Noisy Gradient Meshes

Hettinga, Gerben J.; van Beckhoven, Rowan; Kosinka, Jiří

Published in:
Graphical Models

DOI:
10.1016/j.gmod.2019.101024

Publication date:
2019

Noisy Gradient Meshes: Augmenting Gradient Meshes with Procedural Noise

Gerben J. Hettinga
g.j.hettinga@rug.nl

Rowan van Beckhoven
rowanvanbeckhoven@gmail.com

Jiří Kosinka
j.kosinka@rug.nl

Bernoulli Institute, University of Grnoningen, Nijenborg 9, 9747 AG, Groningen, the Netherlands

Abstract

We extend the gradient mesh vector graphics primitive with procedural noise functions. Specifically, we couple Perlin, Worley and Gabor noise to the gradient mesh. We allow local parameters controlling the noise functions to be defined at the vertices of the mesh. The parameters are interpolated along with the geometry similarly to how colour is interpolated in an ordinary gradient mesh, allowing for spatially varying noise patterns.

These noisy gradient meshes facilitate a sparse representation of high frequency regions along with underlying smooth colour gradients. The meshes are easy to edit and efficient to evaluate on graphics hardware, making them a suitable candidate for inclusion in modern vector graphics authoring tools. We demonstrate the utility of our method on gradient meshes with added noise functions. Additionally, we show that the approach can be used in combination with regular surface meshes where noise functions are used to govern their displacement mapping.

1. Introduction

Vector graphics is a powerful approach for representing scalable illustrations [4]. Vector graphics illustrations can be composed of a multitude of different primitive types such as lines and colour gradients. Existing high-level vector graphics primitives are able to model smooth and sharp colour transitions well and can be used to vectorise raster images featuring large regions with constant or slowly changing colour gradients. However, when trying to model natural images, these methods often result in many small gradient primitives due to high frequency regions present in the image and the inability of traditional vector primitives to model such regions without excessive refinement. Working with many gradient primitives is a tedious process and thus a solution has to be found in which high-level detail can be achieved while keeping the underlying mesh simple.

Procedural noise functions have been used in computer graphics to create detail on otherwise smoothly shaded surfaces through the use of procedural textures, or to displace the surface with procedural displacement mapping [6]. Procedural noise functions have been used for texture synthesis before, but they are rarely used in combination with vector graphic primitives. The incorporation of noise functions into vector graphics is an ideal combination. The functions are very compactly defined, requiring only a few parameters to create a texture of any desired resolution. They can be evaluated at any arbitrary parameter value and thus are infinitely scalable.

In this work we combine procedural noise functions with the gradient mesh vector graphics primitive. The gradient mesh interpolates colour and colour gradients, and can be made to model smoothly varying colour transitions over an image. The combination of the two techniques results in a vector graphics primitive where the parameters of the noise function can be varied over the mesh to obtain variable levels of detail and smoothly varying colours. This greatly broadens the expressiveness of gradient meshes and the range of objects which they can model, but still makes use of the simplicity of the traditional gradient mesh.

The general approach is to find parameters which control the noise and which can be varied such that no artefacts are present in the end result. To do this, the impact of different parameters on the final noise field is evaluated. These parameters can be implemented globally or locally. Global parameters are not bound to the mesh structure and are therefore easy to edit, whereas local parameters are bound to the vertices of the mesh and thus provide more control as they are set individually.

The remainder of the paper is organised as follows. In Section 2, we consider related work where vector graphics and procedural noise functions have been combined. Then we briefly describe the essentials of gradient meshes and procedural noise functions for our primitive in Section 3. Then, in Section 4, we detail the construction of gradient meshes with procedural noise functions. We describe which parameters we use and how they are interpolated. We

also provide, in Section 5, several ways of editing the noise field through post-processing functions. Results of gradient meshes and also of general displacement mapping on surfaces are shown in Section 6. In Section 7, we discuss limitations and applications. Finally, the paper is concluded in Section 8.

2. Related Work

Gradient meshes have been incorporated in vector graphics authoring applications such as Adobe Illustrator [19], Coreldraw [5] and Inkscape [1]. Gradient meshes have also been used for image vectorisation [12, 16]. More recent developments in gradient meshes include a colour interpolation scheme based on subdivision surfaces [13, 18], and gradient meshes which allow T-junctions and hierarchical editing [3]. Nearly all of the research into gradient meshes has focused on extending the capabilities of the primitive to arbitrary meshes, or to use the meshes in a vectorisation process. Their combination with procedural noise functions has not been investigated so far.

The combination of procedural noise functions with other vector graphics primitives has been approached before. Jeschke et al. [9] use the diffusion curve primitive [14] to diffuse the parameters for Gabor noise kernels over the image plane. Locally a fragment shader is used to evaluate the noise kernel. In this way, the noise pattern varies smoothly over the image, interrupted only by hard transitions imposed by the defined diffusion curves. Likewise, Hnaidi et al. [8] use the diffusion curve primitive and procedural noise functions for procedural terrain generation. In their approach they make use of fractal noise composed of octaves of Perlin noise. Along with the height attributes of the diffusion curves, the persistence value of the fractal noise is diffused over the parameter plane to obtain varying levels of details in the generated terrains. Diffusion curves are a natural way to define features of the image, but do not give the user much control over the interpolated colour or noise field away from the curves.

3. Preliminaries

Here we recall some of the key components used in creating noisy gradient meshes. We review the gradient mesh primitive and several of the procedural noise functions used in this work.

