KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT COMPUTERWETENSCHAPPEN
AFDELING INFORMATICA
Celestijnenlaan 200 A — B-3001 Leuven

TILE-BASED METHODS IN COMPUTER GRAPHICS

Promotor:
Prof. dr. ir. Philip Dutré

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Ares LAGAE

April 2007

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT COMPUTERWETENSCHAPPEN
AFDELING INFORMATICA
Celestijnenlaan 200 A — B-3001 Leuven

TILE-BASED METHODS IN COMPUTER GRAPHICS

Examencommissie:
Prof. dr. ir. Jean Berlamont, voorzitter
Prof. dr. ir. Philip Dutré, promotor
Prof. dr. ir. Ronald Cools
Prof. dr. ir. Marc Van Barel
Prof. dr. Philippe Bekaert (Universiteit Hasselt)
Prof. dr. Marc Stamminger
(Friedrich-Alexander-Universität Erlangen-Nürnberg)

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Ares LAGAE

UDC 681.3∗I3

April 2007

©Katholieke Universiteit Leuven – Faculteit Ingenieurswetenschappen
Arenbergkasteel, B-3001 Heverlee (Belgium)

D/2007/7515/23
ISBN 978–90–5682–789–2

Preface

This dissertation is the result of more than four years of research. During that period, many people have helped me, and now the time has finally come to thank them.

First, I would like to thank my advisor prof. dr. ir. Philip Dutré for supporting me, and for giving me the freedom to pursue my own ideas.

I would also like to thank my assessors prof. dr. ir. Ronald Cools and prof. dr. ir. Marc Van Barel for carefully reading this dissertation.

I am grateful to prof. dr. Philippe Bekaert and prof. dr. Marc Stamminger for accepting to be part of my committee. I am also grateful to prof. dr. ir. Jean Berlamont for chairing the committee.

I would like to thank my former and current colleagues Frank Suykens, Vincent Masselus, Pieter Peers, Karl vom Berge, Frederik Anrys, Bart Adams, Olivier Dumont, Muath Sabha, Peter Vangorp, Toon Lenaerts and Jurgen Laurijssen for a stimulating research environment and for the many relaxing coffee breaks.

I would also like to thank Wang Yue, Tuen-Young Ng and Tiow-Seng Tan for generating the texture tile sets based on corner tiles, Stefan Hiller for making available the data of [Hiller et al., 2001], the authors of [Ostromoukhov et al., 2004], [Jones, 2006], [Kopf et al., 2006] and [Dunbar and Humphreys, 2006] for making available example code, and Jarkko Kari for many insightful comments.

I am grateful for funding by the Katholieke Universiteit Leuven (2002–2003) and as Aspirant van het Fonds Wetenschappelijk Onderzoek — Vlaanderen (2003–2007).

Last but not least, I would like to thank my friends, Celine's family, my family, and especially Celine. You all helped me to get to this point.

Ares Lagae
Heverlee, April 2007

Abstract

Many complex signals, such as point distributions and textures, cannot efficiently be synthesized and stored. In this dissertation we present tile-based methods to solve this problem. Instead of synthesizing a complex signal when needed, the signal is synthesized on forehand over a small set of tiles. Arbitrary large amounts of that signal can then efficiently be generated when needed by generating a stochastic tiling.

Tile-based methods are traditionally based on Wang tiles. The colored edges of Wang tiles only constrain the four direct neighboring tiles, but not the four diagonally neighboring tiles. This problem introduces unwanted artifacts in the tiled signals, and complicates methods for synthesizing signals over a set of Wang tiles. To solve this problem we present corner tiles. Corner tiles are unit square tiles with colored corners rather than colored edges. The colored corners of corner tiles constrain all neighboring tiles. We revisit the most important applications of Wang tiles, and we show that corner tiles have substantial advantages for each of these applications.

Stochastic tilings are traditionally generated using scanline stochastic tiling algorithms. However, these algorithms store the complete tiling and are therefore not efficient. To solve this problem, we present direct stochastic tiling algorithms for Wang tiles and corner tiles. These algorithms are capable of evaluating a stochastic tiling locally, without explicitely constructing and storing the tiling up to that point. We also introduce long-period hash functions to generate very large tilings.

Poisson disk distributions and textures are two examples of complex signals. We present tile-based methods for generating Poisson disk distributions and for synthesizing textures. Tile-based methods not only allow to efficiently generate Poisson disk distributions and synthesize textures, but also enable new applications such as tile-based texture synthesis and a procedural object distribution function. This new texture basis function allows to distribute procedural objects over a procedural background, using intuitive parameters such as the scale, size and orientation of the objects. We also present an overview of applications of tiled Poisson disk distributions, and a detailed comparison of methods for generating Poisson disk distributions. We study corner tiles in the context of the tiling problem and aperiodic tile sets, and we construct several new aperiodic sets of Wang tiles and corner tiles.

The tile-based methods we present in this dissertation are a valuable tool for computer graphics, and help to keep up with the continuously increasing demand for more complexity and realism in digitally synthesized images.

Contents

Chapter 1

Introduction

1.1 Tile-Based Methods in Computer Graphics

Computer graphics is a very diverse field of research with many applications, including film and visual effects, advertising, car and flight simulators, architecture, scientific simulations and computer games. These applications are the driving force behind computer graphics and the continuous demand for more quality and complexity in digitally synthesized images.

A common problem in the field of computer graphics is the synthesis and storage of complex signals, such as point distributions or textures. For several of these complex signals, no efficient synthesis algorithms are available, and storing large quantities of these signals is expensive. Tile-based methods provide a solution for both these problems.

As a simple example, consider the use of textures in interactive computer games. A commonly used technique to create the impression of a large texture is tiling a small square texture. This technique clearly avoids synthesizing and storing a large texture, but also introduces visually disturbing artifacts. The large texture is repeating and tile seams are visible. The challenge of tile-based methods is to generate a tiled complex signal as similar as possible to the original complex signal, without obvious repetition and tile seams.

This dissertation presents high-quality tile-based methods based on Wang tiles and corner tiles. Wang tiles are unit square tiles with colored edges, and corner tiles are unit square tiles with colored corners. Wang tiles and corner tiles have a fixed orientation. A tiling is generated by placing the tiles next to each other, such that adjoining edges or corners have matching colors.

Rather than synthesizing a complex signal directly, the signal is synthesized over a small set of tiles on forehand. Arbitrary large quantities of that signal can then efficiently be obtained when needed simply by generating a tiling. The complex signal is synthesized consistently with the continuity constraints imposed by the colored edges. This ensures that no tile seams are noticeable in the tiled complex signals. The tiled signals are generated using stochastic tilings. This ensures that no repetition is noticeable.

