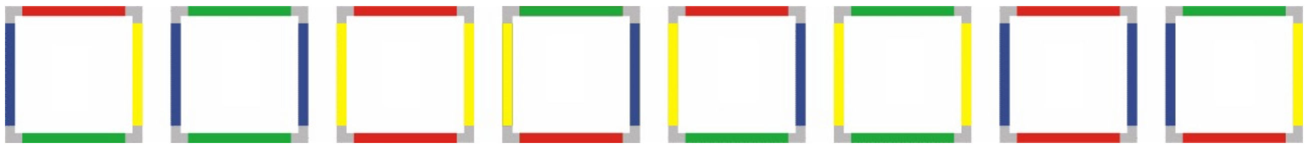
Also publ. as Paper in: International Conference on Computer Graphics and Interactive Techniques (SIGGRAPH), San Diego, Calif., 2003

Wang Tiles for Image and Texture Generation

Michael F. Cohen[1] Jonathan Shade[2,3] Stefan Hiller[4] Oliver Deussen[4]

[1]Microsoft Research [2]WildTangent [3]University of Washington [4]Dresden University of Technology



Abstract

We present a simple stochastic system for non-periodically tiling the plane with a small set of Wang Tiles. The tiles may be filled with texture, patterns, or geometry that when assembled create a continuous representation. The primary advantage of using Wang Tiles is that once the tiles are filled, large expanses of non-periodic texture (or patterns or geometry) can be created as needed very efficiently at runtime.

Wang Tiles are squares in which each edge is assigned a color. A valid tiling requires all shared edges between tiles to have matching colors. We present a new stochastic algorithm to non-periodically tile the plane with a small set of Wang Tiles at runtime.

Furthermore, we present new methods to fill the tiles with 2D texture, 2D Poisson distributions, or 3D geometry to efficiently create at runtime as much non-periodic texture (or distributions, or geometry) as needed. We leverage previous texture synthesis work and adapt it to fill Wang Tiles. We demonstrate how to fill individual tiles with Poisson distributions that maintain their statistical properties when combined. These are used to generate a large arrangement of plants or other objects on a terrain. We show how such environments can be rendered efficiently by pre-lighting the individual Wang Tiles containing the geometry.

We also extend the definition of Wang Tiles to include a coding of the tile corners to allow discrete objects to overlap more than one edge. The larger set of tiles provides increased degrees of freedom.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation - Viewing Algorithms— [I.3.6]: Computer Graphics—Methodology and Techniques

Keywords: non-periodic tiling, Poisson distributions, texture synthesis, Wang Tiles

1 Introduction

Modeling and rendering scenes that capture the complexity of the real world is a difficult and time consuming task. A now well-known means to overcome this problem is to create (or capture) a small example of complexity and then reuse this example many times. Unfortunately, when the same example is used many times in a periodic fashion, the repetition is often apparent and distracting.

We present a new stochastic algorithm to non-periodically tile the plane with a small set of Wang Tiles [Wang 1961; Wang 1965]. This allows Wang Tiles to share the efficiency of reusing example tiles to create large expanses of complex texture, patterns, or pre-lighted geometry at runtime, while avoiding the obvious visual artifacts of periodicity.

Wang Tiles are a set of squares in which each edge of each tile is *colored*. Matching colored edges are aligned to tile the plane. We demonstrate that as few as eight tiles are needed to non-periodically cover the plane. We present new methods to fill the tiles with 2D texture, 2D Poisson distributions, or 3D geometry to efficiently create at runtime as much non-periodic texture (or distributions, or geometry) as needed. We apply these results to modeling and rendering problems.

Wang Tiles only define matching edge constraints. This is insufficient to maintain coherent features that cross more than one edge (i.e., corners). To accommodate such features, we extend the notion of Wang Tiles to include corner constraints in the spirit of [Neyret and Cani 1999]. The increase in degrees of freedom also allows us to modulate the texture by mixing two source textures.

1.1 Relation to Previous Work

There are a variety of methods where small samples are used to generate more complex patterns. Much of the research involved in these methods strive to avoid visual artifacts arising from repeatedly using the same sample data. Related work can be divided into two groups: 2D- and 3D-texturing methods, and geometry creation for complex outdoor scenes and related level-of-detail methods. We also touch on the relation of Wang Tiles to the broader literature on tiling.

Texture Synthesis: Stam was the first to consider the use of square Wang Tiles for texture synthesis [Stam 1997]. Based on a deterministic algorithm that uses a set of 16 tiles with colored edges [Grünbaum and Shephard 1987] he created large non-repetitive textures, water surfaces and caustics by defining texture samples on the tiles following the border constraints introduced by the shared edges. In his paper, he describes the construction of a limited set of patterns. A general algorithm for filling the tiles and quickly assembling a tiling is not given. We extend his work in these directions.

Recent work in texture synthesis provides a means to avoid repetition in two dimensions by using a small texture tile as an example that is then used to create a larger amount of non-repetitive texture that has the same visual properties.

Wei and Levoy [Wei and Levoy 2000] use a statistical model that selects a pixel of the given texture by analyzing the local neighbor-

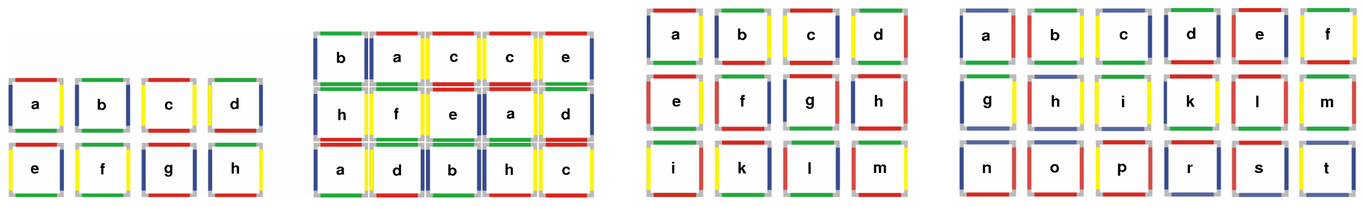


Figure 1: a) Eight Wang Tiles that can stochastically tile the plane; b) A small portion of the plane with a valid tiling. c) 12 Wang Tiles constructed from two horizontal and three vertical colors; d) 18 Wang Tiles constructed from three horizontal and vertical colors.

hood of the pixel in the so far synthesized texture. Efros and Freeman [Efros and Freeman 2001] randomly select square regions from the sample image and fit (or *quilt* those parts to the already created large texture. One of the challenges in this work is to maintain the visual appearance of large coherent regions. The work cited above achieves these goals at a computational and storage cost. We leverage these results to fill a small set of Wang Tiles, such that the resultant tiling maintains the ability to include large coherent features. The tiling itself mitigates the computational and memory requirements when creating large textures.