3.1. Gradient Meshes

The gradient mesh primitive is a vector graphics primitive which is used to smoothly interpolate colours defined at the vertices of a regular quadrilateral mesh. Gradient handles defined at vertices can be used to distort the boundary curves of each of the patches of the mesh into the desired shape. Since colour is interpolated along with the geometry, the gradient handles dictate the flow of colour inside the mesh.

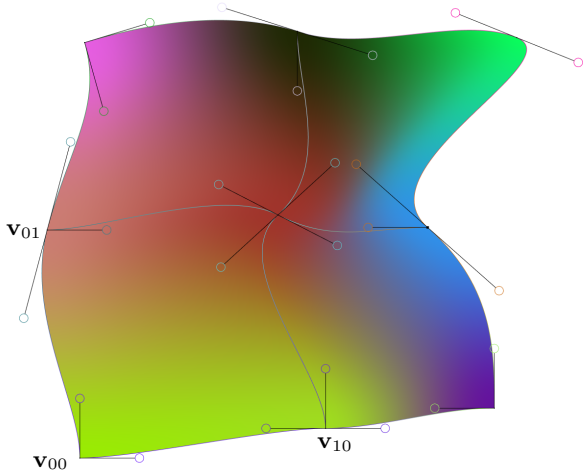


Figure 1. A simple 2×2 gradient mesh (without any noise applied to it) smoothly interpolating colours and gradients defined at the vertices $\mathbf{v}_{ij}$ of the mesh. Gradient handles are denoted by line segments emanating from their corresponding vertices and ending in circles.

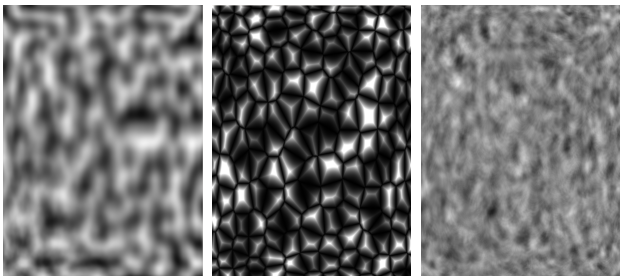
Traditionally, a Ferguson patch [7] is the underlying geometric primitive used for a gradient mesh. In this paper, we choose to represent each (quadrilateral) patch of the primitive as a bicubic Bézier patch instead; see [3] for an in-depth discussion. The vertices $\mathbf{v}_{ij}$ are defined as the tuple (x, y, r, g, b), where the first two components represent position, and the last three control its (RGB) colour. The edges of each patch are given by cubic Bézier curves. In the case of the Bézier patch, the gradient handles and the inner control points inherit the colour value (r, g, b) of the logically closest vertex. In this way, colour is interpolated along with the geometry, and, most importantly, the interpolated colour values will not traverse out of the gamut. A simple gradient mesh consisting of 4 patches interpolating colours and gradients is shown in Figure 1.

3.2. Procedural Noise Functions

We focus on three procedural noise functions in this work: Perlin, Worley and Gabor noise, but other types of noise can be just as easily incorporated, too. We choose these functions because they are mostly standardised, well-known, basic and easily implemented in the pixel shader stage of modern graphics pipelines. Figure 2 shows the basic characteristics of these noise functions.

3.2.1 Perlin Noise

Perlin noise was introduced in 1985 by Ken Perlin [15] as one of the first true procedural noise functions. This noise is based on a uniformly spaced grid, or lattice, defined on the



Perlin Worley Gabor

Figure 2. The three different procedural noise functions considered in this paper. Each gives rise to a distinctive noise pattern.

uv-plane. For a point to be evaluated, its position in the lattice is determined. For each of the four surrounding lattice points, a pseudo-random gradient vector is computed. The inner product of this gradient vector and a vector from each surrounding lattice point to the point is computed. Finally, these inner products are then in turn used in a biquintic interpolation function to guarantee smooth transitions between neighbouring cells; see Figure 2, left.

Note that the underlying lattice structure of Perlin noise is nicely aligned with that of gradient meshes. This observation allows us to combine the two, as well as other types of noise relying on regular lattice structures, as shown below.

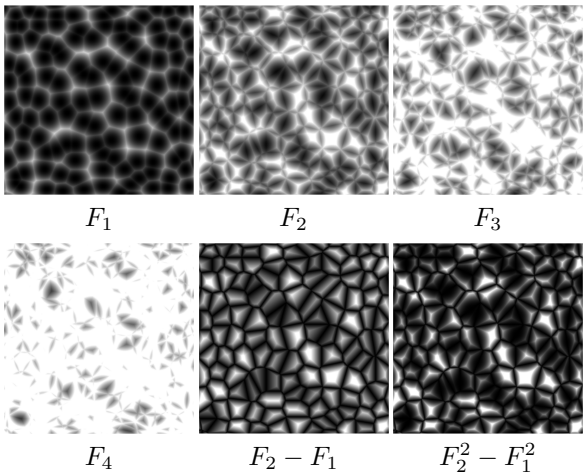
3.2.2 Worley Noise

Worley noise [20] is not a traditional noise function, but rather a texture basis function [10], producing cellular-like textures similar to Voronoi cells; see Figure 2, middle. Each cell of the lattice contains a randomly placed feature point. A function F_k is defined which computes a scalar value based on the k nearest feature points and a chosen metric. Computing F_k up to $k = 4$ produces various patterns, whereas higher values of k give rise to functions that start to look similar in appearance [20].

