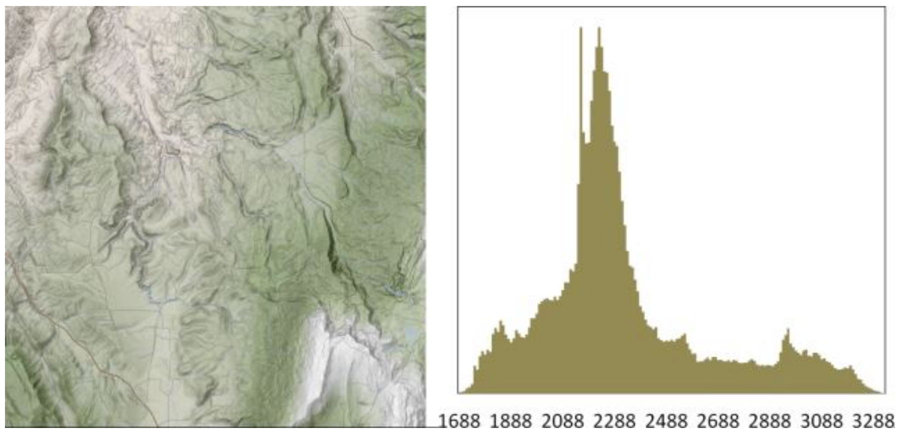

Figure 9. Map of a 400 square kilometer region of Utah from the UAGRC (left) with its height distribution (right). The horizontal axis of the height distribution shows elevation ranges, and the vertical axis shows the number of data points within each range.

[2]Available for free download from `http://gis.utah.gov/data/`.

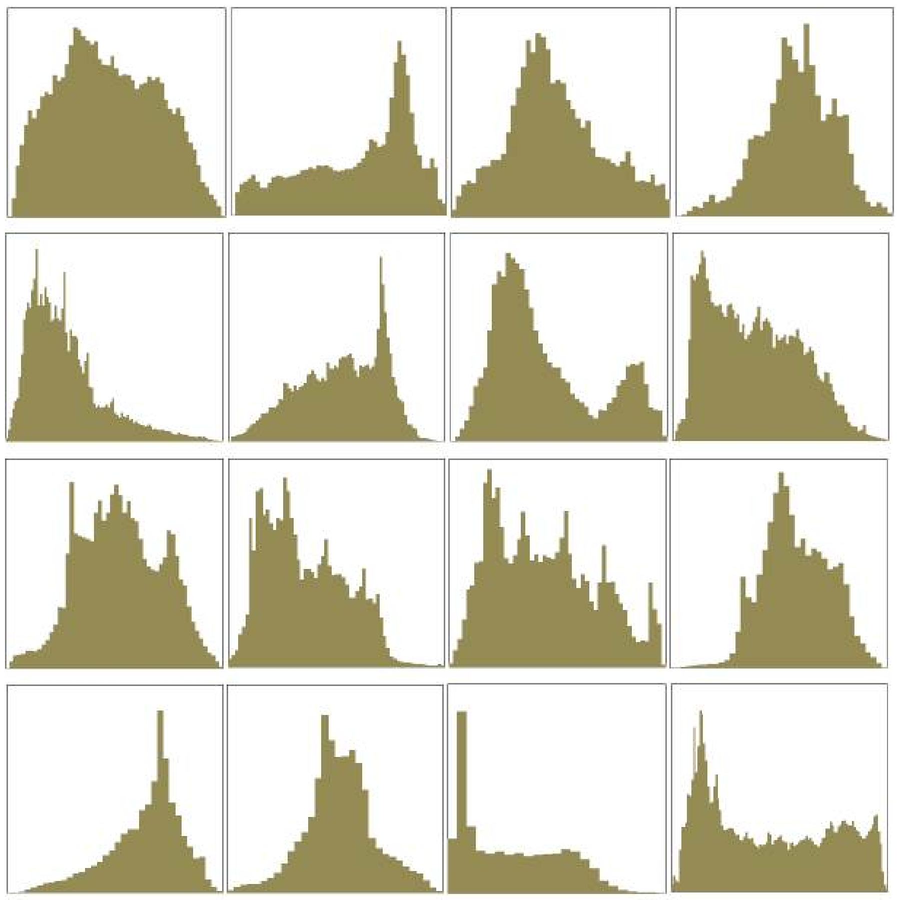

Figure 10. Height distributions of the sixteen 25 square kilometer subgrids of the area shown in Figure 9. The horizontal axes shows elevation ranges, and the vertical axes show the number of data points within each range.

