Crooked Wallpaper

Robert J. MacG. Dawson*
Dept. of Mathematics and Computing Science
St. Mary's University
Halifax, Nova Scotia, Canada B3H 3C3

September 17, 2001

Abstract

A well-known tiling of the plane with rectangles is used as the basis of a protocol to store and transmit periodic bitmaps whose period vectors are not aligned with the screen raster axes. Some related results are presented, including an algorithm to find all choices for the dimensions of the rectangles.

Keywords: periodic tiling, rectangle, wallpaper, bitmap, parallelogram, Euclid's algorithm

AMS subject classification: 52C05

*Supported by a grant from NSERC

1 Introduction

Periodic designs have been used for decorative purposes in many cultures, over thousands of years. Such designs were used for fabric and architectural embellishment, and raised to an art form by such proponents as William Morris and M. C. Escher (among many others). Within the last twenty years, however, the rise of graphical user interfaces on computers – and the advent of widespread surplus processing power – has given rise to several related applications.

Early drawing programs – often with palettes of only 2 or 16 colors – used bitmapped textures, both to facilitate fill effects such as bricks and tiles and to achieve dithered gray regions. These textures consisted typically of 8×8 two-color bitmaps. They did not lend themselves to a great deal of variety, and were somewhat overused by amateur graphic artists; as a result these fills have rather a "period" flavor today. Modern bitmap "painting" programs use larger tiled fills, as well as gradient and algorithmic fills that are created in real time.

The same small and simple bitmaps were used as "wallpaper" by early graphic user interfaces. Given the combination of low video resolution and expensive memory (by the standards of a decade later), they were an appropriate way to individualize desktops. However, while the "pattern" background is still implemented in modern desktops, very few users at the time of writing still use it. Not only have monitor resolutions risen to the point where most 8×8 patterns cannot really be distinguished, but memory (of various types) has become plentiful enough that larger images, with 256 or more colors, can be used as backgrounds.

In fact, the importance of tilable images for wallpaper has diminished; both disc space and RAM have become plentiful enough that the storage of (say) a 1600×1200 pixel bitmap, covering the whole desktop, does not take an unreasonable proportion of system resources. However, another important application for tiled images has emerged within the last decade – the background images for web pages. These are subject to constraints which go beyond those constraining fill textures and wallpaper. For instance, the creator of a web page does not usually know at what size or resolution it will be viewed; so tilable images have a distinct advantage, in that they can fill whatever space needs to be filled. Even more importantly, many users still access the Web over comparatively slow telephone lines; as long as this remains the case, there is a powerful argument for keeping the bitmap that is tiled to create the background image small.

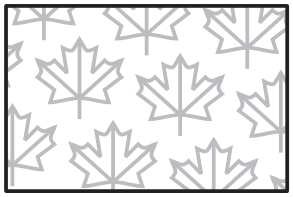

Figure 1: Part of an obliquely-tiled periodic image

Given the constant search for visual novelty in the design of Web pages, it would seem that an extension to HTML that enabled the transmission of obliquely-tiled images (as in Figure 1) would be popular with users. There are many ways in which this could be done; however, some of them are inefficient in that they require redundant data to be transmitted by the server. Others allow only a subset of translational tilings, or require major innovations rather

than making use of existing file formats. We will examine some of these, before looking in detail at a possible protocol for transmission of obliquely-tiled images that has none of these problems.

2 Definitions and basics

Mathematically, we can consider an image in the plane to be a function $I : \mathbb{R}^2 \to P$ where P is the *palette* or set of distinguishable colors; a bitmap is a function $I_Z : \mathbb{Z}^2 \to P$. An image in the plane is periodic if there exist one or more vectors $\vec{q}_i \in \mathbb{R}^2$ (resp. $\mathbb{Z}^2$) such that $I(\vec{v} + \vec{q}_i) = I(\vec{v})$ for all $\vec{v} \in \mathbb{R}^2$ (resp. $\mathbb{Z}^2$). We call the $\{\vec{q}_i\}$ *period vectors*. It is easily seen that for any I, the period vectors form a *lattice* $\mathcal{P}$ (that is, a discrete set of points forming a group under vector addition). The quotient $\mathbb{R}^2/\mathcal{P}$ (resp. $\mathbb{Z}^2/\mathcal{P}$) is called the *prototile*.