Linear or other combinations of F_k result in very interesting patterns, each with its own characteristics; see Figure 3. Since F_k is based on distance, different metrics can be used to obtain varying results. Other ways of influencing the functions is by changing the amount of jitter, or the placement of the feature points, in the cells of the lattice.

3.2.3 Gabor Noise

Gabor noise is a sparse convolution noise function [11]; see Figure 2, right. A noise pattern is generated by taking a weighted sum of arbitrarily placed kernels on a lattice. For the procedural evaluation, the kernels are placed on the lattice similarly to Worley noise. In this way, the noise in a cell of the grid is determined by the eight neighbouring cells, in which the kernels are placed according to a Poisson distribution. The Gabor noise function has several parameters that greatly influence the noise. The density of the kernels can be changed as well as the direction of the noise and the amount of anisotropy.



F_1 F_2 F_3

F_4 $F_2 - F_1$ $F_2^2 - F_1^2$

Figure 3. Worley noise functions and their combinations, producing distinctive patterns.

The Gabor kernel is the Gaussian function multiplied with a 2D cosine function,

$$G(u, v) = Ke^{-\pi a^2(u^2+v^2)} \cos(2\pi f(u\cos\omega + v\sin\omega)),$$

where K is the magnitude, a is the inverse width of the Gaussian function, f is the frequency, and ω is the orientation of the cosine function.

3.2.4 Fractal Noise

As mentioned above, the mutual lattice structure of gradient meshes and procedural noise functions allows us to combine them in a natural way. However, sparse gradient meshes would then hardly benefit from the added noise as the desired level of detail of a noise function is typically much higher than that of the gradient mesh itself.

Therefore, in addition to the above-mentioned procedural noise functions, we also employ fractal Brownian noise, or fractal noise for short, built on top of them. Fractal noise evaluates several layers of a noise function, but each time the frequency of the signal is increased. A parameter p is

Figure 4. Basic Perlin noise (left). Five (middle) and ten (right) octaves are assigned to the top right vertex of the mesh.

used to regulate the persistence of the different layers or frequencies of noise. With higher values of p, higher frequency components increase in magnitude and in visibility; see Figure 4. In the next section we show how to interpolate this value to locally vary the pronunciation of the different octaves of noise over a gradient mesh.

4. Noisy Gradient Meshes

The first step in creating a noisy gradient mesh is to set up the underlying parameter domain on which the procedural noise functions are generated. The parameter domain is the uv-plane on which the lattice for the procedural noise functions is imposed, which coincides with that of the underlying gradient mesh. We assign texture coordinates to the vertices of the gradient mesh. As the gradient mesh is regular rectangular, this is straightforward. A (gradient) mesh of $m \times n$ quadrilaterals with (shared) vertices $\mathbf{v}_{i,j}$, $i = 0, \ldots, m$ and $j = 0, \ldots, n$, corresponds to an $m \times n$ subdivision of the uv-plane. Thus, the texture coordinates of $\mathbf{v}_{i,j}$ are simply $u_i = i$ and $v_j = j$, creating a uniform texture space. And internally, each Bézier patch is parameterised over the unit square. In this manner, the uv-coordinates can be linearly interpolated on the faces of the gradient mesh without loss of continuity. Moreover, we can still globally refine the mesh by also globally subdividing the parameter plane along u and v parameter lines. In this manner, the grid turns into a rectilinear grid and linear interpolation of uv-coordinates maintains continuity.

Having combined the uv-coordinates with the gradient primitive already gives some control over the generated noise fields. As the position of the vertices and tangent handles of the gradient mesh are changed, the underlying parameterisation follows the flow of the mesh. Naturally, it is still possible to purposefully model discontinuities by not enforcing co-linearity or magnitude constraints on the tangent handles.

Nevertheless, this alone is not enough freedom for the user to specify the noise field as it is still defined using only global parameters, showing a similar frequency of noise all over the mesh; see Figure 2. Therefore, we introduce local parameters for each vertex of the gradient mesh. These local parameters are:

- **Persistence:** As mentioned in Section 3.2.4, the persistence parameter p regulates the prevalence of higher octaves of noise. By varying this parameter from low to high, an increasing amount of high frequency noise becomes apparent; see Figure 4. This parameter was also varied over the image plane in [8].
- **Opacity:** The opacity value α simply handles the local blending of the noise field with the colour field.
- **Distortion:** In Section 5.1, we introduce several secondary functions which manipulate the parameter domain or alter the noise field. The distortion parameter d locally regulates the amount of the applied distortion.

The handling of the parameters is done in exactly the same way as of the colour components of a traditional gradient mesh. Each vertex $\mathbf{v}_{ij}$ now becomes the tuple $(x, y, r, g, b, p, \alpha, d)$; cf. Section 3.1. The values of the noise parameters and colour components are assigned to the control points of the bicubic Bézier patch. Once again, the values of the edge and inner control points inherit the value of the logically closest vertex. Consequently, the noise parameters are interpolated with C^1 continuity along with the colour attributes, unless the user explicitly demands lower continuity to model, for example, sharper transitions or discontinuities.

For Gabor noise kernels we include even more parameters which can be interpolated. These are the level of anisotropy and the direction of the kernel; see Section 3.2.3. This creates interesting opportunities for customisation of the noise field. As the anisotropy value is decreased, there is an increase in the general orientation of the kernels. In this way, the noise direction can be guided on vertex level, thereby creating interesting patterns reminiscent of vector field visualisation [17]. Figure 5 shows one such example. In this simple 2×2 noisy gradient mesh, the centre vertex is supplied a higher anisotropy value than the vertices surrounding it. Additionally, the orientation value of the kernels assigned to the boundary vertices is angled approximately towards the centre vertex.

