A Designer Centric Procedural Texture Generator Using Modular Bivariate Quadratic Functions

Ian Parberry

Technical Report LARC-2015-05

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

July, 2015

A Designer Centric Procedural Texture Generator Using Modular Bivariate Quadratic Functions

Ian Parberry
Dept. of Computer Science & Engineering
University of North Texas

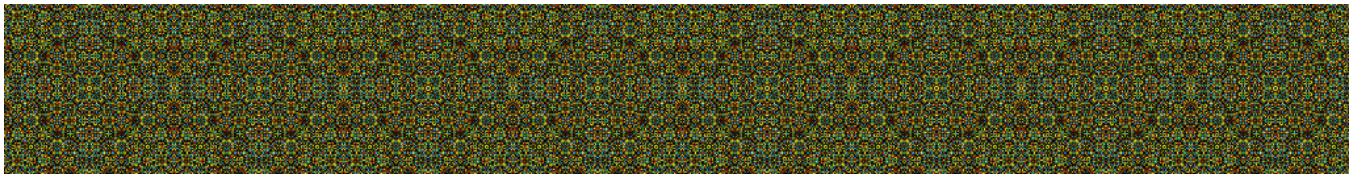

Figure 1: *Procedural ornamentation with pixel* (x, y) *generated using* $38(x^2 + y^2) \bmod 129$ *with four colors.*

Abstract

Modular bivariate quadratic functions are used to procedurally generate textures that resemble ornamentation, skin, scales, feathers, and textiles. They are easy to implement and particularly suited to parallel execution as a pixel shader. A prototype browser-based procedural texture generator with an interface suitable for use by non-mathematicians such as designers and artists is described.

1 Introduction

Applications of virtual reality such as video games require a large number of of bitmapped images called *textures* that are mapped onto 3D surfaces. *Procedural texture generation* refers to the generation of textures using algorithms and mathematical functions as opposed to having artists create them by hand. The obvious approach to procedural texture generation involves mathematically modeling the processes that produce materials in physical world, for example, modeling the geometry of individual threads in a warp knitted fabric using the technique of [Renkens and Kyosev 2010]. However, in practice that would be unnecessarily difficult and expensive. A more viable strategy requiring a more reasonable amount of effort by computer and human is to seek, as Musgrave puts it, "semblance over veracity" [Musgrave 1999a].

Related work includes computer generated ornamentation; for example, using Lindenmayer systems to generate floral ornamentation [Wong et al. 1998], and a discussion of architectural ornamentation [Whitehead 2010]. There is much interest in procedural textile generation in the craft community [Williams 2015]. The techniques used there usually focus on the generation of clean, regular patterns using standard concepts such as tessellation, maze generation, cellular automata, and Latin squares. In contrast, the patterns that we generate will incorporate randomness and low-level noise while keeping to a visual structure provided by discrete mathematics.

Musgrave describes an iterative process in which a texture designer experiments with combinations of various standard algorithms until the desired effect is achieved [Musgrave 1999a]. He views this hit-or-miss process as a natural combination of experimental science and engineering. Our aim is to provide a tool that the designer can use to explore various types of texture. As the author of this paper has pointed out elsewhere [Doran and Parberry 2010], design tools such as this must have controls that are designer-centric in the sense that they are natural to the problem domain as opposed to the underlying mathematical model.

Musgrave also points out the advantages of *point evaluation* methods in which each texture pixel, or *texel* is computed independently without any reference to other texels [Musgrave 1999b]. Many point evaluation methods are based on *Perlin noise* [Perlin 1985] or the related *value noise* [Peachey 1999] p. 67, which computes a smooth noise value at each pixel using a small number of arithmetic operations. Pallister, for example, shows how to use Perlin noise to create cloud textures [Pallister 2001]. Some procedural texture generation algorithms are based on post processing the output of a point evaluation method, for example, Perlin generated a marble-like texture using the sine of Perlin noise [Perlin 1985].

We describe modular bivariate quadratic functions, which are quadratic functions of two variables over the integers modulo q for some choice of q, and show that they can be used to quickly and easily generate textures that resemble skin, scales, feathers, textiles, and ornamentation. For example, in Figure 1, pixel (x, y) has grayscale intensity $38(x^2+y^2) \bmod 129$, which is then mapped to a 4-color palette. Modular bivariate quadratic functions can also be used to generate some striking optical illusions that should probably be avoided in most applications.

The remainder of this paper is divided into four short sections. Section 2 begins with the mathematics of modular bivariate quadratic functions and their application to procedural texture generation. Section 3 continues with some implementation notes. Section 4 describes various styles of texture generated with modular bivariate quadratic functions. Section 5 contains a description of an online texture generation tool for artists and designers that can be used with little or no knowledge of the mathematics of modular bivariate quadratic functions. For completeness, an Appendix provides proofs of the mathematical results used in the body of this paper.

2 Modular Bivariate Quadratic Functions

Let $\mathbb{N} = \{0, 1, 2, \dots\}$ denote the set of natural numbers, and let $[0, 1]$ denote the set of floating point numbers x such that $0 \leq x \leq 1$. Let $\pi : \mathbb{N}^2 \to \mathbb{N}$ be the bivariate quadratic function defined by:

$$\pi(x, y) = ax^2 + by^2 + cxy + dx + ey + f.$$

for some $a, b, c, d, e, f \in \mathbb{N}$. Suppose $p, q \in \mathbb{N}$ where $q \geq 2$ and $1 \leq p < q$. Let $h : \mathbb{N}^2 \to [0, 1]$ be the bivariate quadratic function modulo q defined by:

$$h(x, y) = \frac{\pi(px \bmod q, py \bmod q) \bmod q}{q - 1}.$$

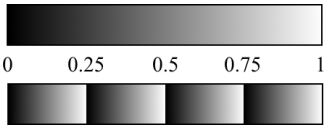

Figure 2: *A grayscale value in the range* $[0, 1]$ *(above) is mapped into four grayscale bands (below).*

When $d = 1$ and all other coefficients are zero, $h(x, y) = px \bmod q$ is known as the *Lehmer hash function* after [Lehmer 1954], in which case it is usually recommended that q be a prime power and p be a primitive root modulo q (for example, $p = 48271$ and $q = 2^{31} - 1$ [Park et al. 1988]). However, we will allow p and q to be arbitrary positive integers that may have a common factor.