Several algorithms have been proposed to extend the methods to arbitrary surfaces [Turk 2001; Wei and Levoy 2000; Tong et al. 2002], but these create large textures to cover the entire geometry. The work presented here is most closely related to the non-periodic texture mapping method based on triangles presented by Neyret et al [Neyret and Cani 1999]. Using this kind of mapping, they are able to texture a variety of objects. Because the triangles are rotated to achieve a mapping, triangles are not well suited for textures with directional content such as strokes or oriented patterns. Wang Tiles avoid these problems since they are square and cannot be rotated, but the tilings are restricted to surfaces that map to a plane. The triangle rotation also requires the edges to have an orientation effectively doubling the number of colors. Beyond these differences, we focus in this paper primarily on new methods to construct the tile interiors themselves.

Modeling of complex ecosystems: A complex data generation problem arises when modeling outdoor scenes. Deussen et al. [Deussen et al. 1998] simulate such scenes by covering the plane with tens of thousands of synthetic plants. Approximate instancing helps them to reduce the geometric complexity but the storage of the instancing information is sometimes a challenge, as stated in [Deussen et al. 2002]. Wang Tiles can store plants or plant instances on a small set of tiles while constructing large distributions through tiling the plane. Lighting calculations can also be leveraged by performing them only once per instance. For example, Layered Depth Images [Gortler et al. 1996; Levoy and Hanrahan 1996; Shade et al. 1998] showed how a point sampled representation of such geometry can be pre-lighted and rendered from changing viewpoints at interactive speeds. We show how the same methods can be applied to Wang Tiles.

Tiling: There is also a vast literature on tiling methods [Grünbaum and Shephard 1987; Glassner 1998], however, the square nature of Wang Tiles avoids most of the issues related to assembling a tiling, thus this rich literature is mostly outside the scope of this work. Expanses of texture from a single square tile can be generated easily by simply mirroring the tile, a process easily supported by texture mapping hardware. Wang Tiles can be thought of as a way to overcome the repetitive nature of such a simple scheme at only a modest cost in memory, while maintaining the interactive speeds of hardware supported texture mapping.

2 Wang Tiles

A Wang Tile set consists of square tiles with color-coded edges. The squares cannot be rotated. A valid tiling of the infinite plane consists of any number of copies from the set laid down such that all contiguous edges have matching colors. This tiling is named after Hao Wang who conjectured in 1961 that any set of tiles that can produce a valid tiling of the plane must also be able to produce a periodic tiling of the plane [Grünbaum and Shephard 1987]. This was later refuted in a series of papers that eventually led to a set of only 13 tiles discovered by Culik that could be shown to be strictly aperiodic[1] [Berger 1966; Culik II 1996; Kari 1996]. Theoreticians have taken an interest in aperiodic sets, using them to simulate the properties of proposed DNA computers, and have shown they can simulate any Turing machine. They have even been featured in a leading role in a science fiction story, "Wang's Carpets" by Greg Egan. We will not draw on these more theoretical aspects of Wang Tiles, but rather on their simplicity; specifically, that they are square and that a small set can lead to a non-periodic tiling of the plane.

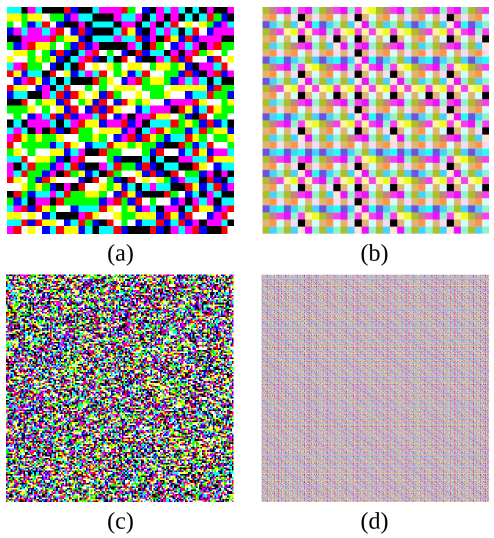


Figure 2: Assemblies of tiles comparing stochastic tilings with strictly aperiodic sets. Each small pseudo-colored square pixel represents one whole tile: (a) 32×32 tiles stochastically assembled (b) 32×32 strictly aperiodic set [Grünbaum and Shephard 1987] (c) 256×256 stochastic (d) 256×256 aperiodic.

We introduce a new stochastic process to laying down individual tiles. This leads to the ability to generate large tilings at runtime that are guaranteed to be non-periodic.

[1]We use the term non-periodic to indicate a tiling which is not periodic (although the set itself could lead to a periodic tiling), as opposed to the term aperiodic that refers to a set of tiles that can never produce a periodic tiling.

A stochastic tiling algorithm: We work with sets that can be trivially shown to be able to tile the plane and also trivially shown not to be aperiodic. In fact, a single tile, such as tile b in Figure 1(a) can tile the plane on its own. When tiling the plane, each choice of tile typically has two constraints. If one is placing tiles from West to East and from North to South, then each tile placed must have N and W edges that match the E and S edges already placed. If there are K colors then there are K^2 combinations of colors for two adjacent edges of the tiles (e.g., the south (S) and east (E) edges)[2]. So long as there is at least one tile in the set with each north (N) and west (W) combination, then the following simple procedure produces a valid tiling of any portion of the plane:

1. select any tile for the NW corner;
2. tile the top row left to right by choosing tiles for which the W edge of the new tile matches the E edge of the previous tile;
3. select a first tile for the next row such that the N edge matches the S edge from above;
4. continue this row by randomly selecting tiles for which the N and W edges match the S and E edges from above and the left respectively (since our set contains all NW combinations, this is always possible); and
5. go to step 3 for as many rows as desired.