By passing the (u, v) coordinates and the interpolated parameters into one of the three procedural noise functions introduced in Section 3.2, a noise field is generated. The noise field varies locally according to the interpolated parameters specified at individual vertices. The noise field alone can also be edited by dragging vertices and gradient

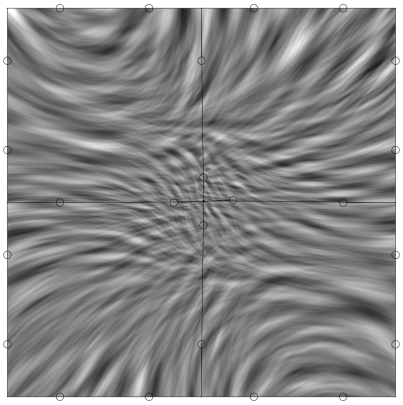


Figure 5. A noisy gradient mesh (4 patches) showing the effect of varying the anisotropy and orientation values for Gabor noise over the mesh.

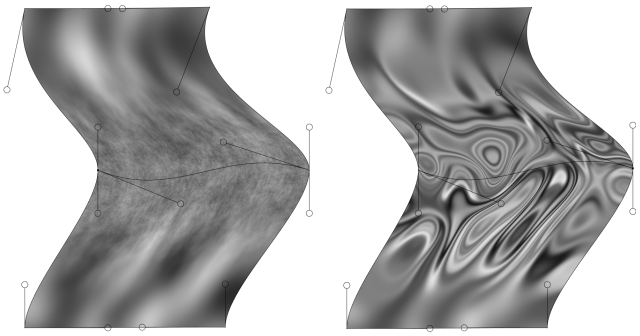


Figure 6. Two noisy gradient meshes showing varying Perlin noise fields. Left: Varying persistence values. Right: Varying distortion amounts. In both examples, the values/amounts are high in the middle, and low at the top and bottom.

handles to spatially distort the mesh and thus the associated noise field.

Additionally, there are global parameters to control the noise. The number of octaves of noise can be set for the entire mesh. For Worley noise, different distance metrics and functions for the Worley kernels can be chosen. Further, the global frequency, or scaling value, can be varied such that the uv-domain is scaled, resulting in smaller lattice cells and a higher frequency of the noise overall. For Gabor noise, we can vary the global kernel density in each cell of the lattice and the width of the kernels.

In Figure 6, we show several extreme effects which can be modelled with the noise-augmented gradient mesh. In this case, we show a simple gradient mesh, but from vertex to vertex we locally create a large difference between some of the parameter values. As can be seen in the figure, the noise smoothly varies from low to high frequency noise (left), and also from high amounts of distortion to no distortion at all (right).

5. Editing and Filtering

In this section, we describe several secondary filtering and manipulation methods we have applied to the noise fields on the gradient meshes.

5.1. Distortion

A simple value ranging in $[0, 1]$ is used to control the distortion amount. In this paper, we experimented with three types of distortion; see Figure 7. We use simple coordinate distortions, where the uv parameterisation is distorted by another (noise) function. We can also distort the noise values using the sine or another trigonometric function such that a wave-like characteristic is imposed on the noise field. In the same vein, we can create a wood-like texture by multiplying the resulting scalar by some function value and wrapping the values on a closed range. An example of distortion with a secondary noise function, with varying amount, is shown in Figure 6, right.

5.2. Filtering, Blending & Colour Mapping

We use five basic filtering functions to filter the noise field: pulse, high-pass, low-pass, band-pass, and normalised band-pass. Each function uses one or two threshold values which define the lower and upper cut-off value; see Figure 8.

The final colour is achieved by combining the noise colour with the colour of the gradient mesh. The blending of two colours can be performed in numerous ways, e.g. by using the traditional colour blending techniques commonly used in raster-based image editing software. Our tool offers 25 such blending options. This gives the user a lot of freedom to generate nicely-looking varying noise patterns, combined with smoothly changing colours.

We also offer the ability to map the value of the noise as a blending parameter between the two colours.

In Figure 9, we show how the different post-processing effects offer yet another possibility to increase the expressiveness of noisy gradient meshes. In this case, the Perlin noise field values are distorted by a sine function, and a band-pass filter is applied to decrease the width of the marble texture.

6. Results

We now present results obtained using our experimental noisy gradient mesh tool. Our tool can be seen in action in the supplementary video, and a screen-shot of its user interface is shown in Appendix. We also show results obtained by displacement maps governed by locally controlled noise linked to spline surfaces.

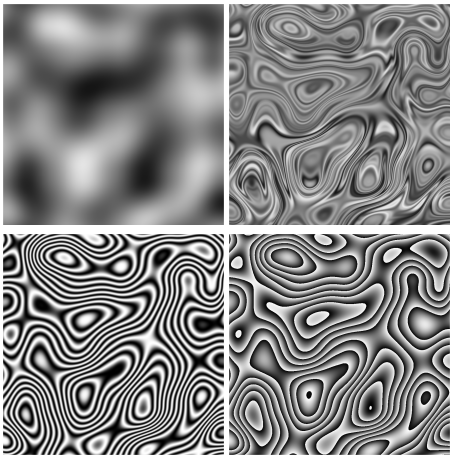


Figure 7. Three different types of distortion applied to a Perlin noise field (top left). Distortion by a secondary noise field (top right), sine distortion (bottom left), and wood-like distortion (bottom right).