A tile-based method for generating a complex signal consists of a method for synthesizing the complex signal over a set of tiles, and a method for generating a stochastic tiling using the set of tiles. The method for synthesizing the complex signal over a set of tiles is dependent on the signal and is typically expensive. The method for generating a stochastic tiling using the set of tiles is independent of the signal and is typically inexpensive. Once the complex signal is synthesized over a set of tiles, arbitrary large quantities of that signal can be generated very efficiently by generating a stochastic tiling. The tile sets are usually small and therefore reduce storage requirements.

This dissertation introduces corner tiles as a better alternative for Wang tiles. The colored edges of Wang tiles only constrain the four direct neighboring tiles, but not the diagonally neighboring tiles. This leads to unwanted artifacts in the tiled complex signals and complicates methods for constructing complex signals over a set of Wang tiles. Corner tiles are not subject to this problem.

This dissertation introduces efficient tiling algorithms for generating stochastic tilings using Wang tiles and corner tiles, and methods for constructing Poisson disk distributions and synthesizing textures over a set of Wang tiles and corner tiles. Although the methods for constructing a complex signal over a set of tiles are dependent on the signal, the general idea behind the methods presented in this dissertation should generalize to other kinds of signals.

Poisson disk distributions are stochastic point distributions in which all points are separated by a minimum distance. Poisson disk distributions have several applications in computer graphics, such as sampling and object distribution. However, no efficient algorithms are available for generating Poisson disk distributions. Constructing a Poisson disk distribution over a set of Wang tiles or corner tiles is challenging, because the minimum distance criterion should be respected over tile boundaries. This dissertation also includes an overview of applications of tiled Poisson disk distributions, and a detailed comparison of several methods for generating Poisson disk distributions.

Textures are ubiquitous in computer graphics, and methods for efficiently synthesizing textures are clearly of interest. Tile-based methods for texture synthesis are an interesting alternative for existing texture synthesis techniques, because the process of texture synthesis is broken up into two parts. In a first part, a texture is synthesized over a set of tiles. In a second part, an arbitrary large texture can be generated very efficiently simply by generating a tiling.

The tile-based methods presented in this dissertation enable efficient generation of Poisson disk distributions and rapid synthesis of textures, but also enable new applications, such as a procedural object distribution in the case of Poisson disk distributions, and tile-based texture mapping in the case of texture synthesis.

Corner tiles are also investigated in the context of the tiling problem and aperiodic tile sets, which is the original context of Wang tiles. Several new aperiodic sets of Wang tiles and corner tiles are introduced in this dissertation.

The methods introduced in this dissertation help to manage the continuous demand for more quality and complexity in digitally synthesized images, and are a valuable tool for computer graphics.

1.2 Contributions

The main contributions of this dissertation are the following.

- We present corner tiles as an alternative for Wang tiles. We revisit the most important applications of Wang tiles, and we show that corner tiles have substantial advantages for each of these applications. We also show that corner tiles result in cleaner, simpler and more efficient applications.
- We introduce direct stochastic tiling algorithms for Wang tiles and corner tiles. Direct stochastic algorithms are capable of evaluating a stochastic tiling locally, without explictly constructing the tiling up to that point. We also present long-period hash functions defined over the integer lattice.
- We propose several tile-based methods for efficiently generating Poisson disk distributions. We also introduce tile-based methods for efficiently generating Poisson sphere distributions and nonuniform Poisson disk distributions.
- We demonstrate a tile-based texture mapping algorithm running on the graphics processing unit, based on corner tiles. The algorithm is faster and uses less texture memory than the existing algorithm based on Wang tiles.
- We present a detailed comparison of methods for generating Poisson disk distributions, including the methods introduced in this dissertation.
- We introduce a procedural object distribution function, a new procedural texture basis function that extends the range of textures that can be generated procedurally.
- We give several methods for constructing small aperiodic sets of corner tiles and small aperiodic sets of Wang tiles.

A more detailed overview of the contributions of this dissertation is provided in chapter 9. A complete list of publications is provided on page 173.

1.3 Overview

This dissertation is organized as follows.

Chapter 2 is introductory. This chapter introduces tilings, Wang tiles and corner tiles, discusses previous applications of tilings in computer graphics, and introduces several useful definitions, conventions and notations.

Chapter 3 presents efficient algorithms for generating stochastic tilings with Wang tiles and corner tiles. This chapter presents scanline stochastic tiling algorithms and direct stochastic tiling algorithms for Wang tiles and corner tiles, and long-period hash functions defined over the integer lattice, used in direct stochastic tiling algorithms.

Chapter 4 introduces Poisson disk distributions and presents several tile-based methods for generating Poisson disk distributions. This chapter shows how to construct a Poisson disk distribution over a set of Wang tiles and corner tiles, and introduces tile-based methods for generating Poisson sphere distributions and nonuniform Poisson disk distributions.

Chapter 5 introduces tile-based methods for texture mapping and texture synthesis. This chapter shows how to synthesize a texture over a set of Wang tiles and corner tiles, presents an efficient tile-based texture mapping algorithm running on the GPU, and discusses the tile packing problem.

Chapter 6 presents a detailed comparison of almost all existing methods for generating Poisson disk distributions, including the methods presented in chapter 4.

Chapter 7 discusses several applications of Poisson disk distributions. These applications include sampling, non-photorealistic rendering, scientific visualization, procedural modeling, and procedural texturing.

Chapter 8 introduces several methods for constructing small aperiodic sets of corner tiles. This chapter presents several small aperiodic sets of corner tiles and also several new aperiodic sets of Wang tiles.

Chapter 9 concludes the dissertation. This chapter includes a summary, a detailed overview of the original contributions of this dissertation, and some directions for future research.

1.4 Notes

This dissertation includes several figures that are difficult to reproduce in print, for example the power spectra and the images for measuring sampling performance of chapter 6. Therefore, the printed version of this dissertation is accompanied by a full resolution electronic version. In the full resolution electronic version, all illustrations are included as vector graphics, and all images are included in full resolution. The reader is encouraged to use the zoom function of the electronic document viewer where appropriate. The full resolution electronic version of this dissertation can also be obtained at `http://www.cs.kuleuven.be/`.

This dissertation describes at several occasions different variants of the same technique, for example in section 3.2 (subsection 3.2.1 and subsection 3.2.2), in section 3.3 (subsection 3.3.1 and subsection 3.3.2) and in chapter 4 (section 4.3, section 4.5 and section 4.6). We have chosen to describe each variant of a technique completely, rather than only pointing out the differences with the previous variant. This makes it easier to use this dissertation as a reference, or to implement a specific variant of a technique. However, it also causes some repetition.

Chapter 2

Wang Tiles and Corner Tiles

2.1 Introduction

The tile-based methods presented in this dissertation are based on Wang tiles and corner tiles. In this chapter we introduce Wang tiles and corner tiles. We briefly sketch their history, and introduce key concepts that will be used in later chapters.