If the set contains at least two tiles for each NW combination (i.e., $2 \times K^2$), then there are always at least two choices at each step. If this choice is made with uniform probability, the plane will always be tiled non-periodically because each step reduces to an independent random process (i.e., a coin flip). One such set of 8 tiles with 2 colors is shown in Figure 1(a). Note that there are, in practical terms, only two colors since the red and green used for the NS edges could be replaced with the blue and yellow of the EW edges with no change in the set.

In Figure 1(a) tiles a through d contain all NW combinations as do tiles e through h. The SE edge combinations are also each represented twice thus tilings can proceed from south to north as well. Many other sets are possible with all required properties. A valid tiling of a small portion of the plane may look like Figure 1(b).

This simple algorithm results in tilings that have a random appearance as can be seen in Figure 2(a)-(d). In Figure 2(a) a tiling constructed from the tile set of Figure 1(a) is shown. Each tile was given its own color to make them distinguishable from one another. In Figure 2(b) a deterministic aperiodic tiling [Grünbaum and Shephard 1987] of 16 tiles can be seen to still contain some repetitive appearance. Figures 2(c) and (d) show larger parts of stochastic and deterministic tiling.

Often larger sets of tiles are needed to fully avoid artifacts. Using three horizontal colors results in 12 tiles (Figure 1(c)) and using three horizontal and three vertical colors results in 18 tiles (Figure 1(d)). The main advantage of a larger set is that it can substantially reduce repetition artifacts as in Figure 6(d). Another advantage is that using such a set we are able to introduce larger patterns to the tiling of the plane. We will further address these issues in Section 3.5 by introducing inhomogeneity into the tiling patterns. But first we continue by describing our methods for creating valid tile sets for two dimensional applications such as textured surfaces or non-repetitive drawings.

[2]The orientable triangles in Neyrets work [Neyret and Cani 1999] would require $(2K)^2/3$, K is effectively doubled by the orientation, but reduced by a factor of 3 since each triangle has 3 edge pair possibilities.

3 Tile Design

Wang Tiles provide the framework for constructing large expanses of texture from a small amount of input. We will first demonstrate and discuss an interactive tile design system. We follow this with an automated system that takes discrete texture primitives, such as images of leaves, as input. We introduce an optimization method that assures that discrete primitives crossing tile boundaries are kept intact while minimizing artifacts within the tile interiors.

3.1 Interactive Tile Design

The simplest way to design a set of Wang Tiles is to create them interactively. A simple tool allows us to place geometric primitives within the set. The primitives can be imported from any drawing tool. If a primitive is moved such that it overlaps an edge, it will automatically appear in all tiles with the same and opposing edges. Thus, primitives can reside partially in one tile and partially in tiles with the opposite edge of the same color. A perspective projection of a portion of the tiled plane provides immediate feedback of the overall appearance and large scale patterns that might occur. Figure 3 shows two manually created sets from line patterns and the resulting tilings with several hundred tiles. Although the texture was created from only eight Wang Tiles, almost no repetition artifacts can be seen.

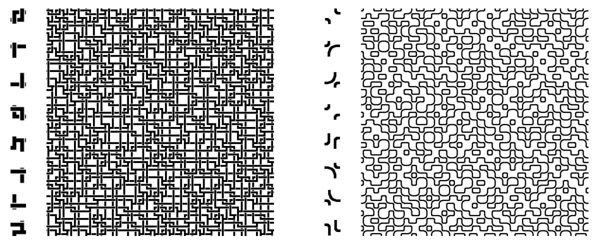


Figure 3: Two interactively designed eight tile sets to the left of resulting tilings.

3.2 Automatic Tile Design for Texture Synthesis

To create a continuous texture from a sample, textures must be found for each tile that fit together across the boundaries with matching colors. In our variant of the algorithm we generate such textures. A tile is created by combining diamond shaped (squares rotated by 45 degrees) sample portions of the source image, one for each edge color of horizontal and vertical edges. Therefore, for a set of eight tiles we need four sample images, and for the set of 18 tiles six sample images.

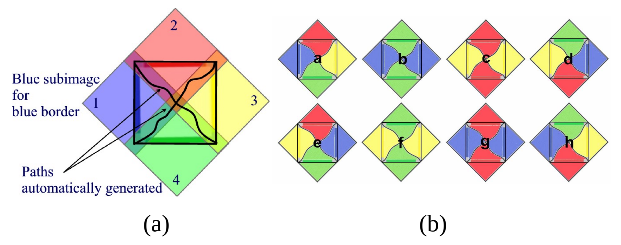


Figure 4: a) Four subimages are combined to form each Wang Tile; b) construction of an eight tile set.

The set of Wang Tiles can also be generated automatically. Our method is derived from work by Efros and Freeman [Efros and Freeman 2001]. In their method a large texture is created from small