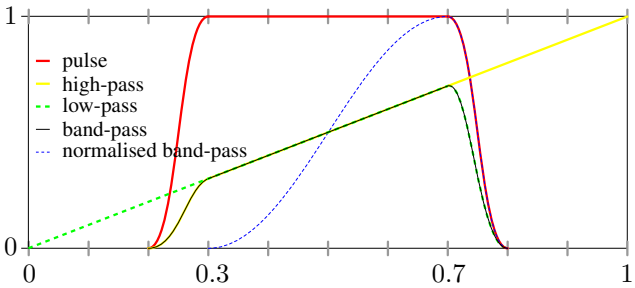


Figure 8. The five filtering functions used in our tool. The low and/or high cut-off values are, in this example, 0.3 and/or 0.7, respectively.

6.1. Noisy Gradient Mesh Results

We show several simple gradient meshes which model various objects in Figure 10. It can be seen from these results that the added procedural noise functions contribute an immense level of detail that would not be possible otherwise with ordinary gradient meshes, at least not with this low number of faces. We also allow for hard transitions in colour and/or noise fields by assigning different parameters not only per vertex, but also per patch of the mesh. The meshes can be scaled to any resolution, which would not have been possible had the noise patterns been defined using (high resolution) raster textures and texture mapping.

It is important to notice that, just like the interpolated colours, the noise field follows the flow of the mesh. This can be seen especially on the dark green bands of the watermelon (Figure 10) or on the pattern on the vase (Figure 9). This flow is the result of the mapping of the parameterisation domain onto the gradient mesh. In this way, the illusion of a volumetric object can be created on top of the effects of colour shading of ordinary gradient meshes. Had the parameterisation not been linked to the geometry of the mesh this effect would not have been visible, and an analogous effect to solid noise [10] on 3D surfaces would have occurred, which is undesirable. In Figure 11, we explicitly show the effect of altering the geometry of the gradient mesh (vertices and gradient handles) on the resulting blended noise and colour field. As can be seen in the figure, the pattern on the top wings is naturally influenced by the change of geometry.


Figure 9. Different steps in modelling a marble-like texture onto a gradient mesh of 12 faces using Perlin noise. Top left: The original colour field of the gradient mesh. Top right: The raw noise field. Bottom left: The noise field after post-processing. Bottom right: The final result, a blend of the post-processed noise field and the colour field.

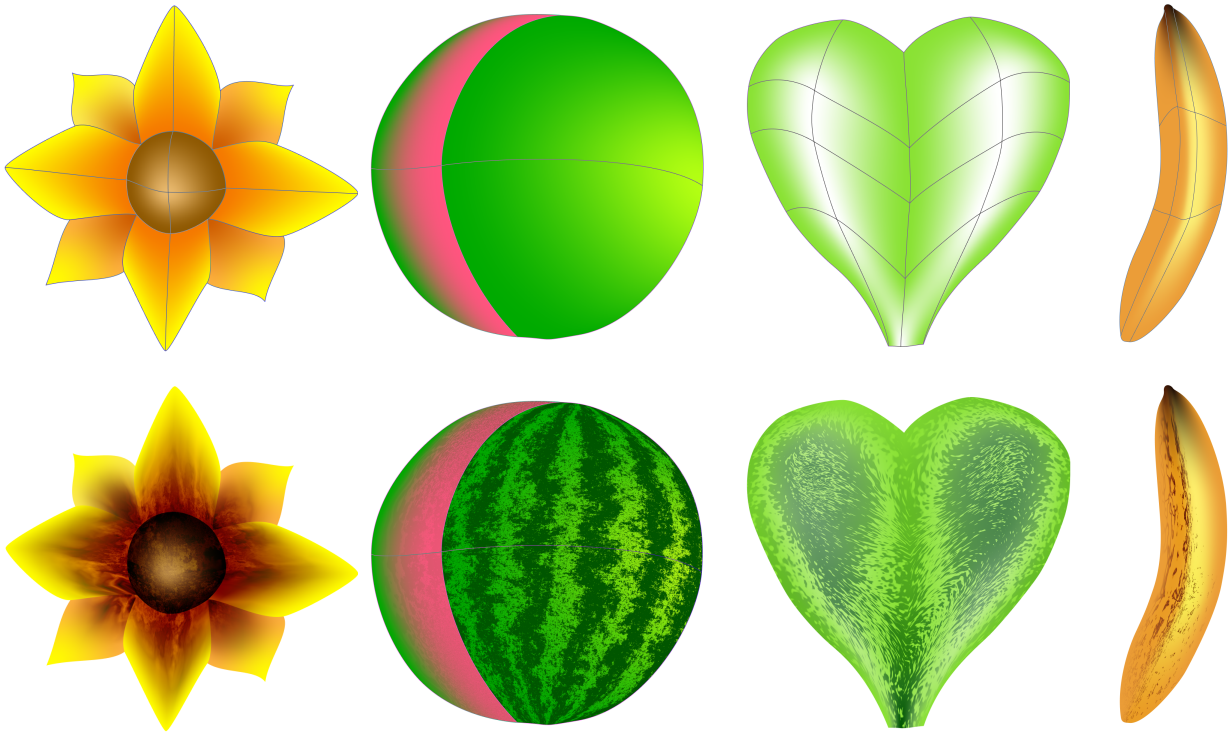
Another clearly visible effect is the varying noise pattern on the banana mesh or on the flower (Figure 10). The results clearly show that our approach provides smooth transitions from uniformly shaded regions to regions with noise, and from regions with low-frequency noise to higher frequencies.