Textures are constructed from modular bivariate quadratic functions by assigning each texel at integer coordinates (x, y) the grayscale value $h(x, y) \in [0, 1]$, with zero denoting black and unity denoting white. The resulting texture is composed of $q \times q$ tiles, that is, it repeats on the horizontal and vertical axis every q pixels, since for all $x, y \in \mathbb{N}$,

$$\pi(p(x + q), py) \equiv \pi(px, p(y + q)) \equiv \pi(px, py) \pmod{q}.$$

(For a proof, see Theorem 1 in the Appendix.)

It will have even smaller tile size if p and q share a common factor. If $x, y \in \mathbb{N}$, let $\gcd(x, y)$ denote the *greatest common divisor* of x and y, that is, $\gcd(x, y)$ is the largest $d \in \mathbb{N}$ such that both $x/d, y/d \in \mathbb{N}$. For convenience, let $\hat{p} = p/\gcd(p, q)$ and $\hat{q} = q/\gcd(p, q)$. Note that both $\hat{p}, \hat{q} \in \mathbb{N}$. The resulting texture will be composed of $\hat{q} \times \hat{q}$ tiles since for all $x, y \in \mathbb{N}$,

$$\pi(p(x + \hat{q}), py) \equiv \pi(px, p(y + \hat{q})) \equiv \pi(px, py) \pmod{q}.$$

(For a proof, see Theorem 3 in the Appendix.)

More texture variations can be obtained by dividing the grayscale range $[0, 1]$ into two or more bands, replacing grayscale value v with $vb - \lfloor i/b \rfloor$. For example, Figure 2 shows a grayscale value being mapped into four bands. Changing the number of bands can change the resulting texture markedly. Color can be introduced by mapping each band into a color drawn from a small palette. Different palettes may bring out different features in the texture.

3 Implementation

$h(x, y)$ can be computed using a small number of arithmetic operations: at most 7 integer multiplications, 3 integer remainders, 5 integer additions a floating point addition, and a floating point division. Since h is a point evaluation function (after [Musgrave 1999b]), that is, $h(x, y)$ can be computed independently of $h(x', y')$ for all $x' \neq x$, $y' \neq y$, it is ideal for parallelization using a pixel shader. The pixel shader can be used to generate a texture image file, or the shader itself can be built into the application.

The advantage of direct implementation of the pixel shader is that there is no need for disk space or video memory to store the textures, nor is there any loading time. The textures can therefore be very large, which eliminates the need for any of the standard techniques (such as stochastic Wang tiling [Cohen et al. 2003]) used to disguise the regular patterns that tend to occur with small tiles. The disadvantage is that there is unnecessary recomputation when textures are repeated (particularly with small tiles), and that mipmapping needs to be done at runtime in the shader instead of using precomputed mipmaps.

When this algorithm is implemented in HLSL using Shader Model 4 under DirectX 11.2, a 1024×1024 texture can be generated at approximately 4000 frames per second on an Intel Core i7-3930K @ 3.2GHz with an NVidia GeForce GTX 660 Ti running Windows 8.1. This means that a complete 1024×1024 texture is generated in less than 250μsec. In contrast, when implemented in JavaScript using HTML 5 and CSS 3, it takes on the order of a second to generate an 800×600 texture in the Chrome browser on the same computer.

4 Texture Styles

This section examines some examples of the textures that can be generated with modular bivariate quadratic functions. It is divided into four subsections on, respectively, ornamentation; skin, scales and feathers; textiles; and others.

4.1 Ornamentation

Suppose we take $\pi(x, y) = x^2 + y^2$. We can restrict p to be at most $q/2$ without loss of generality because for each q and $r < q$, the same texture is generated for both $p = r$ and $q - r$. (For a proof, see Theorem 4 in the Appendix.) The term *rectilinear ornamentation* is appropriate for this style of texture because it is comprised of $\hat{q} \times \hat{q}$ tiles that are symmetric about the horizontal and vertical axes since for all $x, y \in \mathbb{N}$,

$$\pi(p(\hat{q} - x), py) \equiv \pi(px, p(\hat{q} - y)) \equiv \pi(px, py) \pmod{q}.$$

(For a proof, see Theorem 5 in the Appendix.)

The result of this biaxial symmetry is pleasing to the eye for many choices of p and q. Figure 3 shows three examples of rectilinear ornamentation using $q = 131$ and $p = 13, 15, 22$. Each image consists of four identical 131×131 tiles arranged in a 2×2 grid. Figure 4 shows the rightmost texture in Figure 3 mapped to 2, 3, and 4 grayscale bands. Figure 5 shows an example of rectilinear ornamentation with 3 different color palettes.

Rectilinear ornamentation textures may have tile size even smaller than $\hat{q} \times \hat{q}$. Suppose 0 is a quadratic residue in the multiplicative group of integers modulo q, that is, there exists $0 < r \leq q/2$ such that $r^2 \equiv 0 \pmod{q}$. Note that q must be a composite number, since $r|q$. Further suppose that $r|2p^2$. Then any rectilinear ornamentation texture will have tile size $r \times r$ since for all $x, y \in \mathbb{N}$,

$$\pi(px, py) \equiv \pi(p(x + r), py) \equiv \pi(px, p(y + r)) \pmod{q}.$$

(For a proof, see Theorem 6 in the Appendix.) For example, the rectilinear ornamentation texture with $p = 84$ and $q = 512$ has a tile size of 32×32 (which is much smaller than $\hat{q} \times \hat{q}$, that is, 128×128) because $32^2 = 1024 \equiv 0 \pmod{512}$ and $2 \times 84 \times 84 = 14112 = 32 \times 441$.