Geological Survey correlated with digital satellite images) shows a map of approximately 400 square kilometers 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 10 shows the height distributions for all of the 25 square kilometer subgrids from Figure 9, which similarly show various amounts of normalness, noise, spikes, skewness, and kurtosis.

4. Terrain Design Using USGS Elevation Data

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

1. Identify a geographical location whose height distribution suits the needs of the design. For example, the designer might pick Utah for mountainous terrain.

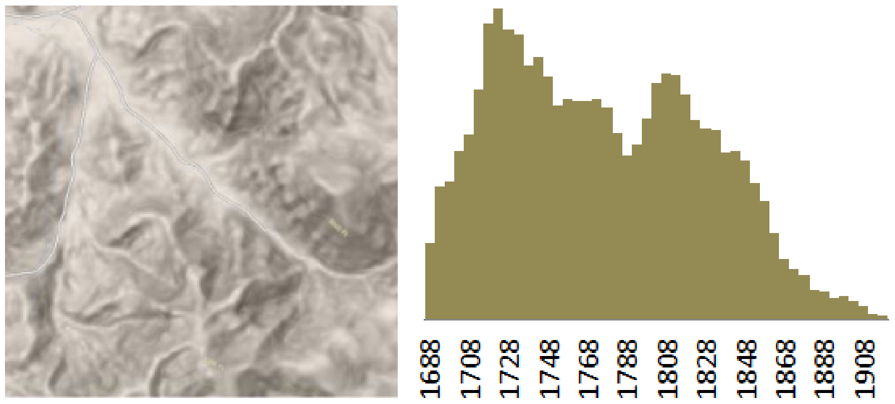

Figure 11. Map of an interesting region of Utah from the UAGRC (left) with its height distribution (right). The horizontal axis of the height distribution shows elevation ranges, and the vertical axis shows the number of data points within each range.

2. Download elevation data for the area chosen in Step 1 and extract a part of it that has interesting characteristics. For example, the designer might pick the 1.34 square kilometers of real terrain from Utah shown in Figure 11 (left).

3. Extract a height distribution from the elevation data obtained in Step 2. For example, Figure 11 (right) shows the height distribution for the terrain shown in Figure 11 (left) in bands of 10 m elevation.

4. Scale the elevations from Step 3 to the range $[-1,1]$. Suppose that the data has elevation values for p points, and there are s elevation bands. Choose a small number (say $s/10$) of scaled bands for the height 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 height distribution table to be chosen from the range $[\alpha,\beta)$. For example, each row of Table 1 shows the corresponding values of α, β, and n. The resulting height 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 12 shows some terrain generated directly from GIS data for the area of Utah shown in Figure 11, while Figures 1 and 13 show some random