Overview

This chapter is organized as follows. Section 2.2 introduces tilings. In section 2.3 we discuss previous applications of tilings in computer graphics. Section 2.4 introduces Wang tiles and briefly sketches their background. In section 2.5 we discuss applications of Wang tiles in computer graphics. Section 2.6 explains the corner problem and introduces corner tiles. In section 2.7 we introduce definitions, conventions and notations. Section 2.8 proposes a convenient scheme for enumerating Wang tile sets and corner tile sets. In section 2.9 we explain the close relationship between Wang tiles and corner tiles. Section 2.10 discusses generalizations of Wang tiles and corner tiles to arbitrary dimensions. In section 2.11 we conclude.

2.2 Tilings

Tilings are in abundance all around us. Not only man-made, but also occurring in nature. Some of the most famous examples of tilings can be seen in the Alhambra at Granada, Spain [Saladin, 1926], and in the work of the Dutch artist M. C. Escher [Escher and Locher, 1971].

A tiling is an arrangement of plane figures that fills the plane without gaps or overlaps, or its generalization to higher dimensions. Each plane figure is a tile. The set of plane figures used in the tiling is the tile set. To tile means to cover the plane with the tiles.

A tiling is periodic if a translation exists that maps the tiling to itself. If this is not the case, the tiling is non-periodic. An aperiodic tile set is a tile set that

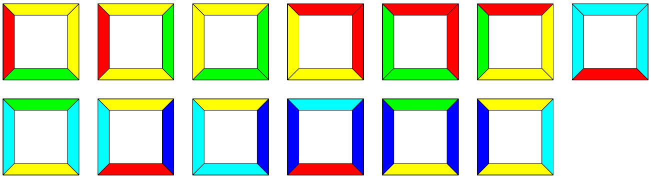


Figure 2.1: The smallest aperiodic Wang tile set currently known.

does not admit a periodic tiling. A tiling generated by an aperiodic tile set is an aperiodic tiling.

The classic work on tilings is *Tilings and Patterns* [Grünbaum and Shepard, 1986]. A good introductory text on aperiodic tilings can be found in *Andrew Glassner's Notebook: Recreational Computer Graphics* [Glassner, 1999, chapter 12].

2.3 Tilings in Computer Graphics

Most applications of tilings in computer graphics simulate tilings in the real world. Kaplan and Salesin [2000] used isohedral tilings to provide a solution to the problem of *Escherization*: given a closed figure in the plane, find a new closed figure that is similar to the original and tiles the plane. Their system creates illustrations much like the ones by the Dutch artist M. C. Escher. Hausner [2001] presented a system for generating decorative tile mosaics. Ostromoukhov et al. [2004] used a hierarchically subdivided Penrose tiling to generate well-distributed point sets.

2.4 Wang Tiles

The tiles we focus on in this dissertation are Wang tiles. A Wang tile set is a finite set of square tiles. The tiles are all the same size, and each edge of a tile has a fixed color. The colors are combined in several specified ways. The plane is tiled using arbitrary many copies of the tiles in the tile set, in such a way that adjoining edges have the same color.

Wang tiles were first proposed by Wang in 1961, and later popularized in an article in *Scientific American* [Wang, 1965]. Wang presented an algorithm to decide whether a given set of Wang tiles could tile the plane. He relied on the conjecture that aperiodic tile sets, tile sets that do not admit periodic tilings, do not exist.

This conjecture was in 1966 refuted by Berger. He showed that any Turing machine can be translated into a Wang tile set, and that the Wang tile set tiles the plane if and only if the Turing machine will never halt. The halting problem is undecidable and thus so is Wang's original problem.

Berger constructed the first aperiodic tile set counting 20426 tiles. This number was reduced repeatedly, often by well known scientists, such as Knuth [1968]. The smallest aperiodic set of Wang tiles consists of 13 tiles over 5 colors [Culik, 1996], and is shown in figure 2.1.

Not only Wang tiles allow the construction of aperiodic tile sets. In 1974, Penrose discovered his famous kite and dart, an aperiodic set of only two tiles. Whether a single aperiodic tile exists is still an open question.

2.5 Wang Tiles in Computer Graphics

Computer graphics is often concerned with the synthesis of complex signals. Wang tiles are an important tool to facilitate the generation of such signals. Instead of synthesizing a complex signal directly, the signal is constructed over a small set of Wang tiles, consistent with the continuity constraints imposed by the colored edges. This is usually more difficult than synthesizing the signal directly, but once the signal is synthesized over the tile set, arbitrary large quantities of this signal can be generated very efficiently by generating a tiling.

Wang tiles were introduced in the field of computer graphics by Stam [1997] who created non-repeating textures of arbitrary size using an aperiodic set of Wang tiles. Shade et al. [2000] and Hiller et al. [2001] used Wang tiles to generate Poisson disk distributions. The latter approach was later adopted by Cohen et al. [2003], in a paper that popularized Wang tiles in the field of computer graphics. The same paper introduced a method for texture synthesis using Wang tiles. Wei [2004] proposed tile-based texture mapping on graphics hardware. Fu and Leung [2005] recently extended texture tiling to surfaces with arbitrary topology.

2.6 Corner Tiles and the Corner Problem

Wang tiles soon proved to be a valuable tool for constructing complex signals in real time. However, the colored edges of Wang tiles do not guarantee continuity of the signal near tile corners. Wang tiles do not constrain their diagonal neighbors. This is illustrated in figure 2.2(a). Any two Wang tiles can be put diagonally to each other by adding two suitable tiles to complete the tiling. This problem, called the corner problem, complicates construction methods and causes unwanted artifacts in the generated signals.

In order to solve the corner problem, we proposed corner tiles [Lagae and Dutré, 2006a], square tiles with colored corners. Corner tiles are similar to Wang tiles, but their colored corners ensure continuity of the signal over both tile edges and tile corners, thus avoiding the corner problem. This is illustrated in figure 2.2(b).

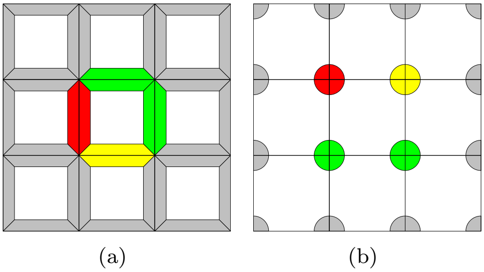


Figure 2.2: The corner problem. (a) Wang tiles only enforce continuity with their four direct neighbors and do not constrain their diagonal neighbors. (b) Corner tiles enforce continuity with all their neighbors.

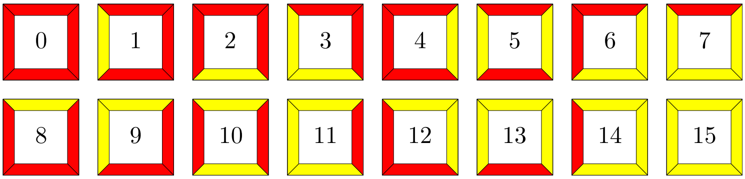


Figure 2.3: The complete Wang tile set over 2 colors.