Taking $\pi(x, y) = x^2 + y^2 + xy$ results in what might be called *diagonal ornamentation* because the $\hat{q} \times \hat{q}$ tiles are symmetric about the diagonal since for all $x, y \in \mathbb{N}$,

$$\pi(p(\hat{q} - x), p(\hat{q} - y)) \equiv \pi(px, py) \pmod{q}.$$

(For a proof, see Theorem 7 in the Appendix.) Figure 6 shows three examples of diagonal ornamentation using $q = 131$ and $p = 13, 15, 22$.

4.2 Skin, Scales and Feathers

Suppose we take $\pi(x, y) = x^2 + y^2 + 2xy + x + 3y$. The xy term results in a diagonal pattern of curves that often resemble skin, scales,

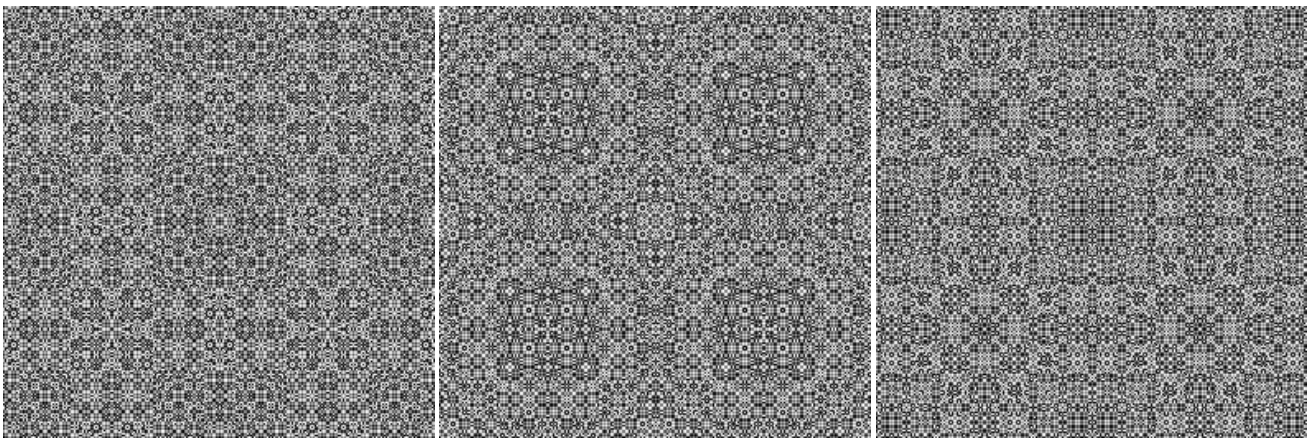

Figure 3: *Rectilinear ornamentation* $(x^2 + y^2)$ *with* $q = 131$, $p = 11, 46, 47$ *(left to right).*

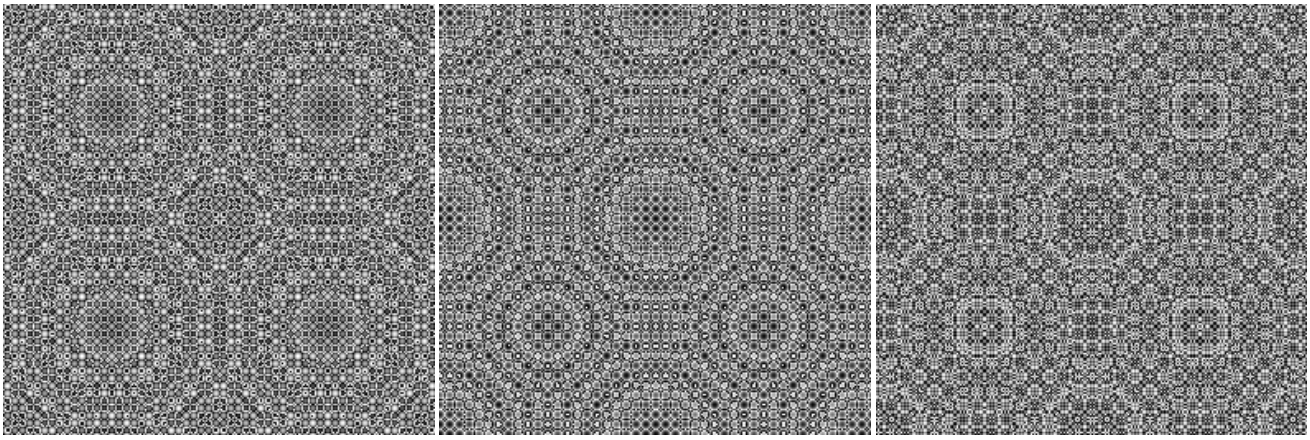

Figure 4: *Rectilinear ornamentation* $(x^2 + y^2)$ *with* $q = 131$, $p = 22$ *using (from left to right) 2, 3, and 4 grayscale bands. Compare this with the rightmost image in Figure 3, which uses a single band.*