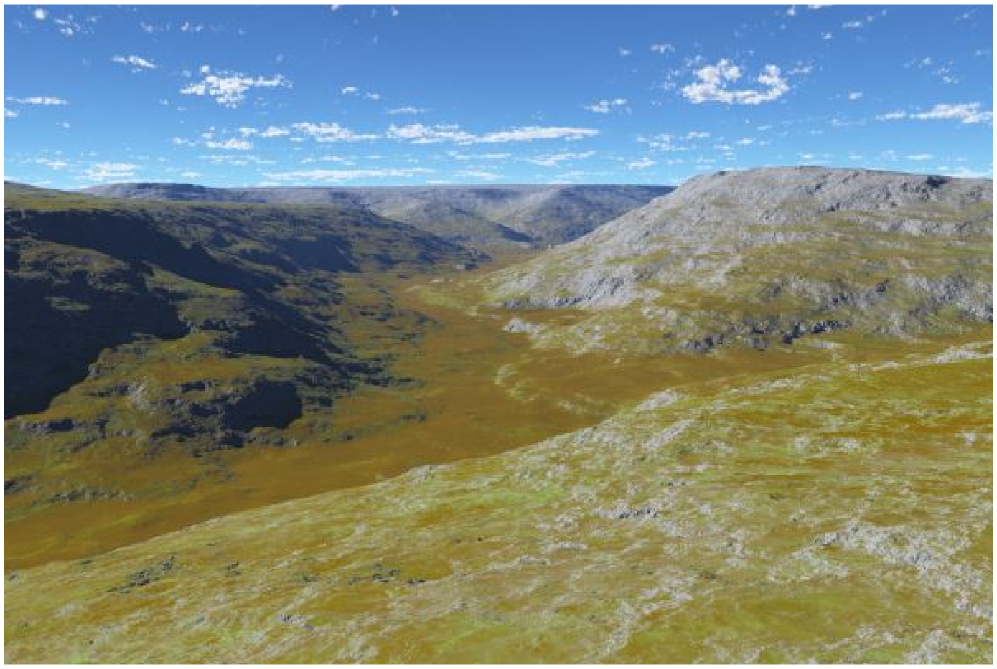

Figure 12. Terrain generated directly from GIS data for the area of Utah shown in Figure 11.

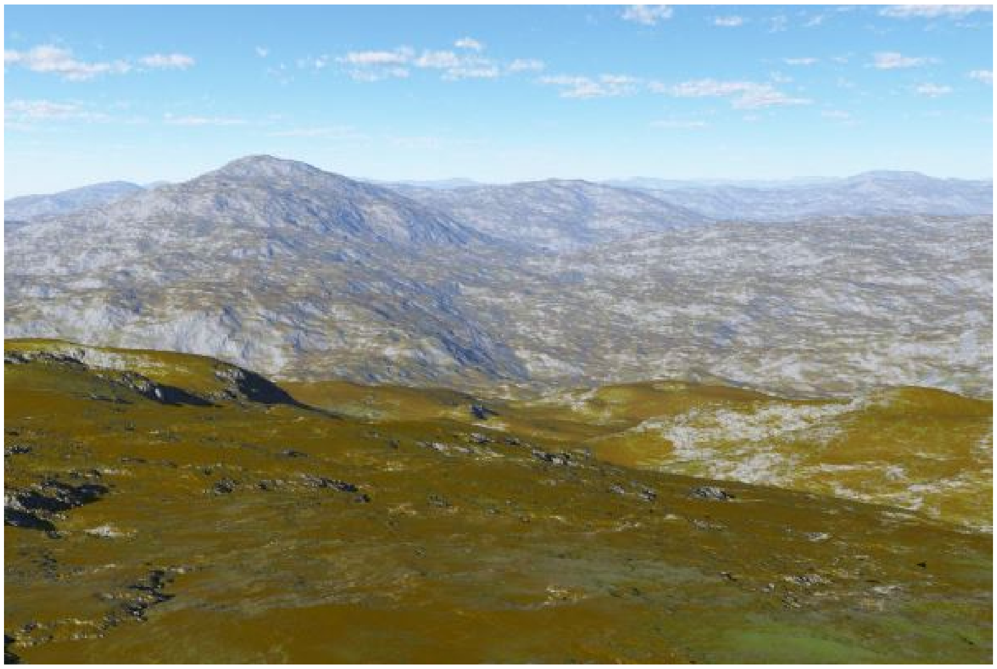

Figure 13. Terrain generated using value noise with a height distribution from the area of Utah shown in Figure 11.

From	To	Count
-1.00	-0.91	8
-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
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 1. The height distribution table corresponding to the terrain in Figure 11. 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 "Count" columns sum to 256.

terrain generated using value noise with a displacement height distribution obtained from GIS data for the same area of Utah. Figures 14 shows four examples of terrain generated from other height distributions.