Cohen et al. [2003] first identified the corner problem. They superimpose corner markings on a Wang tile set in an attempt to solve the problem. Although this allowed them to synthesize textures with different densities, the corner problem remains: for a given corner marking, any two tiles can be put diagonally next to each other. They did not make the observation that the edge colors should be dropped altogether to adequately solve the corner problem.

To our knowledge, tiles with colored corners have not been used previously in computer graphics (or in other domains), except by Ng et al. [2005], who presented a technique for assembling a set of tiles similar to corner tiles from an input texture to synthesize larger textures. Their technique is discussed in detail in section 5.3. Neyret and Cani [1999] use triangular tiles with edge and corner colors to generate pattern-based textures over a triangle mesh, in the spirit of Stam [1997].

2.7 Definitions, Conventions and Notations

Wang tiles are unit square tiles with colored edges. The edges of a Wang tile are named after the compass headings north (N), east (E), south (S) and west (W). The colors of the edges are indicated by c_N, c_E, c_S and c_W.

Corner tiles are unit square tiles with colored corners. The corners of a corner

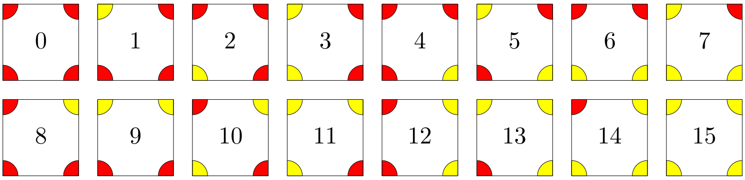


Figure 2.4: The complete corner tile set over 2 colors.

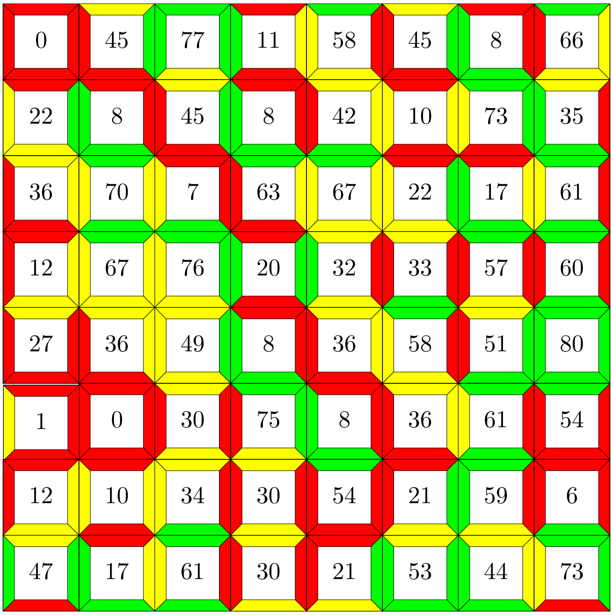


Figure 2.5: A tiling with the complete Wang tile set over 3 colors.

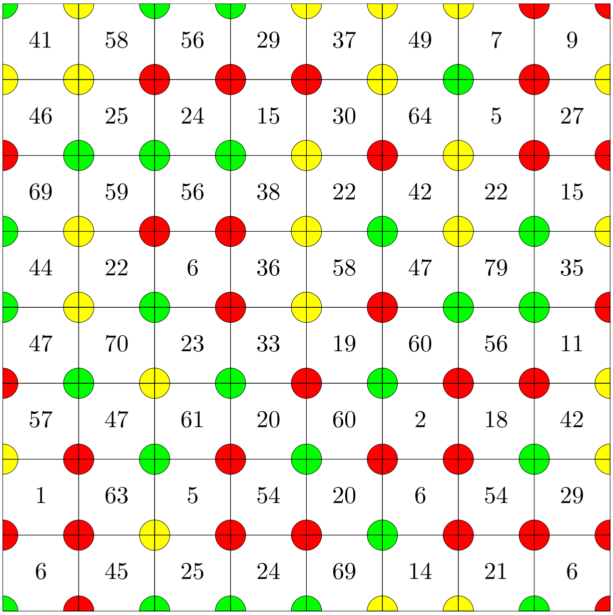


Figure 2.6: A tiling with the complete corner tile set over 3 colors.

tile are named after the compass headings north-east (NE), south-east (SE), south-west (SW), and north-west (NW). The colors of the corners are indicated by c_{NE}, c_{SE}, c_{SW} and c_{NW}.

A tile set is a finite set of tiles. The number of different colors used in the tile set is indicated by C. The C colors are represented by the integers $0, 1, \ldots, C-1$. All illustrations in this dissertation use the colors red, yellow, green, cyan and blue for respectively 0, 1, 2, 3 and 4.

A complete tile set contains a tile for every possible combination of four edge or corner colors. A complete set of Wang tiles or corner tiles over C colors therefore counts C^4 tiles. Figure 2.3 shows the complete Wang tile set over two colors, and figure 2.4 shows the complete corner tile set over two colors. A complete set of Wang tiles or corner tiles over 2, 3, 4, 5, 6, 7 and 8 colors consist of 16, 81, 256, 625, $1,296$, $2,401$ and $4,096$ tiles.

A tiling is constructed by placing the tiles next to each other such that adjoining edges or corners have matching colors. Each tile in the tile set can be used arbitrarily many times. The tiles are placed with their corners on the integer lattice points. By convention, the tile coordinates are the coordinates of the lower left corner of the tile. Figure 2.5 shows a tiling with the complete Wang tile set over three colors, and figure 2.6 shows a tiling with the complete corner tile set over three colors.

The horizontal edges and vertical edges of Wang tiles are independent. This allows Wang tile sets that use a different number of colors for horizontal and vertical edges. The number of colors used for horizontal edges is indicated by C_h, and the number of colors used for vertical edges is indicated by C_v. A complete Wang tile set over C_h horizontal colors and C_v vertical colors consist of $(C_h C_v)^2$ tiles. If C_h colors are used for horizontal edges and C_v colors for vertical edges, then the number of colors used in the tile set is $\max(C_h, C_v)$. In this dissertation, C_h is always a subset of C_v, or vice versa. However, not everyone follows this convention. For example, Cohen et al. [2003] use a tile set with red and green for horizontal edges and blue and yellow for vertical edges. Despite the fact that four different colors are used, this is a tile set over two colors.

2.8 Enumerating Wang Tile Sets and Corner Tile Sets

For efficiently manipulating Wang tiles and corner tiles, an enumeration of the tiles is needed. In this dissertation we use the following scheme.

Wang tiles are uniquely determined by their edge colors c_N, c_E, c_S and c_W. Wang tiles can thus be represented as the 4-digit base-C numbers $c_N c_E c_S c_W$, or as the decimal integers $0, 1, \ldots, C^4 - 1$. A base conversion switches between the corner colors and the tile index.

The tile index i of the Wang tile with edge colors c_N, c_E, c_S and c_W is given by

$$i = c_N C^3 + c_E C^2 + c_S C + c_W, \tag{2.1}$$

or, after factoring out powers of C using Horner's rule, by

$$i = ((c_N C + c_E)C + c_S)C + c_W. \tag{2.2}$$

This conversion of base only requires three integer multiplications and three integer additions.