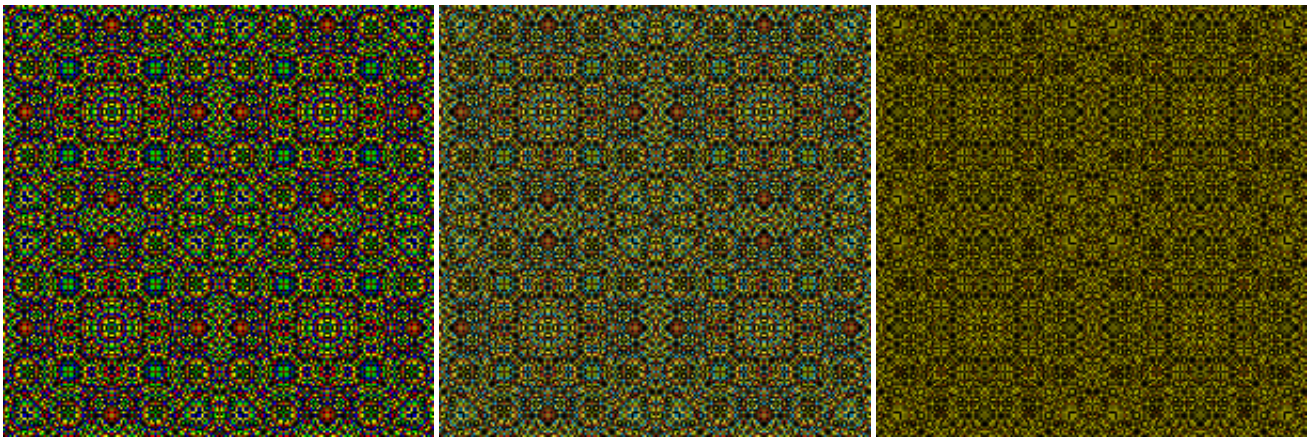

Figure 5: *Rectilinear ornamentation* $(x^2 + y^2)$ *with* $q = 101$, $p = 4$ *using 4 colors with various palettes.*

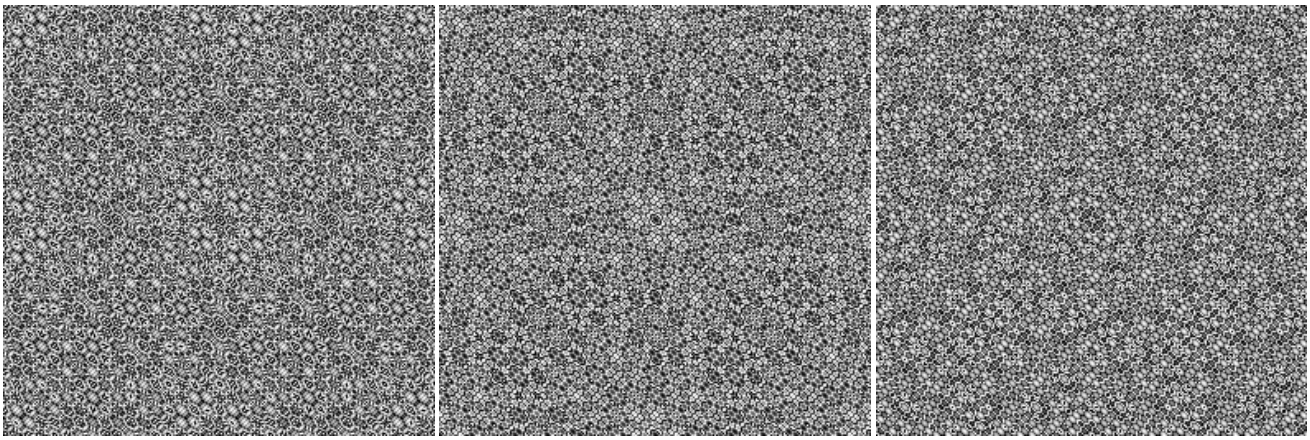

Figure 6: *Diagonal ornamentation* $(x^2 + y^2 + xy)$ *with* $q = 131$*,* $p = 11, 46, 47$ *(left to right).*

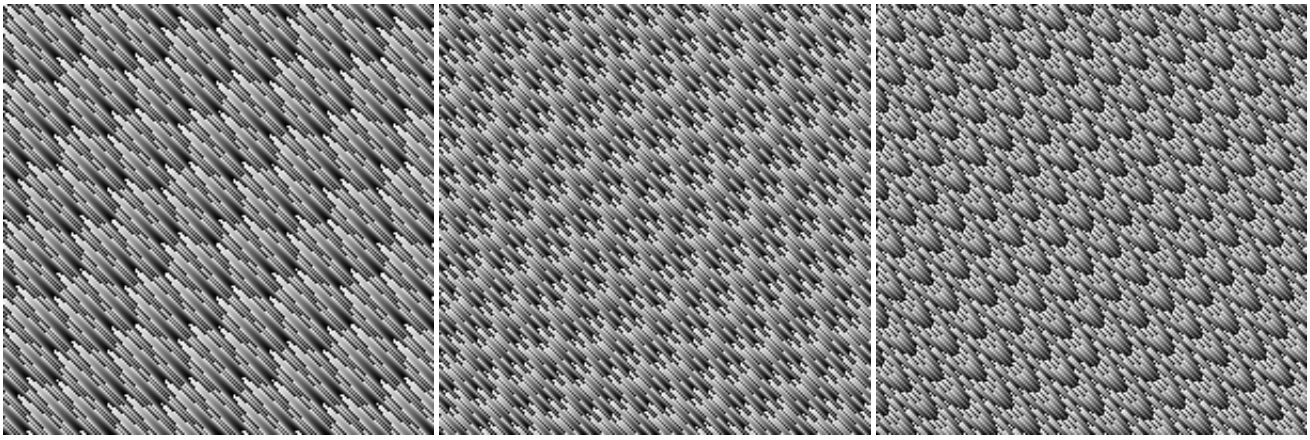

Figure 7: *Scales and feathers* $(x^2 + y^2 + xy)$ *with* $q = 101$*,* $p = 2, 3, 4$ *(left to right).*