It is important to note that $\mathcal{P}$ always has infinitely many bases. To be precise, if $(\vec{p}_1, \vec{p}_2)$ is a basis, so is $(a\,\vec{p}_1 + b\,\vec{p}_2, c\,\vec{p}_1 + d\,\vec{p}_2,)$ where $ad - bc = \pm 1$. It follows that the set of bases for $\mathcal{P}$ is indexed by the *special linear group* $SL(2, \mathbb{Z})$ (see, *e.g.*, [5], page 157, for a discussion of this group.)

Each basis gives rise "naturally" to a parallellogram prototile; however, it should be stressed that the property of being a prototile depends on $\mathcal{P}$, not on the choice of basis. Moreover, the shape of a prototile is not fully determined by $\mathcal{P}$ (see Figure 2, or various of M. C. Escher's works [4], for examples.)

It is easily seen (and well known) that, for any periodic bitmap, the lattice $\mathcal{P}$ has dimension at most 2 (that is, it is generated by some two of its elements.). The same is true for any continuous image whose prototile has nonempty in-

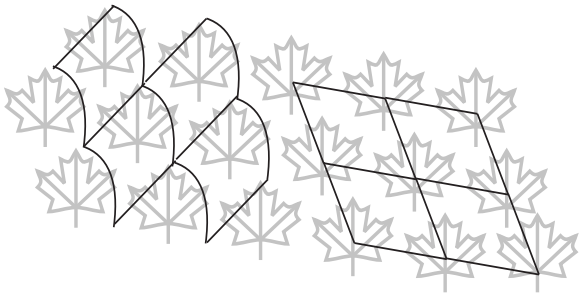

Figure 2: Two prototiles with the same period lattice

terior. If $\dim(\mathcal{P}) = 0$ or 1, the prototile is infinite in extent. Thus, any finite prototile that tiles the (real or integer) plane in a graphically useful fashion has $\dim(\mathcal{P}) = 2$.

It may be observed that most computer applications that use tiled images assume a rectangular prototile, with horizontal and vertical edges $\vec{x}$ and $\vec{y}$ respectively; and the period lattice $\mathcal{P}$ is generated by $\vec{x}$ and $\vec{y}$. That is, they use *edge-to-edge* tilings with rectangular prototiles (Figure 3). In particular, HTML (at the time of writing) is restricted to such a tile (see, for instance, [2]).

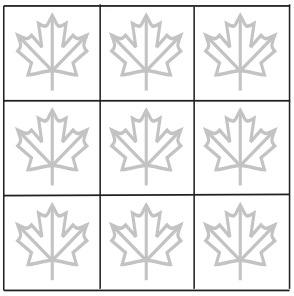

Figure 3: A basic rectangular tiling

3 Rectangular sublattices

In some cases (see Figure 4), an oblique period lattice may have a rectangular sublattice. At the moment, this is the method generally used to create background images with an apparent oblique periodicity. The disadvantage is obvious; the size of the bitmap to be transmitted will be at least twice that of the period module, and possibly much larger.

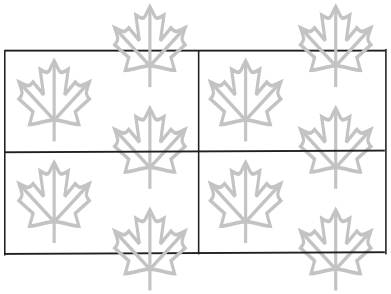

Figure 4: A rectangular sublattice

4 Nonrectangular tiles

It would be possible in principle to transmit a parallellogram tile (Figure 5a). This has the disadvantage that much existing code for decoding and blitting rectangular images could not be reused. It might not be too difficult to generalize existing algorithms and formats for raw bitmap files, or run-length-encoding; however, some important formats, such as JPEG, would seem harder to adapt.