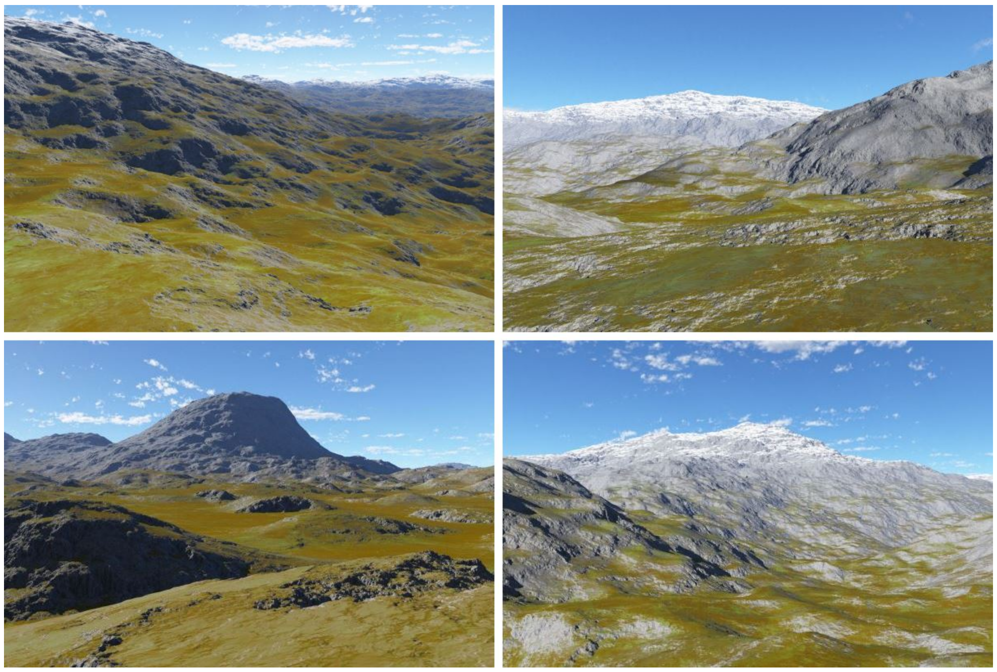

Figure 14. Terrain generated from four different height distributions.

5. Conclusion

We have described how value noise can be used in terrain generation to reduce the need for post-processing to achieve thematic terrain style. Our method allows intuitive designer control of the generated terrain using height distributions obtained from a 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 10.

References

BERRY, J. K. 2004. Moving mapping to analysis of mapped data. *GeoWorld* (December), 18–19. 79

BERRY, J. K. 2013. *Beyond Mapping III*. BASIS Press, Fort Collins, CO. 79

EBERT, D., WORLEY, S., MUSGRAVE, F., PEACHEY, D., AND PERLIN, K. 2003. *Texturing & Modeling, a Procedural Approach*, 3rd ed. Morgan Kaufmann/Elsevier, Amsterdam. 74

PERLIN, K. 1985. An image synthesizer. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, ACM, New York, NY, 287–296. 74, 75, 77

PERLIN, K. 2002. Improving noise. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, ACM, New York, NY, 681–682. 74, 75, 77

SMELIK, R. M., DE KRAKER, K. J., TUTENEL, T., BIDARRA, R., AND GROENEWEGEN, S. A. 2009. A survey of procedural methods for terrain modelling. In *Proceedings of the CASA Workshop on 3D Advanced Media in Gaming and Simulation (3AMIGAS)*. 74

TOBLER, W. 1970. A computer movie simulating urban growth in the Detroit region. *Economic Geography 46*, 2, 234–240. 79

Index of Supplemental Materials

The supplementary material consists of a zip file containing four folders.

1. Data. This folder contains a DEM file `12SVH200800.asc` and the corresponding satellite image downloaded from the Utah Automated Geographic Reference Center. This is used as a running example in the code.

2. Analyze. This folder contains a Visual Studio 2012 Solution and C++ source code that, when compiled and executed, reads `12SVH200800.asc` from the Data folder and outputs two text files, `output.txt` which contains data for a histogram, and `code.txt` which contains a code snippet to be added to the Designer Worlds generator. There is also an Excel spreadsheet `data.xlsx` containing a histogram drawn from the data in `output.txt`.

3. Generate. This folder contains a Visual Studio 2012 Solution and C++ source code for the Designer Worlds Generator. The code from `code.txt` in the Analyze folder has been pasted into the appropriate place in `main.cpp`. Each time this code is executed it will output a DEM file for a piece of random terrain similar to the original DEM file `12SVH200800.asc` in the Data folder. It also contains a generated DEM file `1293054609.asc`.

4. View. This folder contains a Terragen Project File that will render the terrain described by `1293054609.asc` in the Generate folder. There is also an image file `1293054609.jpg` generated by running Terragen on that Project file.

Author Contact Information

Ian Parberry
Dept. of Computer Science and Engineering
University of North Texas
1155 Union Circle #311366
Denton, Texas 76203–5017

http://larc.unt.edu/ian

Ian Parberry, Designer Worlds: Procedural Generation of Infinite Terrain from Real-World Elevation Data, *Journal of Computer Graphics Techniques (JCGT)*, vol. 3, no. 1, 74–85, 2014
http://jcgt.org/published/0003/01/04/

Received:	2013-11-14		
Recommended:	2013-12-20	Corresponding Editor:	Paul Strauss
Published:	2014-03-11	Editor-in-Chief:	Morgan McGuire