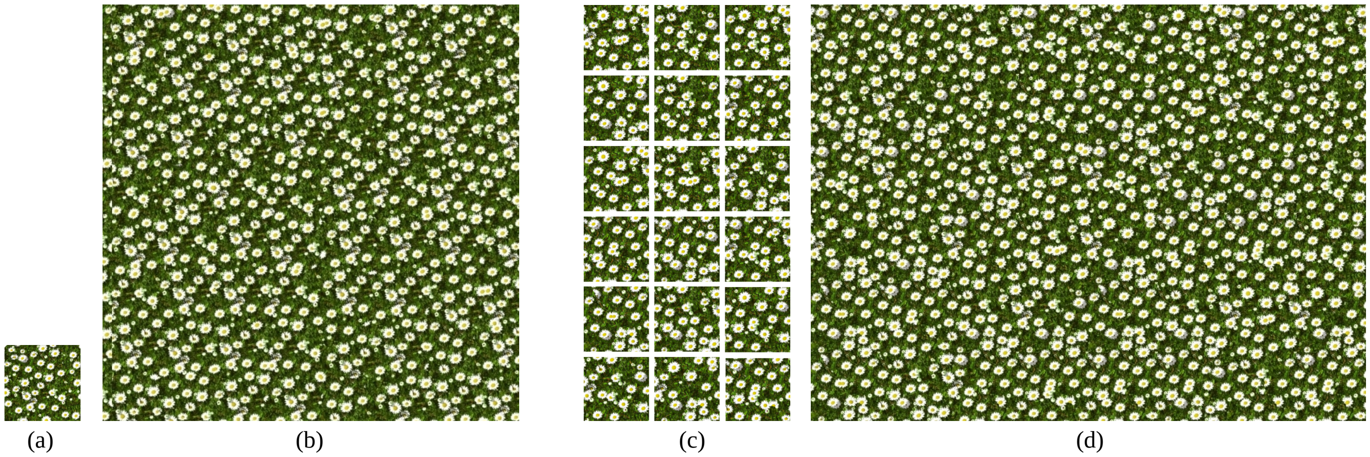


Figure 6: a) Source image; b) texture generated using image quilting; c) 18 Wang Tiles automatically generated based on set in Figure 1(d); d) resulting part of infinite texture (8×6 tiles) with some tile instances highlighted.

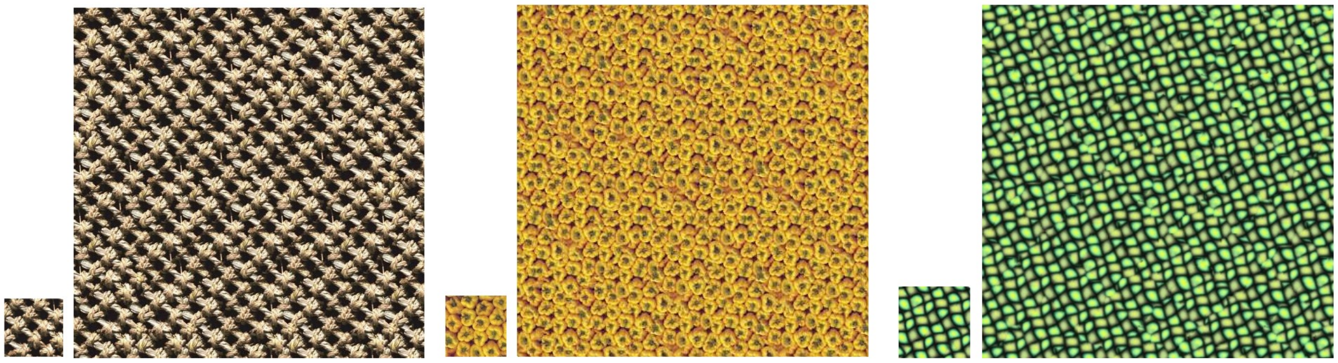


Figure 7: Three textures using 18 Wang Tiles.

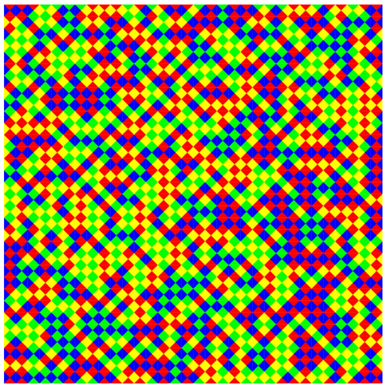


Figure 5: A 32×32 pseudocolored tiling where each diamond shape color represents the label of the sample chosen, or equivalently, one of four edge colors. This shows the non-periodicity achieved with only four subimages.

square sample images obtained randomly from the source image. The sample is fitted to the already created image by optimizing a cutting path in an overlapping region. If the color differences of the pixels along the cutting path are below a given threshold, the sample image is added to the texture.

Each tile is now constructed by combining the four sample diamonds that correspond to the edge colors of the tile. In this case we have to find four cutting paths to combine the four samples to form the tile (cf. Figure 4(a)). This is done using the optimization algorithm given in [Efros and Freeman 2001]. Next, the four combined samples that together have a diamond shape are cut along their diagonals. This forms the final tile slightly smaller than the sample diamonds due to the samples' overlap. Because the same image sample is used for all tiles that share the edge color the tiles will always fit together. If the north edge of a tile is colored red, the lower diagonal part of the "red" diamond is used for synthesizing the upper part of tile, and vice versa for a red south edge. Figure 4 illustrates this process and its use on a set of eight tiles. Figure 5 shows a pseudo-colored result of a tiling with these eight tiles.

Some visible artifacts may remain (see Figure 8) for two reasons: (1) because so few sample patches of the original texture are used, and (2) due to artifacts introduces by the Efros' quilting algorithm. The first reason can be overcome, albeit at some cost, by increasing the number of tiles. Often one can effectively reduce the artifacts, without increasing the number of tiles, by minimizing the quilting errors. To do this, we iterate over sets of samples. For each set, its overall quilting error is the sum of the pixel color errors that occur along all cutting paths (e.g., 32 paths for the eight tile set). If the error is too high, four new sample images are selected, the paths are calculated and the overall error is computed. This is repeated until a small overall error is obtained or a preset number of iterations is completed.

In our trials, a set of eight tiles requires about 3 seconds of optimization to produce good fitting tiles. An 18 tile set takes 15 sec-

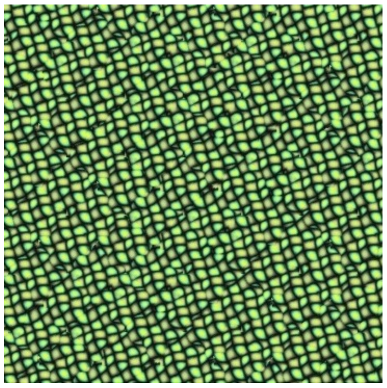


Figure 8: An example of a synthesized texture where the diamond shaped artifacts are apparent.

onds on a standard PC with 2 GHz. Once the set of samples is found, the generation of large textures is very fast. This is a major advantage of tiling or pattern based methods over other texture synthesis techniques where the creation of texture is dependent on the overall amount of texture required [Wei and Levoy 2000; Efros and Freeman 2001].

Figure 6 shows some results. A small image showing some daisies in a meadow is selected (a). In (b) a texture created by the original image quilting algorithms is shown. The tiles of an eighteen tile set are given in (c). (d) shows the result if 8×6 tiles are combined. Figure 7 shows some other examples of images generated by eighteen Wang Tiles.