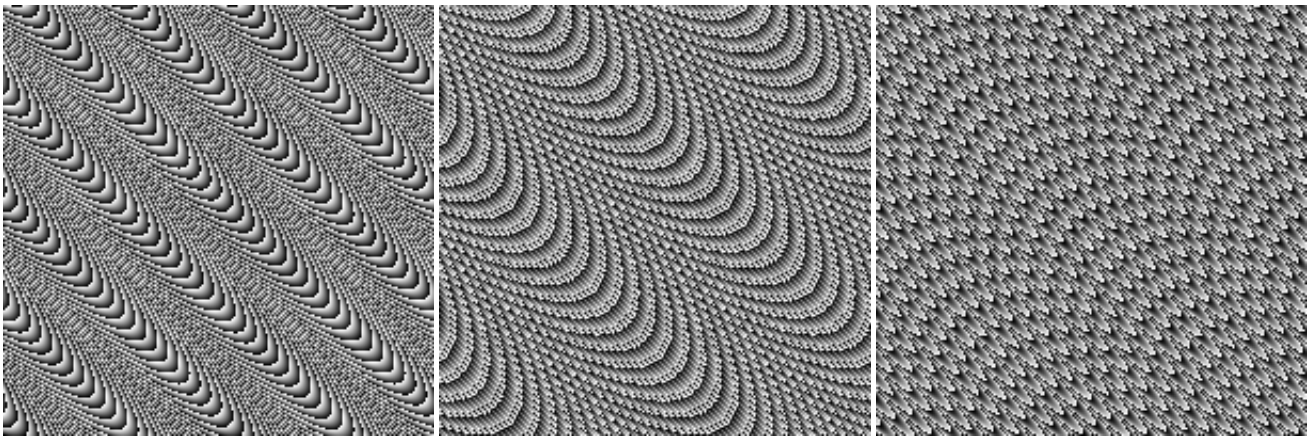

Figure 8: *Optical illusions* $(x^2 + y^2 + xy)$ *with* $q = 101$*,* $p = 10, 44, 45$ *(left to right).*

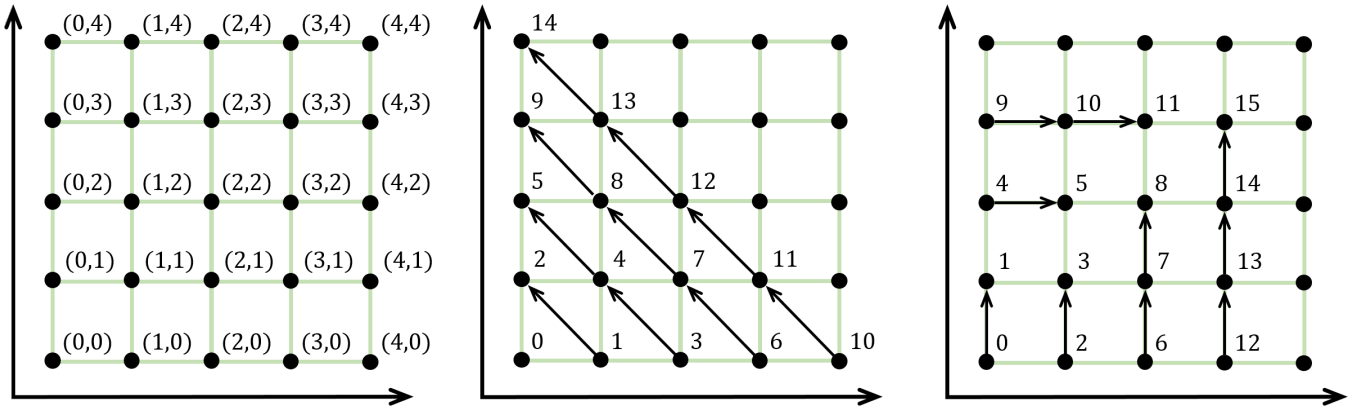

Figure 9: *Left: A one-way infinite two-dimensional grid with the Cartesian coordinates of the points. Center: The points numbered by the Cantor pairing function. Right: The points numbered by the Szudzik pairing function.*

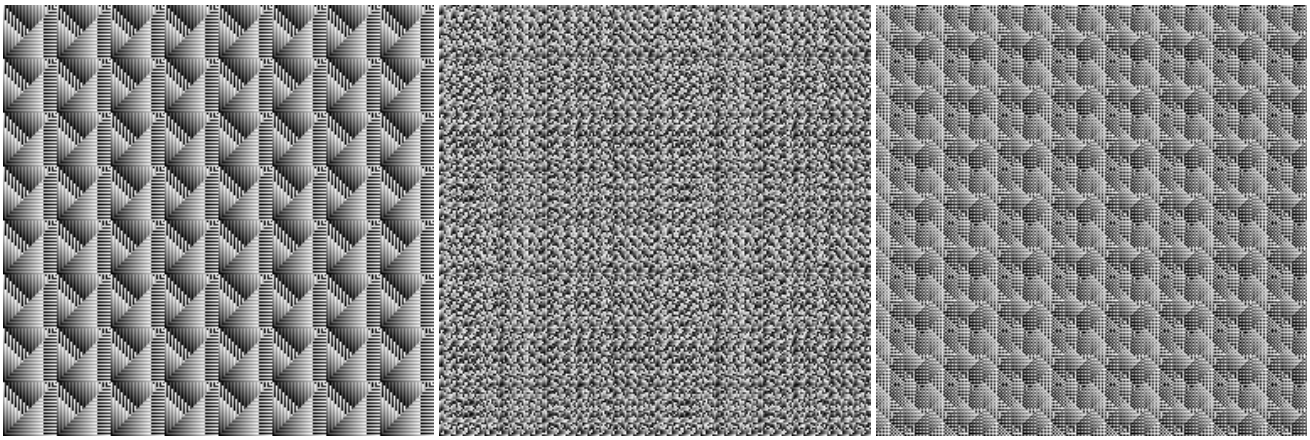

Figure 10: *Textiles* $(x^2 + x + y), (y^2 + x)$ *with* $q = 256$, $p = 8, 38, 136$ *(left to right).*