Again, it would be possible (Figure 5b) to transmit a partially transparent image, in which the region outside the nonrectangular prototile is transparent. This could be done with GIF images but would not adapt easily to raw RGB

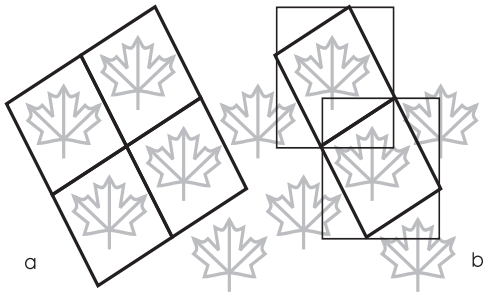

Figure 5: A nonrectangular prototile

bitmaps, or JPEG-compressed images. In these cases the viewing software could compute a clipping mask from the transmitted period data and use it in the blitting process; but this makes the transmission inefficient, as a (potentially large) portion of the transmitted image is thrown away. If a run-length encoded image format is used, the compression of the "invisible" part of the image may be facilitated by using a single flat color on the part to be discarded; however, these file formats are not appropriate for all images.

The same effect can be obtained by "shingling" rectangular images so that each one is partially covered by one or more neighbours. This is fairly efficient for blitting (perhaps less so when the region need only be partially redrawn), but again requires the transmission of a significant region of unused image.

5 Offset rectangular tiles

In some cases, we could simply send a rectangular bitmap and an offset distance y_0 (Figure 6).

It seems obvious that not every doubly periodic image can be transmitted in this way; and this is indeed the case. However, there is more here than meets

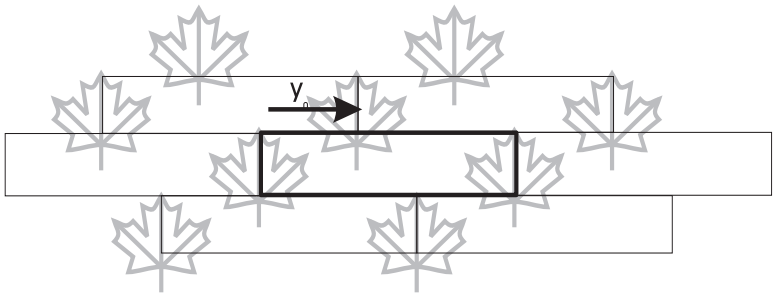

Figure 6: A method that sometimes works

the eye; every doubly periodic *bitmap* does have a rectangular prototile, and can be tiled as in Figure 6.

Suppose that $\mathcal{P}$ has a basis $\{(x_1, y_1), (x_2, y_2)\}$. If one of the coordinates – say y_1 – is zero, then a rectangle of width x_1 and height y_2 forms a prototile as in Figure 6 (without loss of generality, suppose that $x_1, x_2 > 0$).

If, on the other hand, all four coordinates are nonzero, we may assume without loss of generality that $x_1, x_2 > 0$. Consider the following procedure, which acts on a basis:

```
procedure squash
begin
   if x1 ≥ x2 then
    begin
      y1 := y1 − y2;
      x1 := x1 − x2;
    end
   else
    begin
      y2 := y2 − y1;
      x2 := x2 − x1;
    end
end
```

Listing 1: The procedure **squash**

This restricts to Euclid's algorithm (see Euclid, [1], VII.2, or the treatment in Knuth, [3], I.1) on the x coordinates; and if $\{\vec{p_1}, \vec{p_2}\}$ is a basis for $\mathcal{P}$, so is $\mathbf{squash}(\vec{p_1}, \vec{p_2})$. (To see this, note that

$$\mathbf{squash}\begin{pmatrix}\vec{p_1'}\\ \vec{p_2'}\end{pmatrix} = \begin{bmatrix}1 & -1\\ 0 & 1\end{bmatrix}\begin{pmatrix}\vec{p_1}\\ \vec{p_2}\end{pmatrix} \text{ or } \begin{bmatrix}1 & 0\\ -1 & 1\end{bmatrix}\begin{pmatrix}\vec{p_1}\\ \vec{p_2}\end{pmatrix}$$

and both matrices are invertible over $\mathbb{Z}$.) It thus follows that for some n,

$$\mathbf{squash}^n((x_1, y_1), (x_2, y_2)) = \{(0, y_1'), (\gcd(x_1, x_2), y_2')\} \tag{1}$$

and this basis, with a zero coordinate, gives the desired tiling.(Of course, this relies on the discrete nature of a bitmapped image; a continuous periodic image with periods (x_1, y_1) and (x_2, y_2) has a rectangular prototile if and only if x_1/x_2 or y_1/y_2 is rational.)