6.2. Displacement Mapping

We applied the same concept of linking a procedural noise function to geometry, but this time to parametric spline patches and displacement mapping. In this case, we use uniform cubic B-splines as the geometry patch, but e.g. NURBS would work just as well. We convert the B-splines to the Bézier format so that we can attach the noise parameters to the control points of each Bézier patch in exactly the same manner as with noisy gradient meshes. In this way, we are sure that the noise parameters are interpolated rather than approximated by the B-spline, although the continuity of the noise parameters is only C^1, whereas the geometry is C^2 continuous. If desired, the interpolation property can be relaxed and the noise parameters can be approximated by a cubic B-spline, too, resulting in C^2 continuity of the parameters as well, but then without exact interpolation.

Two results can be seen in Figure 12. We vary the persistence of the frequency component along the u and v (parametric) directions of the mesh, and the noise is used to displace the geometry along its normal field. As can be seen from the result, the frequency signal of the displacement smoothly varies over the entirety of the mesh. The technique could be applied to generate terrain based on a sparse representation of the geometry, e.g. a B-spline surface, and details can be varied locally to model terrains with different features. In this way, a designer has more control over the procedural generation of terrains.

Similarly, one could apply the procedural noise with local control on parametric geometries to modify their normal field (normal mapping), but also other characteristics such as specularity or transparency.



Flower (16 patches) Watermelon (4 patches) Leaf (16 patches) Banana (9 patches)

Figure 10. Simple gradient meshes with different types of noise. Top row: Ordinary gradient meshes (with sharp colour transitions). Bottom row: Noisy gradient meshes. The flower model makes use of Worley noise, banana and watermelon meshes make use of Perlin noise, and Gabor noise was used on the leaf mesh.

7. Discussion

We have described only several opportunities to manipulate noisy gradient meshes; the possibilities are rather endless. In theory, any scalar parameter value of a procedural noise function can be interpolated. However, this depends on the nature of the different parameters and the precise evaluation of the procedural noise function on graphics hardware. An even more elaborate system based on block shaders [2] could be used to create more complex textures. It would also be possible to have local weights for each noise parameter. For instance, different types of procedural noise functions could be combined by defining a weight for each type per vertex, and then interpolating these over the mesh using blending based on the interpolated weights.

Our approach makes extensive use of programmable shaders. We use tessellation shaders to evaluate the gradient primitives and thus the interpolation of the noise parameters. Fragment shaders handle the evaluation of the procedural noise functions, based on the interpolated parameters. The use of graphics hardware makes it possible to render the meshes in real-time such that interactive editing of the meshes is possible (see the supplementary video).

Even in the case of a full HD pixel-dense rendering of a mesh with 3600 primitives and Gabor noise with a kernel density of 20, we are still able to render the noisy gradient mesh at sub-25 millisecond rates. The same mesh renders at sub-10 millisecond rates for Perlin and Worley noise with 10 octaves of fractal noise. The performance was evaluated on an NVIDIA GTX Titan V. It should be noted that by evaluating the noise functions in fragment shaders, Gabor noise is not continuous over the whole lattice as the support of the Gaussian kernel is truncated for each cell of the lattice. Visually, this can be expected to be hardly noticeable [11], as also our results confirm. The other two considered types of noise, Perlin and Worley, vary smoothly over the mesh.

The approach in this paper works directly only for quadrilateral meshes with regular topologies. It should be possible to support T-junctions similarly to locally refinable gradient meshes [3] as long as the gradient meshes remain rectangular and enforce C^1 continuity. For meshes with arbitrary manifold topology [13], the geometry should have at least C^1 continuity in order to avoid visible discontinuities or unexpected distortions in the generated noise fields. We have experimented with lowering the condition of C^1 parameter functions to G^1, but in this way the parameterisation of the noise functions is only C^0, showing hard transitions in the generated noise fields. The same argument

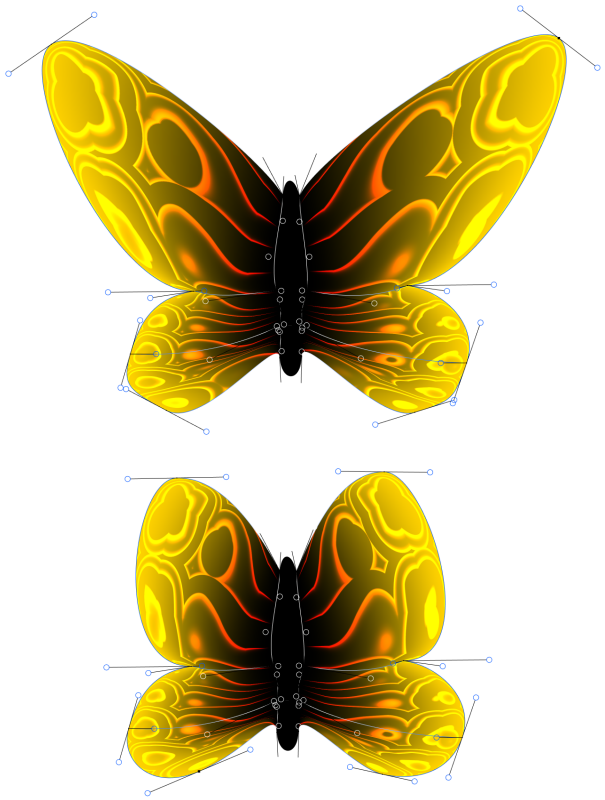


Figure 11. Top: An artistic depiction of a butterfly based on Worley noise. Bottom: The geometry (i.e., vertex positions and gradients) has been edited to modify the shape of the wings. Note that the noise pattern, whose parameters have not been altered, naturally follows the transformation.

applies to displacement mapping of arbitrary topology surfaces. However, if the mesh is sufficiently dense the discontinuities will not be visible.