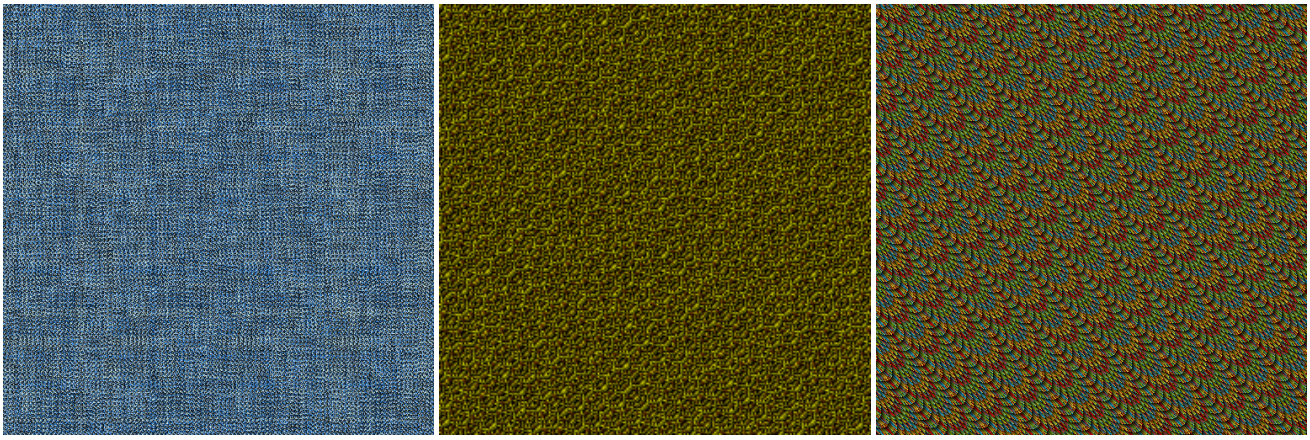

Figure 11: *Left: textile with* $p = 5260$, $q = 26509$. *Center: scales with* $p = 11$, $q = 217$. *Right: feathers with* $p = 2101$, $q = 4319$.

or feathers. Figure 7 shows three examples using $q = 101$ and $p = 2, 3, 4$. Each image consists of four 101×101 tiles arranged in a 2×2 grid. The bivariate quadratic $x^2+y^2+2xy+x+3y$ is double the *Cantor pairing function* $\pi_c(x, y) = (x + y)(x + y + 1)/2 + y$, which was used by Cantor in the 19th century to show that the rational numbers are countable [Cantor 1878].

These parameters will also occasionally generate an optical illusion that tricks the eyes into seeing colors and three-dimensionality. The effect will, however, vary with the individual observer. Figure 8 shows three examples using $q = 101$ and $p = 10, 44, 45$. Since the colors change when the page is rotated about an axis in the plane of the page, the presence of a reflective diffraction grating is indicated.

Theory predicts that a diffraction grating with black stripes of width d will refract light of wavelength λ by an angle of $\sin^{-1}(\lambda/d)$ (considering first order diffraction only). Taking $d = 250\mu$m (approximately the pixel width of a 30 inch 2560×1600 monitor) and $\lambda = 600$nm (the wavelength of yellow light), this gives a deflection angle of $\sin^{-1}(0.6/250) = 0.1375^\circ$. Therefore, if the image is viewed at 500mm from the eye, yellow light is deflected by $500 \tan 0.1375^\circ \approx 1.2$mm, sufficient to explain the effect.

4.3 Textiles

Dissatisfied with the Cantor pairing function $\pi_c(x, y)$ mentioned in Section 4.2, Matthew Szudzik defined the *Szudzik pairing function* $\pi_s(x, y)$, as follows [Szudzik 2006]:

$$\pi_s(x, y) = \begin{cases} y^2 + x & \text{if } x < y \\ x^2 + x + y & \text{otherwise.} \end{cases}$$

Notice that while the Cantor pairing function numbers the grid points (Figure 9, left) in diagonal stripes (Figure 9, center), the Szudzik pairing function numbers them in vertical and horizontal stripes (Figure 9, right).

We generalize the Szudzik pairing function as follows. Suppose $h_1, h_2 : \mathbb{N}^2 \to [0, 1]$ are bivariate quadratic functions modulo q. Then,

$$h_s(x, y) = \begin{cases} h_0(x, y) & \text{if } x < y \\ h_1(x, y) & \text{otherwise.} \end{cases}$$

As above, we use $h_s(x, y)$ as a grayscale value for each pixel at coordinates (x, y). Figure 10 shows three examples using Szudzik's pairing function (i.e. $h_0(x, y) = x^2+x+y$ and $h_1(x, y) = y^2+x$) with $q = 256$ and $p = 8, 38, 123$.

4.4 Others

The modular bivariate quadratic functions used above are probably not all of the interesting ones, and they are certainly not necessarily the only ones that give the type of results described. For example, slight variants can be obtained by varying the value of f. We leave further exploration of the possibilities to the interested reader.

5 A Texture Design Tool

A designer in search of a texture needs to explore the parameter space of the modular bivariate quadratic functions until an appealing texture is found. Preferably he or she should be provided with a tool that lets them quickly and easily explore interesting possibilities without the mathematics getting in the way. The author has created a prototype for such a design tool called *Biq* (see Figure 12).

Biq runs in a web browser and is implemented in JavaScript, HTML 5, and CSS 3. It is available online at `http://larc.unt.edu/ian/research/texturegen/designtool/`. The current texture is drawn in the large box at right of the window, and the controls run down the left-hand side. The user sets the controls and clicks on the "Draw" button at bottom right to recompute the texture.

At the top of the controls there is a pair of edit boxes for the parameters p and q. These are named using the designer centric terms "Variant" and "Repeat", respectively. Below them is a read-only text box labeled "Tile size", the value of which ($\hat{q} = q/\gcd(p, q)$, see Section 2) is recomputed automatically every time the used clicks on the "Draw" button.