The edge colors c_N, c_E, c_S and c_W of the Wang tile with tile index i are given by

$$\begin{aligned} c_N &= (i/C^3)\,\%\,C \\ c_E &= (i/C^2)\,\%\,C \\ c_S &= (i/C)\,\%\,C \\ c_W &= i\,\%\,C \end{aligned} \tag{2.3}$$

where % is the modulo division, and / is the integer division. This conversion of base can be implemented using only three modulo divisions and three integer divisions

$$\begin{aligned} c_W &\leftarrow i \,\%\, C \\ i &\leftarrow i/C \\ c_S &\leftarrow i \,\%\, C \\ i &\leftarrow i/C \\ c_E &\leftarrow i \,\%\, C \\ i &\leftarrow i/C \\ c_N &\leftarrow i \end{aligned} \tag{2.4}$$

where ← indicates assignment.

For corner tiles, we use a similar scheme. Corner tiles are uniquely determined by their corner colors c_{NE}, c_{SE}, c_{SW} and c_{NW}. Corner tiles can thus be represented as the 4-digit base-C numbers $c_{NE}c_{SE}c_{SW}c_{NW}$, or as the decimal integers $0, 1, \ldots, C^4 - 1$. A base conversion switches between the corner colors and the tile index.

The tile index i of the corner tile with corner colors c_{NE}, c_{SE}, c_{SW} and c_{NW} is given by

$$i = ((c_{NE}C + c_{SE})C + c_{SW})C + c_{NW}. \tag{2.5}$$

The corner colors c_{NE}, c_{SE}, c_{SW} and c_{NW} of the corner tile with tile index i are given by

$$\begin{aligned} c_{NE} &= (i/C^3) \,\%\, C \\ c_{SE} &= (i/C^2) \,\%\, C \\ c_{SW} &= (i/C) \,\%\, C \\ c_{NW} &= i \,\%\, C \end{aligned} \tag{2.6}$$

where % is the modulo division, and / is the integer division. This conversion of base can be implemented using only three modulo divisions and three integer divisions

$$\begin{aligned} c_{NW} &\leftarrow i \,\%\, C \\ i &\leftarrow i/C \\ c_{SW} &\leftarrow i \,\%\, C \\ i &\leftarrow i/C \\ c_{SE} &\leftarrow i \,\%\, C \\ i &\leftarrow i/C \\ c_{NE} &\leftarrow i \end{aligned} \tag{2.7}$$

where ← indicates assignment.

When the number of colors is a power of two, the base conversions can be implemented very efficiently using bitwise operators.

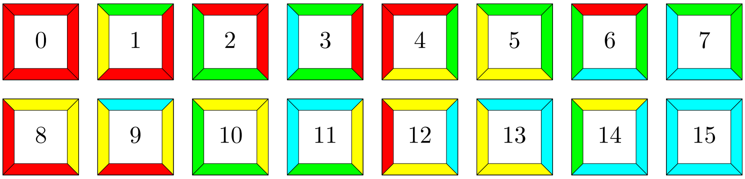


Figure 2.7: The Wang tile set equivalent to the corner tile set over 2 colors.

2.9 Corner Tiles as Wang Tiles

Corner tiles are closely related to Wang tiles. In fact, every corner tile set can be transformed into an equivalent Wang tile set. This is done by encoding any combination of two corner colors into an edge color. This operation squares the number of colors. Figure 2.7 shows the Wang tile set equivalent to the complete corner tile set over two colors, shown in figure 2.4. This is a Wang tile set of 16 tiles over 4 colors. We will use this technique in section 8.4 to construct an aperiodic set of Wang tiles from an aperiodic set of corner tiles.

In general, a Wang tile set cannot be transformed into an equivalent corner tile set, and a Wang tile set equivalent to a corner tile set is not subject to the corner problem. This shows that corner tiles are in some way more restrictive than Wang tiles.

2.10 Dominoes, Wang Cubes and Corner Cubes

Wang tiles and corner tiles easily generalize to arbitrary dimension. The one-dimensional and three-dimensional equivalents are especially useful.

In one dimension, Wang tiles and corner tiles are dominoes. These gaming pieces are well known and have been studied extensively in the field of recreational mathematics [Ball, 1926]. We will use dominoes in section 5.5 to solve the Wang tile packing problem.

In three dimensions, Wang tiles and corner tiles become Wang cubes and corner cubes. Wang cubes have received some attention in the field of discrete mathematics and in computer graphics. Culik and Kari [1995] showed that an aperiodic set of 21 Wang cubes exists. Lu and Ebert [2005] used Wang cubes for example-based volume illustrations. We will use corner cubes in section 4.6 to generate Poisson sphere distributions.

2.11 Conclusion

In this chapter we have introduced Wang tiles and corner tiles. We have discussed the history of Wang tiles and corner tiles, and we have introduced important concepts that will be used in later chapters.

Chapter 3

Tiling Algorithms for Wang Tiles and Corner Tiles

3.1 Introduction

After synthesizing a signal over a set of Wang tiles or corner tiles, arbitrary large quantities of that signal can be generated very efficiently by generating a tiling. Because periodicity in the signal is visually disturbing, applications in computer graphics require random or stochastic tilings, such as the ones shown in figures 2.5 and 2.6. Stochastic tilings are inherently non-periodic. The stronger mathematical guarantee of provable aperiodicity is not that useful in computer graphics. A mathematical proof of aperiodicity does not necessarily provide an algorithm for actually generating the aperiodic tiling, and even if it does, these algorithms are often very complex. Also, aperiodicity does not imply small scale non-periodicity. Aperiodic tilings are sometimes very structured. For these reasons, stochastic tilings are better suited for most applications in computer graphics.

Overview

This chapter is organized as follows. In section 3.2 we discuss scanline stochastic tiling algorithms, and in section 3.3 we discuss direct stochastic tiling algorithms, two classes of stochastic tiling algorithms. Section 3.4 discusses hash functions defined over the integer lattice, an essential ingredient of direct stochastic tiling algorithms. In section 3.5 we conclude.

3.2 Scanline Stochastic Tiling Algorithms

Scanline stochastic tiling algorithms are stochastic tiling algorithms that generate a tiling by placing tiles in scanline order. In this section, we discuss a scanline tiling algorithm for Wang tiles and a scanline stochastic tiling algorithm for corner tiles.

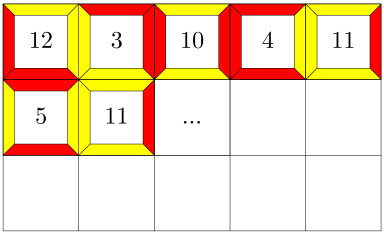


Figure 3.1: A scanline stochastic tiling algorithm for Wang tiles.