3.3 Automatic Tile Design for Object Distributions

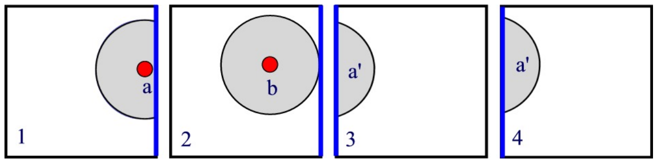


Figure 9: Dart throwing problem for Poisson distributions near edges. The example above illustrates the problem with two tiles with blue east edges and two tiles with corresponding blue west edges. A dart landing at a has a disc (and thus an influence) that extends beyond the border of tile 1 in which it lands. This extension into tiles 3 and 4 appears as the grayed out areas a'. No new darts can land in these regions. A new dart can land at b. Its disc lies fully inside tile 2. This new dart at b will now always have an empty space to its right of almost 2 disc radii since the dart at a has precluded new darts landing nearby in tiles 3 and 4. This illustrates the problem with pure dart throwing for defining a Poisson distribution with Wang Tiles.

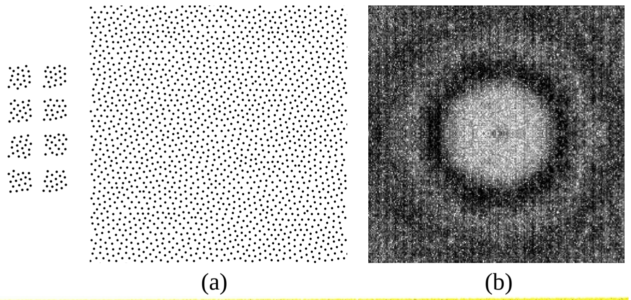
The creation of pseudo-random distributions is another area where Wang Tiles can be used efficiently. For complex ecosystems like those presented in [Deussen et al. 1998], tens of thousands of objects have to be distributed. Instancing can minimize the amount of geometry needed, but even the storage of plant instances can consume lots of memory for large distributions. Alternatively, a few objects can be positioned on a small set of Wang Tiles and then the full population can be generated by combining the tiles non-periodically.

Poisson disc distributions: To represent distributions, the tiles each contain a set of point positions. The goal is find a set of point distributions on each tile, that when combined will result in an overall distribution with desired properties. Purely random distributions are trivial to create by simply placing random positions on the tiles. When combined, the overall distribution remains random since there are no dependencies between points.

However, in many cases purely random distributions are not the best choice. In a crowd each person might position themselves to leave a certain distance from any other person. Biological and environmental forces also do not lead to random placements of plants. In both cases the distribution can often be described by a so called Poisson disc distribution that avoids crowding. In such a distribution, a small disc around each object defines a region in which no other object of the distribution is found.

Using Wang Tiles, the challenge is to find positions on each tile that generate a Poisson disc distribution across all possible tilings. This presents a challenge since points placed near the border of a tile have influence on the placement of points in any other tile with a matching opposite edge.

Often, dart throwing is used to generate Poisson disc distributions: a series of random positions is created one at a time (the darts). Each position is added to the already generated point set if it does not overlap with any of the discs of points already inserted. This works fine (given infinite time) for an infinite plane but is not sufficient in the case of Wang Tiles. If a dart falls near a tile boundary, one has to check all the discs of all points that possibly lie on the opposite tile with the same edge. In essence, for a point near a border, part of its disc is repeated in all possible neighboring tiles. Thus, the total area in all tiles that cannot accept a new dart due to this point is larger than one disc. This causes less points to be inserted near the border (see Figure 9) distorting the distribution in the overall tilings.



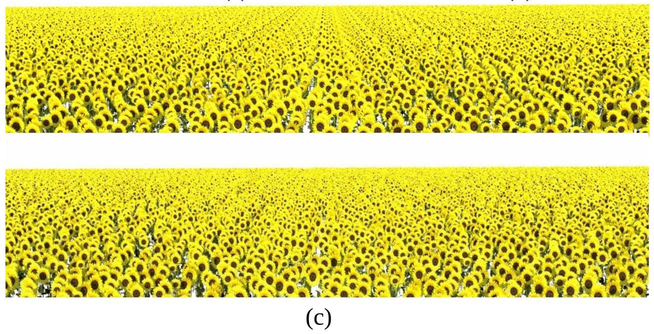


Figure 10: Eight Wang Tiles with joint Poisson distributions, a portion of the plane resulting from these tiles; b) Fourier transform showing some weak grid structure, from [Hiller et al. 2001]; c) comparison of a distribution generated from one tile with 160 plants (upper row) and another one created from 8 tiles with 20 plants each (lower row).