It can be difficult to reproduce noise patterns by hand as faithfully as possible using the provided parameters and post-processing methods. However, the noise functions can easily model arbitrary patterns like the imperfections on fruit or blemishes on the skin of a human. This breaks the clean appearance of gradient meshes and the resulting image has a more natural look. Naively overlaying a noise field over an image does not produce the same effect as our approach does as we can guide the noise by changing the geometry of the mesh, as demonstrated in Figure 11.

8. Conclusion

In this paper, we have presented noisy gradient meshes: a combination of procedural noise functions with gradient meshes. There is a direct mapping of lattice-based noise functions onto gradient meshes as they both rely on a grid-like structure. The interpolation of local noise parameters and the linking of the parameterisation domain of the noise functions and the gradient mesh makes it so that the

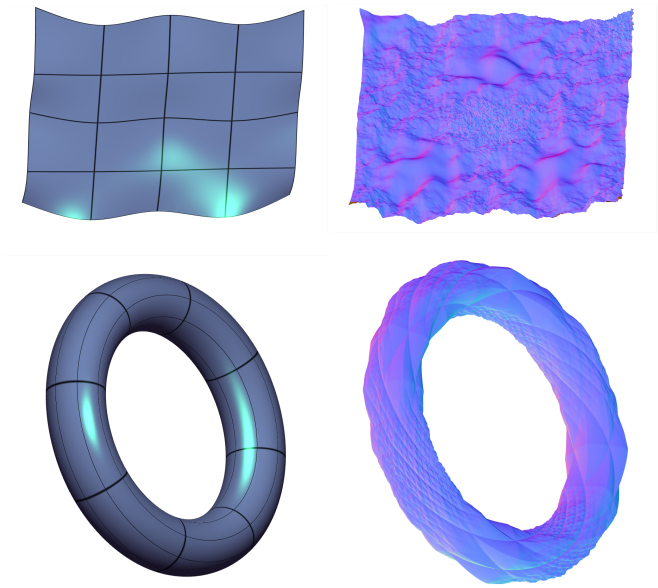


Figure 12. A B-spline surface composed of 16 patches (top row) and a B-spline torus composed of 64 patches (bottom row), both without displacement (left column) and with procedural displacement (right column). The final 'noisy' surfaces are colour-coded according to their normal fields.

noise field is distorted by moving the control points or adjusting the gradients of the mesh. The evaluation of the noise functions is quick and the possibilities are endless, especially when considering the composition of different noise functions and post-processing editing functions such as colour blending modes and different types of distortion. The method is simple, yet effective as the user can create highly detailed and scalable images using a small number of patches.

Future work includes combining our noisy gradient meshes with vectorisation approaches similar to [9]. Further extensions might be to generalise the technique to also allow for local refinement and T-junctions, in order to be able to create even sparser noisy gradient meshes.

Acknowledgements

This research is in part based on the second author's Bachelor thesis at the Anonymous Institution. We would like to thank Steven Gustavson for his publicly available implementations of Perlin and Worley noise, and Victor Shepardson for his implementation of Gabor noise.

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research.