The texture style (ornamentation, scales and feathers, or textiles) is selected using the radio buttons immediately below the "Tile size" text box. The mathematical underpinnings of the algorithm in the form of the coefficients of the modular bivariate quadratic functions are shown below these, but are disabled in this mode. They are enabled, along with the radio buttons labeled "Diagonal" and "Linear" for Cantor and Szudzik style numbering respectively, only when the designer selects the "Advanced" style radio button.

The color is controlled by the color controls immediately below the coefficients. The number of color bands (from $1 - 4$) is selected using the "Colors" edit box. Since the user may not be artistically inclined, there are 8 custom palettes provided, in addition to the default "Grayscale" and the advanced "Custom" setting. These are selected using a set of radio buttons. If "Custom" is selected, then clicking on one of the four rectangular color controls will pop up a color picker (the details of which are browser dependent). Once chosen, the colors are displayed in the color controls.

When the designer finds an interesting texture, he or she can right-click on the image and save it into a file. Checking the "Draw single tile only" checkbox next to the "Draw" button enables the drawing and saving of a single tile. Below the texture image and the control panel there is a gray box that contains a link to Biq with parameters that reproduce the current texture so that the user can reproduce it at a later date.

6 Conclusion

We have seen how to generate textures resembling textiles, scales, feathers, and ornamentation using modular bivariate quadratic functions using a prototype design tool named Biq that can be used without knowledge of the underlying mathematics. Further work that could be attempted include further development of the design tool, and an exploration of the mathematical properties of modular bivariate quadratic functions. An extension to 3D textures is obvious using modular trivariate quadratic functions, which have the form:

$$\pi(x, y, z) = ax^2+by^2+cz^2+dxy+eyz+fxz+gx+hy+iz+j.$$

The corresponding design tool should be called Triq. The details are left to the interested reader.

Supplementary material including images, the online prototype of Biq, and source code can be found at `http://larc.unt.edu/ian/research/texturegen`.

References

CANTOR, G. 1878. Ein beitrag zur mannigfaltigkeitslehre. *Journal für die Reine und Angewandte Mathematik 84*, 242–258.

COHEN, M. F., SHADE, J., HILLER, S., AND DEUSSEN, O. 2003. Wang tiles for image and texture generation. *ACM Transactions on Computer Graphics 22*, 3, 287–294.

Figure 12: *Biq: A tool for procedural texture generation.*

DORAN, J., AND PARBERRY, I. 2010. Controlled procedural terrain generation using software agents. *IEEE Transactions on Computational Intelligence and AI in Games 2*, 2, 111–119.

LEHMER, D. 1954. Random number generation on the BRL high-speed computing machines. *Mathematical Reviews 15*, 559.

MUSGRAVE, F. K. 1999. Fractal solid textures: Some examples. In *Texturing & Modeling: A Procedural Approach*. AP Professional, Inc., ch. 11, 293–324.

MUSGRAVE, F. K. 1999. Procedural fractal terrains. In *Texturing & Modeling: A Procedural Approach*. AP Professional, Inc., ch. 12, 325–340.

PALLISTER, K. 2001. Generating procedural clouds using 3D hardware. In *Game Programming Gems 2*, M. DeLoura, Ed. Charles River Media, 463–473.

PARK, S. K., MILLER, K. W., AND STOCKMEYER, P. K. 1988. Technical Correspondence. *Communications of the ACM 36*, 7, 105–110.

PEACHEY, D. 1999. Building procedural textures. In *Texturing & Modeling: A Procedural Approach*. AP Professional, Inc., ch. 2, 7–96.

PERLIN, K. 1985. An image synthesizer. In *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, SIGGRAPH '85, 287–296.

RENKENS, W., AND KYOSEV, Y. 2010. Geometry modelling of warp knitted fabrics with 3D form. *Textile Research Journal 81*, 4, 437–443.

SZUDZIK, M. 2006. An elegant pairing function. In *The NKS2006 Wolfram Science Conference*.

WHITEHEAD, J. 2010. Towards proccedural[sic] decorative ornamentation in games. In *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*, 9:1–9:4.

WILLIAMS, C., 2015. Textiles and digital spaces. `https://xxxclairewilliamsxxx.wordpress.com/tools-to-create-and-explore-digital-patterns/`. (WordPress blog entry; accessed May 29, 2015).

WONG, M. T., ZONGKER, D. E., AND SALESIN, D. H. 1998. Computer-generated floral ornament. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, ACM, New York, NY, USA, SIGGRAPH '98, 423–434.

Appendix A Mathematical Proofs

Following are proofs of the identities used in Sections 2 and 4.1.

A.1 Identities from Section 2

The following result implies that textures constructed by using $h(x, y) \in [0, 1]$ as a grayscale value repeat every q texels along both the x and y axes, that is, they are composed of $q \times q$ tiles.

Theorem 1. *For all $x, y \in \mathbb{N}$,*

$$\pi(p(x+q), py) \equiv \pi(px, p(y+q)) \equiv \pi(px, py) \pmod{q}.$$

Proof. Suppose $x, y \in \mathbb{N}$.

$$\begin{aligned}
&\pi(p(x+q), py) \\
&\quad = ap^2(x+q)^2 + bp^2y^2 + cp^2(x+q)y + dp(x+q) + epy + f \\
&\quad \equiv ap^2x^2 + bp^2y^2 + cp^2xy + dpx + epy + f \pmod{q} \\
&\quad \equiv \pi(px, py) \pmod{q}.
\end{aligned}$$

The case of $\pi(px, p(y+q))$ is similar. □

The following is helpful when p and q share a common factor.

Lemma 2. *$p\hat{q} = q\hat{p}$.*

Proof.

$$p\hat{q} = p\frac{q}{\gcd(p,q)} = \frac{p}{\gcd(p,q)}q = \hat{p}q. \qquad \square$$

The following result strengthens Theorem 1 when p and q share a common factor. The textures actually have a tile size of $\hat{q} \times \hat{q}$.