If transmission times are the only criterion, this method is optimally efficient; the transmitted image is a single prototile. However, the width $\gcd(x_1, x_2)$ may be very large compared with the width of the area to be tiled. In this case, a large number of partial blits will take place (Figure 7), which may reduce the efficiency of the tiling significantly.

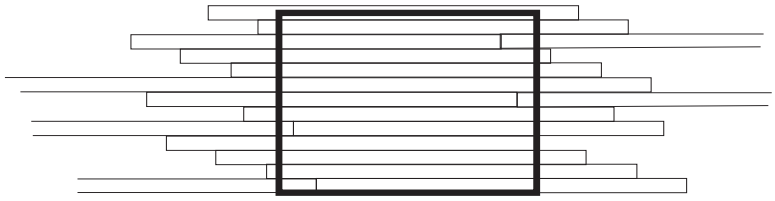

Figure 7: An inefficient method of tiling

6 Two rectangular tiles

It has been known for a long time that the plane may be tiled with translates of two arbitrary rectangles (the origins of this tiling are undoubtedly lost in antiquity). Specifically, the union of two rectangles whose edge lengths are (a_1, b_1) and (a_2, b_2), positioned as in Figure 8, will tile the plane with periods (a_1, b_2) and $(a_2, -b_1)$.

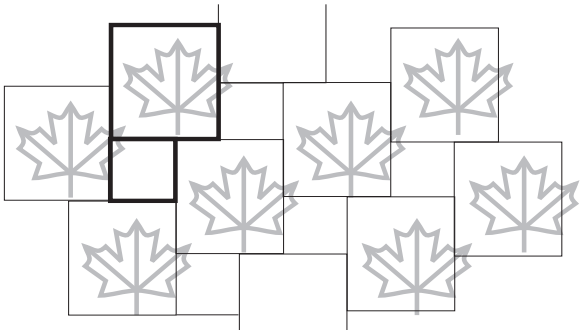

Figure 8: A tiling with two rectangles

As these images are rectangular, with edges in the standard directions, any standard bitmapped image format can be used to store the two parts of the prototile. Moreover, by recovering the width and height of each image, the application displaying the periodic image can determine two periods that form a basis for $\mathcal{P}$. This would make it very easy to extend (for instance) HTML to support such a protocol.

It should be noted that, in general, the dimensions of the two rectangles are not unique. As observed above, there are always infinitely many choices for the generators of $\mathcal{P}$, related by the elements of $SL(2, \mathbb{Z})$, and if the number of negative coordinates among $\{x_1, y_1, x_2, y_2\}$ is odd (without loss of generality, 1)

we may construct such a pair of rectangles.

If one coordinate is 0, then the corresponding rectangle is degenerate, and we get tilings of the form considered in the previous section. If an even number of coordinates are strictly negative, we obtain a generalized tiling in which one rectangle appears in a negative sense, and area of the other is larger than that of the prototile. Such a generalized tiling is, while mathematically interesting, not useful for our purposes. It is thus of interest to determine, for any period lattice $\mathcal{P}$, the set of bases with exactly three positive coordinates.

If we repeat the operation **squash** one time fewer than in (1), we obtain a basis

$$\mathbf{squash}^{n-1}((x_1, y_1), (x_2, y_2)) = \{(\gcd(x_1, x_2), y_1''), (\gcd(x_1, x_2), y_2'')\}$$

and, as $\begin{bmatrix} m & 1-m \\ m+1 & -m \end{bmatrix} \in SL(2, \mathbb{Z})$, it follows that

$$\{(\gcd(x_1, x_2), my_1'' + (1-m)y_2''), (\gcd(x_1, x_2), (m+1)y_1'' - my_2'')\} \qquad (2)$$

is also a basis for any integer m. In particular, if $m = \lceil y_1''/(y_1'' - y_2'') \rceil - 1$, we obtain a basis in which one of y_1, y_2 is positive and the other is negative.