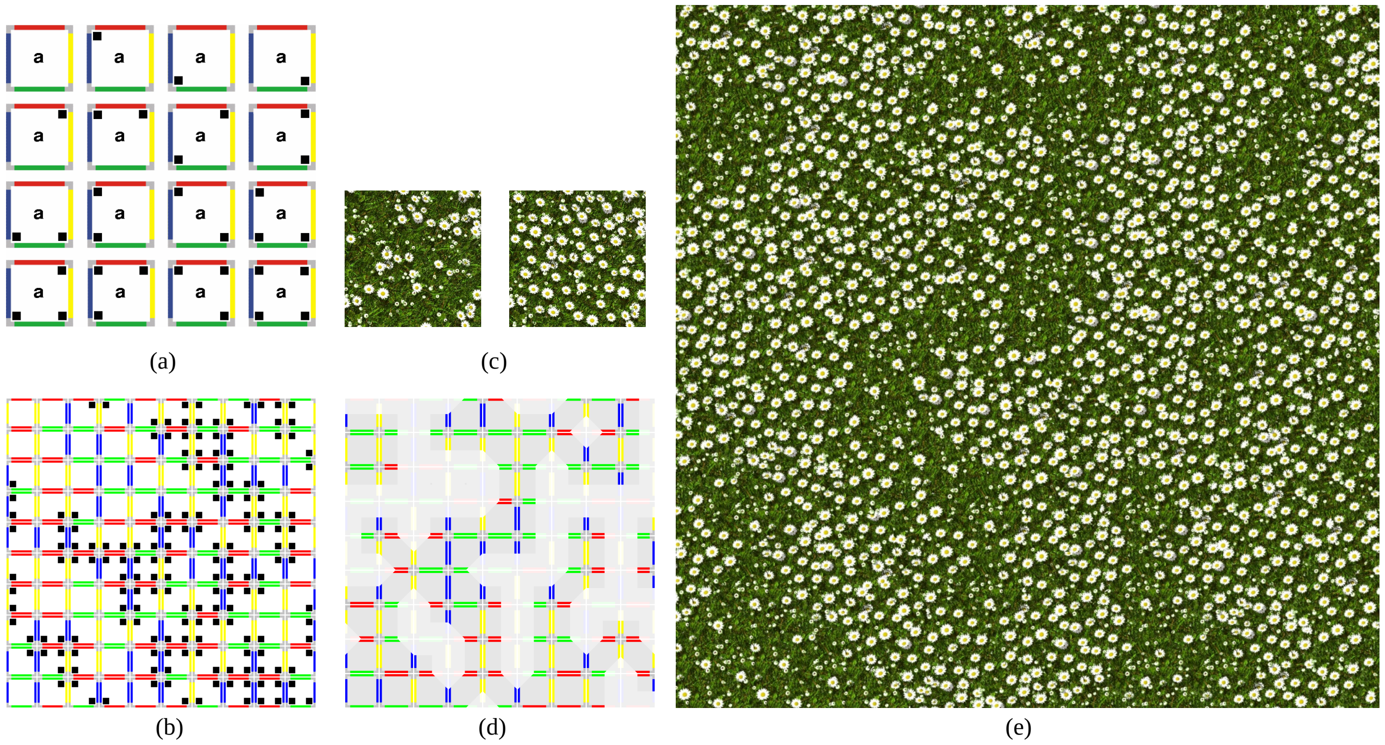


Figure 11: The corner problem: a) A Wang tile with all combinations of marked corners. b) Tiling from the expanded tile set that shows various combinations of marked/unmarked tile corners. Constructing an inhomogeneous tiling: c) two source images; d) tiling that contains density information. e) a tiled plane (12x12) constructed from a tile set with inhomogeneities.

Lloyd's method: An alternative for creating Poisson disc distributions is to use Lloyd's method to optimize an initial set of point positions [Deussen et al. 2000; Hausner 2001; McCool and Fiume 1992]. Lloyd's method has the advantage over dart throwing in that points can interact with each other.

A starting Poisson distribution point set is created independently in each tile. Then, one tile is selected and surrounded by 8 tiles from the set. A Voronoi diagram is constructed across the nine tile set, and the points within the selected center tile are relaxed to satisfy the Poisson disc constraints across the tile boundaries. The local tilings and relaxation steps are repeated several times for all tiles until the distribution stabilizes. This results in consistent appearing point distributions, such as one shown in Figure 10 for a set of eight Wang Tiles. Details and a numerical evaluation of this optimization process can be found in Hiller et al. [Hiller et al. 2001] and as an application to sampling in Deussen [Deussen et al. 2000]. Figures 12 and 13(e) show this process applied to positioning sunflowers in a simulated sunflower field.

3.4 The Corner Problem

Edge coloring and the texture synthesis process ensures that the tiles always fit together. But a problem arises if an object or a visible artifact of the texture is placed across a corner of a tile. In this case the vertical edge color constraint enforces all tiles with same colored opposite vertical edges to include the remainder of this object to ensure the fitting condition. As the same object is also on the horizontal edge, all tiles that contain the corresponding horizontal edges have to be adapted similarly. In total, in a two colored set of eight tiles six tiles have to be adapted, in a three colored set (18 tiles) twelve tiles. This results in visible repetitive patterns.

We are able to create a tile set that avoids such corner problems, albeit at the cost of adding more tiles to the set. As shown in the next paragraph, the solution also offers the possibility to generate tilings that contain inhomogeneities.

A solution is found by coding the corners of a tile as an additional bit of information, essentially coloring the corners as well as the edges[3]. For a single tile, $2^4 = 16$ possibilities of corner codings can occur (cf. Figure 11(a)). A set of tiles with two colored edges in this case is expanded to $16\times8=128$ tiles. The number can be reduced to 64 tiles and still maintain the stochastic selection since there are only three corners that must match as each tile is inserted. In the expanded set for each of these two possibilities all corner combinations in the tile have to be stored. In Figure 11(b) a tiling with marked and unmarked corners in arbitrary combinations is shown.

3.5 Introducing Inhomogeneity

So far the generated textures and distributions are quite homogeneous. The selection process for textures and object positions enables us to avoid visible patterns, but the generated textures appear uniform. To improve the appearance it would be helpful to allow the tiling process to locally change the texture content.

The corner marked tile sets can help here. To create an inhomogeneous tiling, the texture synthesis process takes two source images instead of one. The tiles are generated by choosing one source image for creating all those parts of a tile with marked corners and the other one for the rest (cf. Figures 4(b) and 11(c)). Of course, the two images must potentially fit together via the quilting algorithm to achieve natural textures.

This enables us to create tilings with two different density levels as shown in Figure 11(c). Here, the marked corners are shown as regions of different density. In Figure 11(e) a result can be seen using

[3]It should be noted that the triangular tiles in [Neyret and Cani 1999] already take into account the corners as part of their coding as well.

two images of daisies, one with a high daisy density and the other one with low density. The texture quilting optimization process in this case lasts longer (about 1 minute on a 2 GHz PC). But, once this is run, generating large amounts of textures at runtime is relatively free both in texture memory and time.