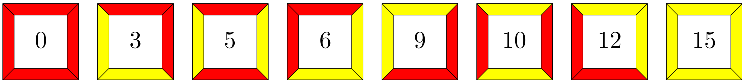


Figure 3.2: A compact Wang tile set over 2 colors.

3.2.1 A Scanline Stochastic Tiling Algorithm for Wang Tiles

In 2003, Cohen et al. presented a scanline stochastic tiling algorithm for Wang tiles.

The Wang tiles are placed in scanline order, from west to east, and from north to south. A random Wang tile is selected for the NW corner. The first row is completed by adding Wang tiles for which the color of the W edge corresponds to the color of the E edge of the Wang tile to the left. The leading Wang tile of each new row is selected so that its N edge matches the S edge of the Wang tile above. The row is completed by choosing Wang tiles for which the N and W edges match the S and E edges from the Wang tiles above and to the left. This is illustrated in figure 3.1.

To ensure a non-periodic tiling, the Wang tile set is constructed such that there are two Wang tiles for each combination of N and W edge colors. Each time a Wang tile has to be selected, the choice is made at random. A Wang tile set over C colors will contain $2C^2$ Wang tiles, since there are C^2 combinations of N and W edge colors. A Wang tile set obtained this way is called a compact Wang tile set, because it is significantly smaller than a complete Wang tile set. Figure 3.2 shows a compact Wang tile set over two colors.

Compact Wang tile sets are useful when the size of the Wang tile set should be minimized. A compact Wang tile set is quadratic in the number of colors, while a complete Wang tile set is quartic in the number of color. For 2, 3, 4, 5, 6, 7 and 8 colors, a compact Wang tile set counts 8, 18, 32, 50, 98, 128 Wang tiles while a complete Wang tile set consists of 16, 256, 625, $1,296$, $2,401$ and $4,096$ Wang tiles.

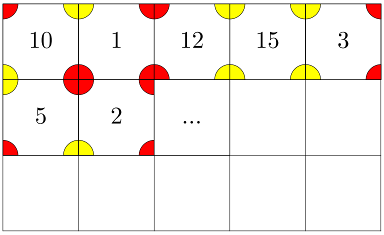


Figure 3.3: A scanline stochastic tiling algorithm for corner tiles.

3.2.2 A Scanline Stochastic Tiling Algorithm for Corner Tiles

In 2006, we presented a scanline stochastic tiling algorithm for corner tiles [Lagae and Dutré, 2006a]. The scanline stochastic tiling algorithm for corner tiles is very similar to the scanline stochastic tiling algorithm for Wang tiles.

The corner tiles are placed in scanline order, from west to east, and from north to south. A random corner tile is selected for the NW corner. The first row is completed by adding corner tiles for which the color of the NW corner and the color of the SW corner corresponds to the color of the NE corner and the color of the SE corner of the corner tile to the left. The leading corner tile of each new row is selected so that its NW and NE corners match the SW and SE corners of the corner tile above. The row is completed by choosing corner tiles for which the NE, NW and SW corners match the SE and SW corners from the corner tile above and the NE and SE corners from the corner tile to the left. This is illustrated in figure 3.3.

To ensure a non-periodic tiling, the corner tile set is constructed such that there are two corner tiles for each combination of NE, NW and SW corner colors. Each time a corner tile has to be selected, the choice is made at random. A corner tile set over C colors will contain $2C^3$ corner tiles, since there are C^3 combinations of NE, NW and SW corner colors. A corner tile set obtained this way is called a compact corner tile set, because it is significantly smaller than a complete corner tile set.

Compact corner tile sets are useful when the size of the corner tile set should be minimized. A compact corner tile set is cubic in the number of colors, while a complete corner tile set is quartic in the number of color. For 2, 3, 4, 5, 6, 7 and 8 colors, a compact corner tile set counts 16, 54, 128, 250, 432, 686 and $1,024$ corner tiles while a complete corner tile set consists of 16, 256, 625, $1,296$, $2,401$ and $4,096$ corner tiles.

Note that the complete corner tile set over two colors and the compact corner tile set over two colors are identical. Also note that compact Wang tile sets are smaller than compact corner tile sets.

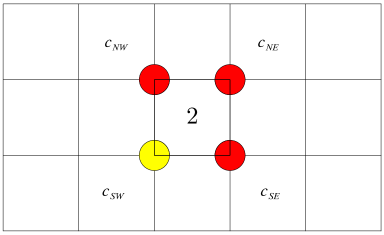


Figure 3.4: A direct stochastic tiling algorithm for corner tiles.

3.3 Direct Stochastic Tiling Algorithms

Several applications, such as tile-based texture mapping (see section 5.4), and the procedural object distribution texture basis functions (see section 7.6), require local evaluation of the tiling. In order to evaluate the tiling at a specific location, scanline stochastic tiling algorithms must construct and store the tiling up to that point. This is clearly not efficient. To address this problem, we propose direct stochastic tiling algorithms. Direct stochastic tiling algorithms are able to compute which tile is at a given location without explicitly constructing and storing the tiling up to that point.

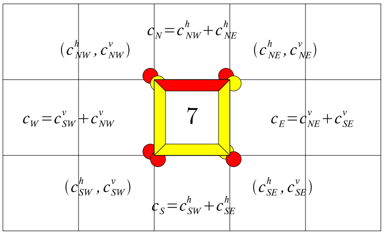
The direct stochastic tiling algorithms we present are based on hash functions defined over the integer lattice. These hash functions associate a random color with each lattice point. A tiling is obtained by transforming the colored lattice. The hash functions we use are efficient in time and space, and the transformation can be performed locally. This enables efficient direct stochastic tiling algorithms.

In this section, we discuss several direct stochastic tiling algorithms for Wang tiles and a direct stochastic tiling algorithm for corner tiles. For clarity, we will first discuss the direct stochastic tiling algorithm for corner tiles. The hash functions used in the direct stochastic tiling algorithms are discussed in section 3.4

3.3.1 A Direct Stochastic Tiling Algorithm for Corner Tiles

In 2006, we proposed a direct stochastic tiling algorithm for corner tiles [Lagae and Dutré, 2006a].

Corner tiles are placed with their corners on the integer lattice points. The coordinates of a corner tile are the coordinates of the integer lattice point corresponding to the lower left or SW corner. To generate a tiling with a corner tile set over C colors, the direct stochastic tiling algorithm for corner tiles uses a hash function h defined over the integer lattice. This hash function associates a random color $h(x, y) \in \{0, 1, \ldots, C-1\}$ with each location (x, y). The corner colors c_{NE}, c_{SE}, c_{SW} and c_{NW} of the corner tile at tile coordinates



$c_N = c^h_{NW} + c^h_{NE}$
(c^h_{NW}, c^v_{NW})
(c^h_{NE}, c^v_{NE})
$c_W = c^v_{SW} + c^v_{NW}$
7
$c_E = c^v_{NE} + c^v_{SE}$
(c^h_{SW}, c^v_{SW})
(c^h_{SE}, c^v_{SE})
$c_S = c^h_{SW} + c^h_{SE}$

Figure 3.5: A direct stochastic tiling algorithm for Wang tiles using two hash functions. The first hash function is used for the horizontal edges, and the second hash function is used for the vertical edges.