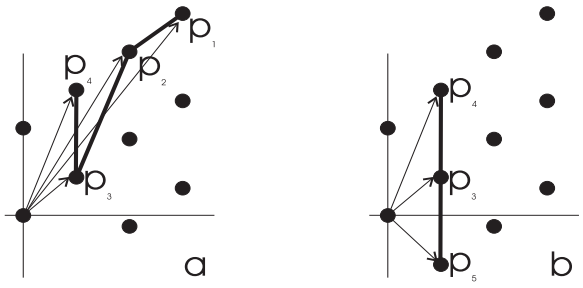

Figure 9: Change of basis

Figure 9 illustrates this. (Each line segment represents a basis, whose vectors

are the endpoints.) We start with the basis $(\vec{p_1}, \vec{p_2})$ and apply **squash**, obtaining $(\vec{p_2}, \vec{p_3})$ and then $(\vec{p_3}, \vec{p_4})$. Both the vectors of this basis have the minimal positive x coordinate, $x = \gcd(x_1, x_2)$. Any two adjacent lattice points on the vertical line $x = \gcd(x_1, x_2)$ form a basis; exactly one of these, here $(\vec{p_3}, \vec{p_5})$ corresponds to a segment that intersects the x axis.

We can then (as will be seen below) obtain every other such basis by repeated application of **stretch**, defined as follows:

```
procedure stretch
begin
  if y1 ≥ −y2 then
   begin
     y1 := y1 + y2;
     x1 := x1 + x2;
   end
  else
   begin
     y2 := y2 + y1;
     x2 := x2 + x1;
   end
end
```

Listing 2: The procedure **stretch**

Figure 10 shows the effect of repeating this operation, starting with the basis $(\vec{p_3}, \vec{p_5})$ reached above. Adding $\vec{p_3}$ to $\vec{p_5}$ yields $\vec{p_6}$; and the basis $(\vec{p_3}, \vec{p_6})$ has $x_3, x_6, y_3 > 0 > y_6$. Repeating **stretch** three more times, we eventually obtain the basis $(\vec{p_6}, \vec{p_9})$ which has one coordinate equal to 0.

The following lemma gathers some trivial observations about the **squash** and **stretch** operations.

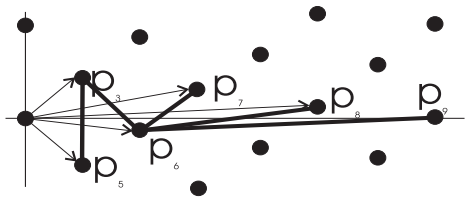

Figure 10: The **stretch** operation

Lemma 1 *Let* **squash**$((x_1, y_1), (x_2, y_2)) = ((x'_1, y'_1), (x'_2, y'_2))$, **stretch**$((x_1, y_1), (x_2, y_2)) = ((x''_1, y''_1), (x''_2, y''_2))$, *and* $x_1, x_2, y_1 > 0 > y_2$. *Then* $x'_2, x''_1, x''_2, y'_1 > 0 > y'_2, y''_2$; $x'_1 \geq 0$ *with equality iff* $x_1 = x_2$; *and* $y''_1 \geq 0$ *with equality iff* $y_1 = -y_2$. □

While **squash** and **stretch** are not generally inverses, they are inverses for the bases in which we are most interested.

Lemma 2 *If* $\vec{p_1} = (x_1, y_1)$, $\vec{p_2} = (x_2, y_2)$, *and* $x_1, x_2, y_1 > 0 > y_2$, *then*

$$\textbf{stretch squash}(\vec{p_1}, \vec{p_2}) = \textbf{squash stretch}(\vec{p_1}, \vec{p_2}) = (\vec{p_1}, \vec{p_2}) \ .$$

Proof: Let $(\vec{p'_1}, \vec{p'_2}) = \textbf{squash}(\vec{p_1}, \vec{p_2})$, and $(\vec{p''_1}, \vec{p''_2}) = \textbf{stretch squash}(\vec{p_1}, \vec{p_2})$. If $x_1 \geq x_2$, then $x'_1 = x_1 - x_2$, $x'_2 = x_2$, $y'_1 = y_1 - y_2$, and $y'_2 = y_2$. Then $y'_1 > -y'_2$, so $x''_1 = x'_1 + x'_2 = x_1$, $x''_2 = x'_2 = x_2$, $y''_1 = y'_1 + y'_2 = y_1$, and $y''_2 = y_2$.