4 Three Dimensional Applications

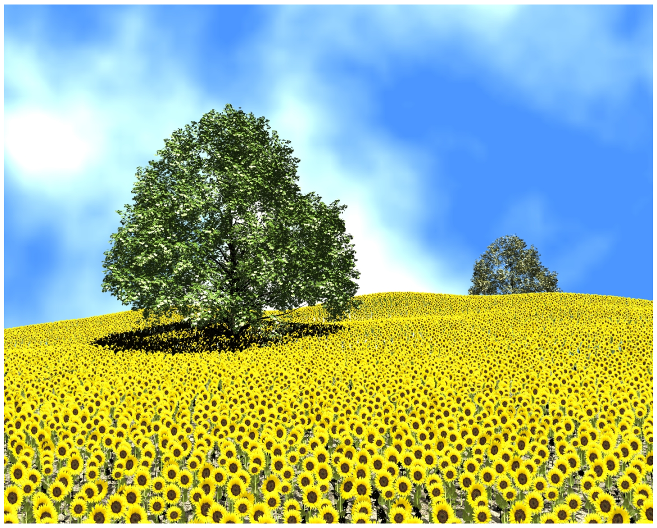


Figure 12: A Wang tiled field filled with sunflowers

The manual effort required to model natural environments and the computational cost required to render such scenes can both be very high. Systems that attempt to render realistic looking natural environments have solved either one problem, or the other, but not both. The systems presented in Weber and Penn [Weber and Penn 1995], Mech and Prusinkiewicz [Měch and Prusinkiewicz 1996], Deussen et al. [Deussen et al. 1998], and Prusinkiewicz et al. [Prusinkiewicz et al. 2001] are capable of creating realistic plants and environments. However, the models created by these systems are of such high detail that they can not be rendered in real-time. Deussen et al. [Deussen et al. 2002] achieve frame rates from 4-8 fps for complex vegetation, but they need lots of memory to store their plant models. This makes their approach not so well suited for larger areas. At the other end of the spectrum, geometric level-of-detail algorithms [Hoppe 1996; Perbert and Cani 2001] have increased the effective geometric complexity that can be rendered in real-time, but provide no direct support for realistic shading.

We address both problems with the use of Wang Tiles as containers for 3D geometry. In our example application of a sunflower field, we use tiles to group a set of flowers into a single lighting environment. We use eleven variations of a model of a sunflower and position them stochastically in each of 8 Wang Tiles based on the Poisson disc distribution discussed earlier. Sunflowers that intersect a tile boundary are repeated in all tiles with opposite matching colors. A tiling draped over a height field can be rendered with a standard ray tracer as seen in Figure 12.

4.1 Layered Depth Image Tiles

A lower quality, but real-time rendering can be achieved by Wang Tiles in which the geometry within each tile is pre-shaded. This is possible since Wang Tiles are not allowed to be reoriented when constructing the tilings. We discuss one possible represention in which the geometry is sampled into a hierarchy of 3D textures. The

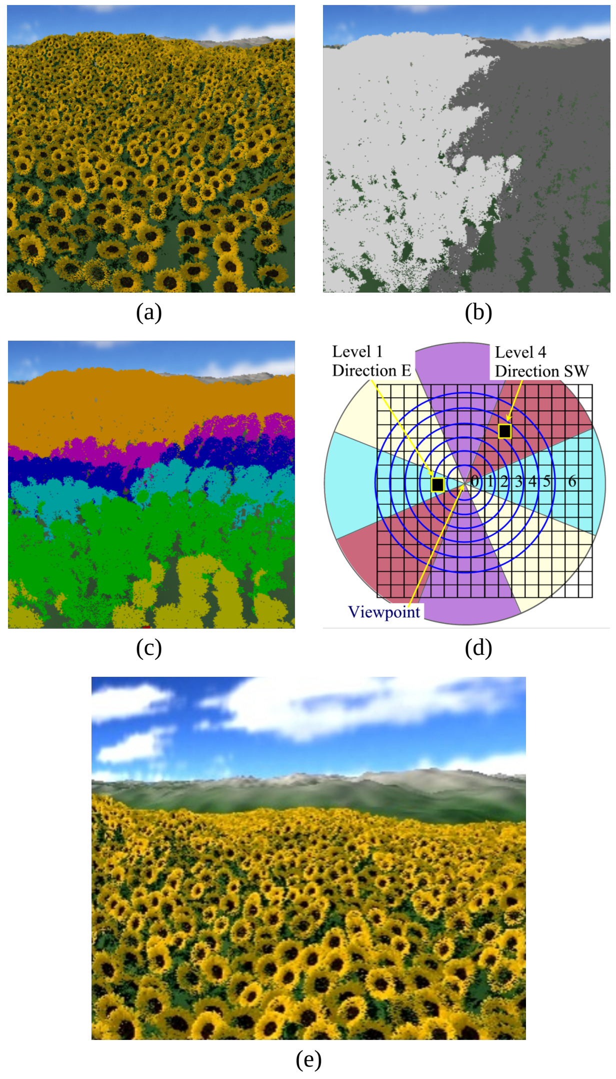


Figure 13: LOD selection. (a) A typical portion of the sunflower field. (b) This view straddles two of the 8 wedges that divide the space of directions around a tile. (c) Levels of detail false-colored. Red (not seen below us) is highest detail, orange is lowest detail. (d) Layout of directions and resolution levels from viewpoint; e) A Frame from interactive LDI renderer with 20 flowers per tile. Sky is environment mapped, mountains are geometry. Frames rendered at between 2.7 and 4.0 fps on a P4 1.7 GHz PC.

3D textures are each represented as Layered Depth Images (LDIs) that convert complex geometry into a sparse set of pre-shaded samples [Shade et al. 1998].

Real time rendering is enhanced by extending LDIs with a novel view-dependent multi-resolution formulation of the representation. Since tiles further from the viewpoint require fewer samples, we create 7 levels of detail of each tile, each with successively fewer sample points. Tiles viewed from a particular direction need include only those points of geometry seen from that direction. To address this we create 8 variations of each tile specialized for viewing from each of 8 compass directions.

The complete system consists of an offline and an online component. Offline, we create and pre-shade the hierarchical set of tiles composed of instances of the eleven sunflower models. At runtime, we compute a tiling of the plane that is draped over a terrain and render the visible tiles in back to front order. Each tile is selected from the appropriate hierarchy level and from the appropriate viewing direction based on its distance and orientation to the camera. Each sample point within each tile is splatted to the screen in back to front order as described in the original LDI paper [Shade et al. 1998] (see Figure 13).

Because we focus here on the application and construction of the Wang Tiles, we leave the details of how the original geometry is sampled to create the hierarchical LDIs to a technical report [Shade et al. 2002].