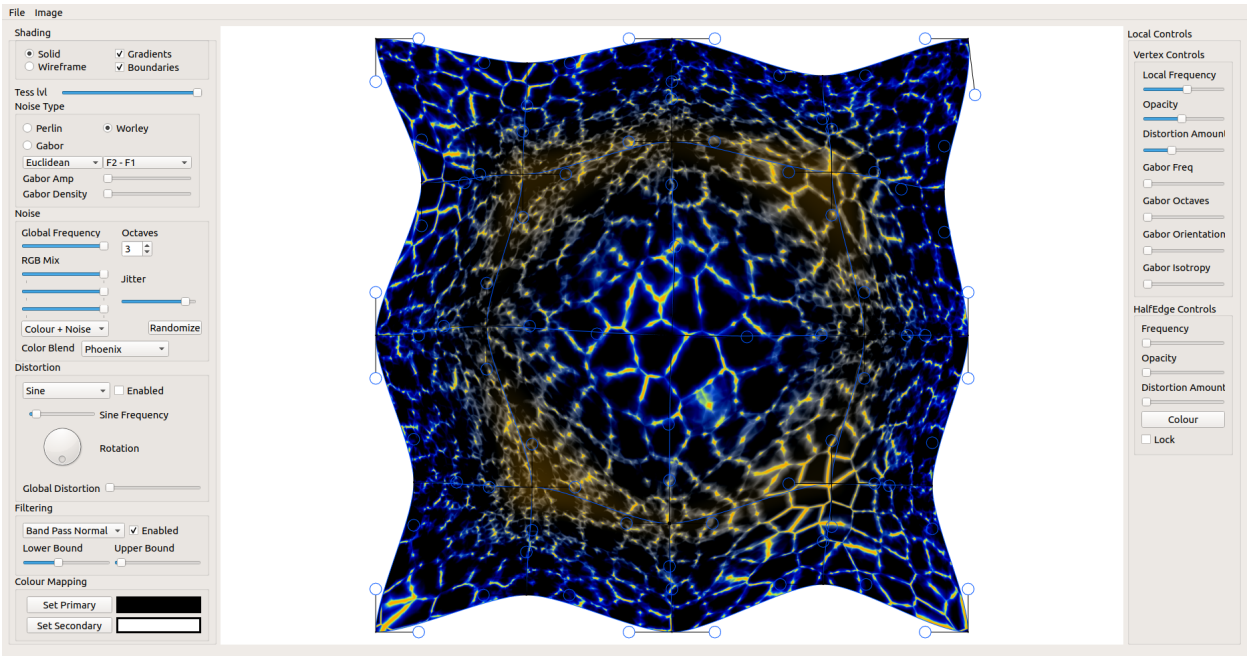


Figure 13. A screen-shot showing our experimental noisy gradient mesh tool.

References

[1] Mesh gradients - Inkscape. http://wiki.inkscape.org/wiki/index.php/Mesh_Gradients. Accessed: 2018-09-21. 2

[2] G. D. Abram and T. Whitted. Building block shaders. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, pages 283–288, New York, NY, USA, 1990. ACM. 7

[3] P. J. Barendrecht, M. Luinstra, J. Hogervorst, and J. Kosinka. Locally refinable gradient meshes supporting branching and sharp colour transitions. *The Visual Computer*, 34(6):949–960, Jun 2018. 2, 7

[4] P. Barla and A. Bousseau. Gradient art: Creation and vectorization. In *Image and Video-Based Artistic Stylisation*, pages 149–166. Springer, 2013. 1

[5] F. D. Coburn and P. McCormick. *CorelDRAW 9: The Official Guide*. Osborne McGraw-Hill, 2017. 2

[6] D. S. Ebert, F. K. Musgrave, D. Peachey, K. Perlin, and S. Worley. *Texturing & modeling: a procedural approach*. Morgan Kaufmann, 2003. 1

[7] J. Ferguson. Multivariable curve interpolation. *Journal of the ACM (JACM)*, 11(2):221–228, 1964. 2

[8] H. Hnaidi, E. Guérin, S. Akkouche, A. Peytavie, and E. Galin. Feature based terrain generation using diffusion equation. In *Computer Graphics Forum*, volume 29, pages 2179–2186, 2010. 2, 4

[9] S. Jeschke, D. Cline, and P. Wonka. Estimating color and texture parameters for vector graphics. In *Computer Graphics Forum*, volume 30, pages 523–532, 2011. 2, 8

[10] A. Lagae, S. Lefebvre, R. Cook, T. Derose, G. Drettakis, D. S. Ebert, J. P. Lewis, K. Perlin, and M. Zwicker. State of the art in procedural noise functions. In *EG 2010-State of the Art Reports*. The Eurographics Association, 2010. 3, 6

[11] A. Lagae, S. Lefebvre, G. Drettakis, and P. Dutré. Procedural noise using sparse gabor convolution. *ACM Transactions on Graphics (TOG)*, 28(3):54, 2009. 3, 7

[12] Y.-K. Lai, S.-M. Hu, and R. R. Martin. Automatic and topology-preserving gradient mesh generation for image vectorization. In *ACM Transactions on Graphics (TOG)*, volume 28, page 85. ACM, 2009. 2

[13] H. Lieng, J. Kosinka, J. Shen, and N. A. Dodgson. A colour interpolation scheme for topologically unrestricted gradient meshes. In *Computer Graphics Forum*, volume 36, pages 112–121, 2017. 2, 7

[14] A. Orzan, A. Bousseau, H. Winnemöller, P. Barla, J. Thollot, and D. Salesin. Diffusion curves: a vector representation for smooth-shaded images. In *ACM Transactions on Graphics (TOG)*, volume 27, pages 92:1–92:8. ACM, 2008. 2

[15] K. Perlin. An image synthesizer. *SIGGRAPH Comput. Graph.*, 19(3):287–296, July 1985. 2

[16] J. Sun, L. Liang, F. Wen, and H.-Y. Shum. Image vectorization using optimized gradient meshes. In *ACM Transactions on Graphics (TOG)*, volume 26, pages 11:1–11:7. ACM, 2007. 2

[17] J. J. Van Wijk. Spot noise texture synthesis for data visualization. *ACM Siggraph CG*, 25(4):309–318, 1991. 4

[18] T. W. Verstraaten and J. Kosinka. Local and hierarchical refinement for subdivision gradient meshes. *Computer Graphics Forum*, 37(7):373–383, 2018. 2

[19] B. Wood. *Adobe Illustrator CC Classroom in a Book*. Adobe Press, 2017. 2

[20] S. Worley. A cellular texture basis function. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 291–294. ACM, 1996. 3

Appendix - Editing Tool

For this paper, we have built our own experimental noisy gradient mesh tool, in which one can design and edit noisy gradient meshes. By dragging and dropping vertices and tangent handles, the geometry of the gradient mesh can be manipulated. Colours can be assigned to vertices, or to edges to create sharp transitions in colour. Likewise, local noise parameters can be supplied to vertices or to edges via the right panel of the GUI; see Figure 13. Global parameters for the noise functions can be set in the left panel, which also provides several rendering options such as tessellation level and whether to render mesh boundaries and gradient handles.