(x, y) are given by $h(x+1, y+1)$, $h(x+1, y)$, $h(x, y)$ and $h(x, y+1)$. The index of the corner tile can now be obtained using equation 2.5. The direct stochastic tiling algorithm for corner tiles is illustrated in figure 3.4.

Because the color of each corner is chosen at random, the direct stochastic tiling algorithm for corner tiles results in a complete corner tile set over C colors.

The direct stochastic tiling algorithm for corner tiles is very efficient. It requires only four hash function evaluations.

3.3.2 Direct Stochastic Tiling Algorithms for Wang Tiles

In this subsection, we discuss a direct stochastic tiling algorithm for Wang tiles using two hash functions, a direct stochastic tiling algorithm for compact sets of Wang tiles, and a direct stochastic tiling algorithm for Wang tiles using a hash function defined at the tile edges.

3.3.2.1 A Direct Stochastic Tiling Algorithm for Wang Tiles using Two Hash Functions

In 2005, we proposed a direct stochastic tiling algorithm for Wang tiles using two hash functions [Lagae and Dutré, 2005a].

Wang tiles are placed with their corners on the integer lattice points. The coordinates of a Wang tile are the coordinates of the integer lattice point corresponding to the lower left corner. To generate a tiling with a Wang tile set over C colors, the direct stochastic tiling algorithm for Wang tiles uses two hash functions h_h and h_v defined over the integer lattice. These hash functions associate a pair of random colors $(h_h(x, y), h_v(x, y)) \in \{0, 1, \ldots, C-1\}^2$ with each location (x, y). The hash function h_h is used to compute the color of the horizontal edges, and the hash function h_v is used to compute the color of the vertical edges. The edge colors of a Wang tile are computed as the sum modulo C of the random colors associated with the corners of the Wang

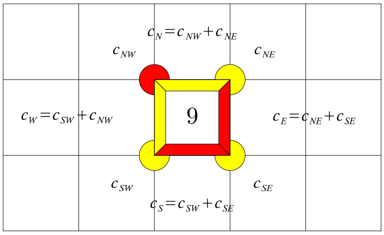


Figure 3.6: A direct stochastic tiling algorithm for compact sets of Wang tiles.

tile. If the pair of random colors associated with the NE, SE, SW and NW corner of the Wang tile at tile coordinates (x, y) are $\left(c^h_{NE}, c^v_{NE}\right)$, $\left(c^h_{SE}, c^v_{SE}\right)$, $\left(c^h_{SW}, c^v_{SW}\right)$ and $\left(c^h_{NW}, c^v_{NW}\right)$, then the edge colors c_N, c_E, c_S and c_W are given by $(c^h_{NW} + c^h_{NE})\%C$, $(c^v_{NE} + c^v_{SE})\%C$, $(c^h_{SE} + c^h_{SW})\%C$, $(c^c_{SW} + c^v_{NW})\%C$. The index of the Wang tile can now be obtained using equation 2.2. The direct stochastic tiling algorithm for Wang tiles using two hash functions is illustrated in figure 3.5.

Because the color of each edge is chosen at random, the direct stochastic tiling algorithm for Wang tiles using two hash functions results in a complete Wang tile set over C colors.

The direct stochastic tiling algorithm for Wang tiles is more expensive than the direct stochastic tiling algorithm for corner tiles. It requires eight hash function evaluations, four integer additions and four integer modulo divisions.

The direct stochastic tiling algorithm can easily be modified for tile sets over a different number of colors for horizontal edges C_h and vertical edges C_v by generating pairs of random colors $(h_h(x, y), h_v(x, y)) \in \{0, 1, \ldots, C_h - 1\} \times \{0, 1, \ldots, C_v - 1\}$ and performing edge computations for horizontal and vertical edges modulo C_h and C_v. This modified direct stochastic tiling algorithm for Wang tiles also results in a complete Wang tile set.

An algorithm similar in spirit was proposed concurrently by Wei [2004].

3.3.2.2 A Direct Stochastic Tiling Algorithm for Compact Sets of Wang Tiles

In 2005, we also proposed a direct stochastic tiling algorithm for compact sets of Wang tiles [Lagae and Dutré, 2005a].

The direct stochastic tiling algorithm for compact sets of Wang tiles is very similar to the direct stochastic tiling algorithm for Wang tiles using two hash functions. However, to generate a tiling with a compact Wang tile set over C colors only a single hash function h is used. This hash function associates a random color $h(x, y) \in \{0, 1, \ldots, C - 1\}$ with each location (x, y) The edge colors of a Wang tile are computed as the sum modulo C of the random colors associated with the corners of the Wang tile. If the random color associated

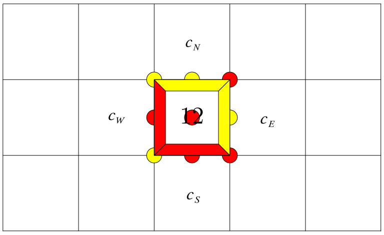


Figure 3.7: A direct stochastic tiling algorithm for Wang tiles using a hash function defined at the tile edges.

with the NE, SE, SW and NW corner of the Wang tile at tile coordinates (x, y) is c_{NE}, c_{SE}, c_{SW} and c_{NW}, then the edge colors c_N, c_E, c_S and c_W are given by $(c_{NW} + c_{NE})\%C$, $(c_{NE} + c_{SE})\%C$, $(c_{SE} + c_{SW})\%C$, $(c_{SW} + c_{NW})\%C$. The direct stochastic tiling algorithm for compact sets of Wang tiles is illustrated in figure 3.6.

This algorithm results in a compact set of C^3 Wang tiles over C colors. Note that, except for two colors, these compact Wang tile sets are different from the compact Wang tile sets produced by the scanline stochastic tiling algorithms for Wang tiles, that counted $2C^3$ tiles.

This algorithm was used recently by Kopf et al. [2006] for generating blue noise in real time (see section 6.14).

3.3.2.3 A Direct Stochastic Tiling Algorithm for Wang Tiles using a Hash Function Defined at the Tile Edges

The direct stochastic tiling algorithm for Wang tiles using a hash function defined at the tile edges is a more efficient variant of the direct stochastic tiling algorithm for Wang tiles using two hash functions.

Compared with the direct stochastic tiling algorithm for corner tiles, direct stochastic tiling algorithm for Wang tiles are more complicated. This is because the lattice defined by the colored corners of corner tiles and the lattice over which the hash function is defined are both square lattices. In contrast, the lattice defined by the colored edges of Wang tiles is a diamond lattice (a square lattice rotated 45 degrees). The key observation of the direct stochastic tiling algorithm for Wang tiles using a hash function defined at the tile edges is that a diamond lattice can be obtained by discarding points in a square lattice.