The case in which $x_1 < x_2$, and both cases for **squash stretch**$(\vec{p_1}, \vec{p_2})$, proceed in the same way. □

With these lemmas, we can show that the following algorithm, given a basis $((x_1, y_1), (x_2, y_2))$ for a period lattice $\mathcal{P}$, generates all the bases for $\mathcal{P}$ that correspond to pairs of nondegenerate rectangles.

```
procedure list_bases (int x_1, y_1, x_2, y_2)
begin
    while (x_1 <> x_2) squash;
    dy := y_1 - y_2;
    y_1 := y_1 mod dy;
    y_2 := y_1 - dy;
    while (y_1 > 0)
    begin
       output(x_1, y_1, x_2, y_2);
       stretch;
    end
end
```

Listing 3: The procedure **list_bases**

Proposition 1 *The algorithm* **list_bases** *outputs exactly the bases* $\{(x_1, y_1), (x_2, y_2)\}$ *for* $\mathcal{P}$ *that have* $x_1, x_2, y_1 > 0 > y_2$

Proof: As **squash**, restricted to the x coordinates, is a step of Euclid's algorithm, the first **while** loop halts after producing a basis $\{(x_1', y_1'), (x_2', y_2')\}$ with $x_1' = x_2' = \gcd(x_1, x_2)$. The next three lines convert this to a basis $\{(x_1'', y_1''), (x_2'', y_2'')\}$ which has $x_1'' = x_2'' = \gcd(x_1, x_2)$ and also $y_1'' \geq 0 > y_2''$.

If $y_1'' = 0$, $\mathcal{P}$ is generated by $(0, y_1'')$ and $(x_1'', 0)$; in this case there are no proper two-rectangle period modules, and the program halts without output. Otherwise $y_1'' > 0$, and the second **while** loop outputs one or more bases. As **stretch** restricts to Euclid's algorithm on $(y_1, -y_2)$, this loop must also halt eventually. An induction using Lemma 1 guarantees that every basis produced by the second **while** loop has $x_1, x_2, y_1 > 0 > y_2$ except for the last one, which is never output. It remains to show that all such bases are found. Given any such basis B, if we **squash** until $x_1' = x_2' = \gcd(x_1, x_2)$, we must already have $y_1' > 0 > y_2'$; so $\mathbf{squash}^n(B) = \hat{B}$. But then, by Lemma 2, $\mathbf{stretch}^n(\hat{B}) = B$. □

We note in passing that Proposition 1 implies that there are only finitely many pairs of rectangles that form prototiles for any $\mathcal{P}$. It may also be observed that **list_bases** produces the bases in increasing order of $x_1 + x_2$ and decreasing order of $y_1 - y_2$. This may be useful in selecting a basis to use.

As observed in Section 5, inefficiencies in blitting result when the total area of tiles that extends beyond the area to be painted is large. The basis choice that minimizes this waste depends upon the shape of the region to be tiled, which is typically not known when the bitmaps are generated. It may be shown (details are omitted) that when there is a tiling with rectangles whose proortions are the same as that of the (rectangular) tiled region, this choice of rectangles is optimal. As we do not need an exactly optimal solution, and usually do not know the exact shape of the region to be tiled anyway, it is appropriate to use some heuristic such as choosing a basis for which $|(x_1+x_2)-(y_1-y_2)|$ is minimal (that is, for which the sum of the widths of the rectangles is as near as possible to the sum of their heights.) As **list_bases** produces the bases in increasing order of $x_1 + x_2 - y_1 + y_2$, this search can be done efficiently.

7 Conclusions

We have examined various ways of transmitting, and painting a doubly-periodic bitmapped pattern whose period vectors are not parallel to the raster axes. One of these, using two rectangles, is efficient both to transmit and to paint. The different ways of decomposing the period module are seen to be closely related to the bases of the period lattice. An algorithm was given by which those bases which yield usable decompositions can be listed in a natural order.

References

[1] Euclid, *The Elements* (T. Heath, ed.), Dover, 1956

[2] M. Holzschlag, *Special Edition Using HTML 4*, 5th edition, Que, 1999

[3] D. E. Knuth, *The Art of Computer Programming*, Addison-Wesley, 1997

[4] J. L. Locher (ed.), *Escher: The Complete Graphic Work*, Thames and Hudson, 1982

[5] J. J. Rotman, *The Theory of Groups: An Introduction*, Allyn and Bacon, 1973