We are able to render the sunflower field in real-time on a 2GHz PC with an Nvidia GeForce4 graphics card. Across the data sets, our software renderer was able to maintain a rendering rate of between 4.5 and 5.7 million depth pixels per second. This corresponds to about 4 fps for the sunflower field. Given the point-rendering capabilities of modern graphics hardware, namely point-sprites in modern graphics APIs, we believe point-sampled tiles can reach 30 fps.

5 Discussion and Future Work

We have shown the versatility and usefulness of Wang Tiles for image and texture synthesis, and for generating large point distributions. Wang Tiles were interesting in their own right as a theoretical entity. They should also now prove their usefulness in a variety of practical graphics applications.

Computer games can benefit from the extremely compact representation for texture using tiles. We also look forward to seeing how well a hardware point based renderer works with the LDI tiles.

Based on a reviewer comment, we are also investigating the possibility of generating the Poisson distributions with an algorithm similar to the way we texture the tiles in section 3.2.

In addition, techniques that use volumetric textures [Neyret January-March 1998] or slices [Perbert and Cani 2001] can potentially save a lot of memory by tiling non-periodically with only small samples of their data represented in the tiles.

Another three-dimensional application is to create tilings of volumes with Wang-like cubes. In this case, each tile is a small cube with sides colored. Doing so, we would need a set of 32 tiles to tile the space non-periodically. Applications for such tilings can be particle systems that represent clouds or stars in galaxies. We are certain there are many other applications of these elegant entities we have not thought of.

Acknowledgements

The authors would like to acknowledge the many useful discussions with Don Mitchell during preparation of an early version of this work. We also wish to thank the reviewers (both in this and past years) for many helpful comments.

References

BERGER, R. 1966. The undecidability of the domino problem. *Memoirs American Mathematical Society*, 66, 1–72.

CULIK II, K. 1996. An aperiodic set of 13 wang tiles. *Discrete Mathematics 160*, 245–251.

DEUSSEN, O., HANRAHAN, P., PHARR, M., LINTERMANN, B., MĚCH, R., AND PRUSINKIEWICZ, P. 1998. Realistic modeling and rendering of plant ecosystems. In *SIGGRAPH 98 Conference Proceedings*, 275–286.

DEUSSEN, O., HILLER, S., VAN OVERVELD, K., AND STROTHOTTE, T. 2000. Floating points: A method for computing stipple drawings. *Computer Graphics Forum, Eurographics 2000 Conference Proceedings 19*, 4, 40–51.

DEUSSEN, O., COLDITZ, C., STAMMINGER, M., AND DRETTAKIS, G. 2002. Efficient rendering of complex ecosystems using points and lines. In *IEEE Visualization 2002*, IEEE, 219–226.

EFROS, A., AND FREEMAN, W. 2001. Image quilting for texture synthesis. In *Proceedings of SIGGRAPH 2001*, 341–346.

GLASSNER, A. 1998. Aperiodic tiling, part 1. *IEEE Computer Graphics and Applications 18*, 4 (May).

GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. 1996. The lumigraph. In *Proceedings of SIGGRAPH 1996*, 43–54.

GRÜNBAUM, B., AND SHEPHARD, G. C. 1987. *Tilings and Patterns*. W. H. Freeman and Company. ISSN 0716711931.

HAUSNER, A. 2001. Simulating decorative mosaics. In *SIGGRAPH 2001 Conference Proceedings*, 573–578.

HILLER, S., DEUSSEN, O., AND KELLER, A. 2001. Tiled blue noise samples. In *Proceedings of Vision Modelling Visualization 2001*, Infix-Verlag, 265–272.

HOPPE, H. 1996. Progressive meshes. In *Proceedings of SIGGRAPH 1996*, 99–108.

KARI, J. 1996. An small aperiodic set of wang tiles. *Discrete Mathematics 160*, 259–264.

LEVOY, M., AND HANRAHAN, P. 1996. Light field rendering. In *Proceedings of SIGGRAPH 1996*, 31–42.

MCCOOL, M., AND FIUME, E. 1992. Hierarchical poisson disk sampling distributions. In *Graphics Interface '92*, 94–105.

MĚCH, R., AND PRUSINKIEWICZ, P. 1996. Visual models of plants interacting with their environment. In *Proceedings of SIGGRAPH 1996*, 397–410.

NEYRET, F., AND CANI, M.-P. 1999. Pattern-based texturing revisited. In *Proceedings of SIGGRAPH 1999*, 235–242.

NEYRET, F. January-March 1998. Modeling, animating, and rendering complex scenes using volumetric textures. *IEEE Transactions on Visualization and Computer Graphics 4*, 1, 55–70.

PERBERT, F., AND CANI, M. 2001. Animating prairies in real-time. In *2001 ACM Symposion on Interacive 3D Graphics*, 103–110.

PRUSINKIEWICZ, P., MÜNDERMANN, L., KARWOWSKI, R., AND LANE, B. 2001. The use of positional information in the modelling of plants. In *Proceedings of SIGGRAPH 2001*, 289–300.

SHADE, J., GORTLER, S., HE, L., AND SZELISKI, R. 1998. Layered depth images. In *Proceedings of SIGGRAPH 1998*, 231–242.

SHADE, J., COHEN, M. F., AND MITCHELL, D. 2002. Tiling layered depth images. ftp://ftp.cs.washington.edu/tr/2002/12/UW-CSE-02-12-07.pdf.

STAM, J. 1997. Aperiodic texture mapping. Tech. rep., R046. European Research Consortium for Informatics and Mathematics (ERCIM). http://www.ercim.org/publication/technical_reports/046-abstract.html.

TONG, X., ZHANG, J., LIU, L., WANG, X., GUO, B., AND SHUM, H.-Y. 2002. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics 21*, 3 (July), 665–672. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002).

TURK, G. 2001. Texture synthesis on surfaces. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, 347–354. ISBN 1-58113-292-1.

WANG, H. 1961. Proving theorems by pattern recognition II. *Bell Systems Technical Journal 40*, 1–42.

WANG, H. 1965. Games, logic, and computers. *Scientific American* (November), 98–106.

WEBER, J., AND PENN, J. 1995. Creation and rendering of realistic trees. In *Proceedings of SIGGRAPH 1995*, 119–128.

WEI, L.-Y., AND LEVOY, M. 2000. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of SIGGRAPH 2000*, 479–488.