Theorem 3. *For all $x, y \in \mathbb{N}$,*

$$\pi(p(x+\hat{q}), py) \equiv \pi(px, p(y+\hat{q})) \equiv \pi(px, py) \pmod{q}.$$

Proof. Suppose $x, y \in \mathbb{N}$.

$$\begin{aligned}
&\pi(p(x+\hat{q}), py) \\
&\quad = a(px+p\hat{q})^2 + bp^2y^2 + cp(px+p\hat{q})y + \\
&\qquad d(px+p\hat{q}) + epy + f \\
&\quad = a(px+\hat{p}q)^2 + bp^2y^2 + cp(px+\hat{p}q)y + \\
&\qquad d(px+\hat{p}q) + epy + f \quad \text{(by Lemma 2)} \\
&\quad \equiv ap^2x^2 + bp^2y^2 + cp^2xy + dpx + epy + f \pmod{q} \\
&\quad \equiv \pi(px, py) \pmod{q}.
\end{aligned}$$

The case of $\pi(px, p(y+\hat{q}))$ is similar. □

A.2 Identities from Section 4.1

The following result shows that when $\pi(x,y) = ax^2 + by^2 + cxy + f$, we can restrict p to $p < q/2$ without loss of generality because the texture with $p = r$ is identical to the texture with $p = q - r$.

Theorem 4. *If $d = e = 0$, then for all $x, y \in \mathbb{N}$,*

$$\pi((q-p)x, (q-p)y) \equiv \pi(px, py) \pmod{q}$$

Proof. Suppose $\pi(x,y) = ax^2 + by^2 + cxy + f$ and $x, y \in \mathbb{N}$.

$$\begin{aligned}
&\pi((q-p)x, (q-p)y) \\
&\quad = a(q-p)^2x^2 + b(q-p)^2y^2 + c(q-p)^2xy + f \\
&\quad \equiv a(-p)^2x^2 + b(-p)^2y^2 + c(-p)^2xy + f \pmod{q} \\
&\quad \equiv ap^2x^2 + bp^2y^2 + cp^2xy + f \pmod{q} \\
&\quad \equiv \pi(px, py) \pmod{q}.
\end{aligned}$$

□

The following result shows that the $\hat{q} \times \hat{q}$ tiles constructed with $\pi(x,y) = ax^2 + by^2 + ey + f$ are symmetric about the x axis and those constructed with $\pi(x,y) = ax^2 + by^2 + dx + f$ are symmetric about the y axis.

Theorem 5. *If $c = d = 0$, then for all $x, y \in \mathbb{N}$,*

$$\pi(p(\hat{q}-x), py) \equiv \pi(px, py) \pmod{q}.$$

If $c = e = 0$, then for all $x, y \in \mathbb{N}$,

$$\pi(px, p(\hat{q}-y)) \equiv \pi(px, py) \pmod{q}.$$

Proof. Suppose $\pi(x,y) = ax^2 + by^2 + ey + f$ and $x, y \in \mathbb{N}$.

$$\begin{aligned}
&\pi(p(\hat{q}-x), py) \\
&\quad = a(p\hat{q}-px)^2 + bp^2y^2 + epy + f \\
&\quad = a(\hat{p}q-px)^2 + bp^2y^2 + epy + f \quad \text{(by Lemma 2)} \\
&\quad \equiv ap^2x^2 + bp^2y^2 + epy + f \pmod{q} \\
&\quad \equiv \pi(px, py) \pmod{q}.
\end{aligned}$$

The case of $\pi(px, p(\hat{q}-y))$ is similar. □

The following result shows that any rectilinear ornamentation texture can, under certain conditions, have tile size smaller than $\hat{q} \times \hat{q}$.

Theorem 6. *If $c = d = f = 0$ and there exists $0 < r \leq q/2$ such that $r^2 \equiv 0 \pmod{q}$ and $r | 2p^2$, then for all $x, y \in \mathbb{N}$,*

$$\pi(p(x+r), py) \equiv \pi(px, p(r+y)) \equiv \pi(px, py) \pmod{q}.$$

Proof. Suppose $\pi(x,y) = ax^2 + by^2$ and $x, y \in \mathbb{N}$. Further, suppose there exists $0 < r \leq q/2$ such that $r^2 \equiv 0 \pmod{q}$ and $r | 2p^2$. Then

$$\begin{aligned}
\pi(p(x+r), py) &= ap^2x^2 + a(2p^2)rx + ap^2r^2 + bp^2y^2 \\
&\equiv ap^2x^2 + bp^2y^2 \pmod{q}.
\end{aligned}$$

The case of $\pi(px, p(r+y))$ is similar. □

The following result shows that the $\hat{q} \times \hat{q}$ tiles constructed with $\pi(x,y) = ax^2 + by^2 + cxy + f$ are symmetric about the diagonal.

Theorem 7. *If $d = e = 0$,*

$$\pi(p(\hat{q}-x), p(\hat{q}-y)) \equiv \pi(px, py) \pmod{q}.$$

Proof. Suppose $\pi(x,y) = ax^2 + by^2 + cxy + f$ and $x, y \in \mathbb{N}$.

$$\begin{aligned}
&\pi(p(\hat{q}-x), p(\hat{q}-y)) \\
&\quad = a(p\hat{q}-px)^2 + b(p\hat{q}-py)^2 + c(p\hat{q}-px)(p\hat{q}-py) + f \\
&\quad = a(\hat{p}q-px)^2 + b(\hat{p}q-py)^2 + c(\hat{p}q-px)(\hat{p}q-py) + f \\
&\qquad \text{(by Lemma 2)} \\
&\quad \equiv ap^2x^2 + bp^2y^2 + cp^2xy + f \pmod{q} \\
&\quad \equiv \pi(px, py) \pmod{q}.
\end{aligned}$$

□