Wang tiles are placed with their corners on the integer lattice points. The coordinates of a Wang tile are the coordinates of the integer lattice point corresponding to the lower left. To generate a tiling with a Wang tile set over C colors, the direct stochastic tiling algorithm for Wang tiles uses a hash function h defined over a square lattice twice as dense as the integer lattice. The edge colors c_N, c_E, c_S and c_W of the Wang tile at tile coordinates (x, y) are given by $h(2x+1,2y+2)$, $h(2x+2,2y+1)$, $h(2x+1,2y)$ and $h(2x,2y+1)$. The remaining five random colors are ignored. The index of the Wang tile can now be obtained using equation 2.2. The direct stochastic tiling algorithm for Wang tiles using a hash function defined at the tile edges is illustrated in figure 3.7.

Because the color of each edge is chosen at random, the direct stochastic tiling algorithm for Wang tiles using a hash function defined at the tile edges results in a complete Wang tile set over C colors. In contrast with the direct stochastic tiling algorithm for Wang tiles using two hash functions, the direct stochastic tiling algorithm for Wang tiles using a hash function defined at the tile edges only requires four hash function evaluations.

3.4 Hash Functions

Hash functions defined over the integer lattice are an essential ingredient of the direct stochastic tiling algorithms discussed in section 3.3. Hash functions are also used extensively in procedural modeling and texturing [Perlin, 1985; Ebert et al., 2002]. In this section, we discuss traditional hash functions based on permutation tables and we propose long-period hash functions based on permutation tables.

3.4.1 Traditional Hash Functions Based on Permutation Tables

Hash functions used in procedural modeling and texturing are typically based on permutation tables. A permutation table P of size N contains a random permutation of the integers $\{0,1,\ldots,N-1\}$. The permutation table is zero-based, the first element is $P[0]$.

A random permutation of the elements $\{0,1,\ldots,N-1\}$ can be generated by starting with the permutation $\{0,1,\ldots,N-1\}$, and then exchanging the ith element with an element randomly selected from the first i elements, for $i \in 0,1,\ldots,N-2$. Note that, for i equal to $N-1$, this operation has no effect.

A one-dimensional hash function is defined as

$$h(x) = P[x\%N], \tag{3.1}$$

where x is an integer, $P[i]$ is the $(i+1)$th element of P, and % is the modulo division. This hash function is a periodic function with period N. The range of this hash function is N.

A two-dimensional hash function can be defined using two permutation tables P_x and P_y of size N

$$h(x,y) = (P_x[x\%N] + P_y[y\%N])\%N, \tag{3.2}$$

where x and y are integers. However, in order to avoid the storage of two permutation tables, the same permutation table is generally used twice

$$h(x, y) = P[(P[x\%N] + y)\%N]. \tag{3.3}$$

This hash function is a periodic function with period (N, N). The range of this hash function is N.

A three-dimensional hash function is defined similarly as

$$h(x, y, z) = P[(P[(P[x\%N] + y)\%N]) + z)\%N], \tag{3.4}$$

where x, y and z are integers. This hash function is a periodic function with period (N, N, N). The range of this hash function is N.

This family of hash functions is used extensively in procedural modeling and texturing, and we also use them in our direct stochastic tiling algorithms. The hash functions are easy to implement and efficient to evaluate.

3.4.2 Long-Period Hash Functions Based on Permutation Tables

The period of hash functions based on permutation tables is short. This causes unwanted repetition artifacts in procedural textures and tilings. Increasing the period is easy but also expensive, because the length of the period is equal to the size of the permutation table.

In 2006, we proposed long-period hash functions based on permutation tables [Lagae and Dutré, 2006d]. The key observation for constructing long-period hash functions is that hash functions based on permutation tables are periodic functions, and that the addition of periodic functions yields a new periodic function with a larger period.

In this subsection we define long-period hash functions, we study the period and range, distribution and efficiency of long-period hash functions, and we formulate guidelines for designing long-period hash functions.

3.4.2.1 Definition

A one-dimensional long-period hash function is defined as

$$h(x) = \left(\sum_{i=0}^{M-1} P_i[x\%N_i]\right)\%N_j, \tag{3.5}$$

where x is an integer, $P_0, P_1, \ldots, P_{M-1}$ are M permutation tables with size $N_0, N_1, \ldots, N_{M-1}$, and N_j is one of these sizes.

A two-dimensional long-period hash function is defined as

$$h(x, y) = \left(\sum_{i=0}^{M-1} P_i[(P_i[x\%N_i] + y)\%N_i]\right)\%N_j, \tag{3.6}$$

where x and y are integers

A three-dimensional long-period hash function is defined similarly as

$$h(x,y,z) = \left(\sum_{i=0}^{M-1} P_i[(P_i[(P_i[x\%N_i] + y)\%N_i] + z)\%N_i] \right) \%N_j. \tag{3.7}$$

where x, y and z are integers. These long-period hash functions are also called combined hash functions.

The period of a combined hash function is the least common multiple of the periods of the combining hash functions. In order to maximize the period of the combined hash function, the periods of the combining hash functions should be relatively prime. The range of the combined hash function is determined by the final modulo divisor N_j, which is one of $N_0, N_1, \ldots, N_{M-1}$. Note that, in contrast with traditional hash functions, the range and period of the combined hash functions are different.

A similar technique was used in 1988 by L'Ecuyer to construct a long-period pseudo-random number generator by combining several shorter-period linear congruential generators. However, with the advent of recent pseudo-random number generators [Matsumoto and Nishimura, 1998], this technique has largely become obsolete in its original context.

3.4.2.2 Distribution

Traditional hash functions produce uniformly distributed values over the integer lattice, and most applications of these hash functions rely on this property. The following theorem shows that combined hash functions will also produce uniformly distributed values.

Theorem. *If $X_0, X_1, \ldots, X_{N-1}$ are N independent discrete random variables, such that X_0 is uniform between 0 and $d-1$, where d is a positive integer, then*

$$X = \left(\sum_{i=0}^{N-1} X_i \right) \%d \tag{3.8}$$

follows a discrete uniform probability law between 0 and $d-1$.

Note that there are no requirements on the distribution of the random variables $X_1, X_2, \ldots, X_{N-1}$. This theorem was first hinted at by Wichmann and Hill [1982], and later proved by L'Ecuyer [1988].

For long-period hash functions, all combining hash functions are uniformly distributed. Therefore, the final modulo divisor N_j can be any one of the sizes of the permutation tables of the combining hash functions $N_0, N_1, \ldots, N_{M-1}$. However, some care must be taken in selecting the appropriate permutation table sizes. Suppose a combined hash function is built from a small permutation table of size N_s and a large permutation table of size N_l, with $N_s \ll N_l$. The period of the combined hash function is $N_s N_l$. However, if the final modulo divisor is N_l, then the period will contain N_s almost identical parts. This is because the range of the hash function using the permutation table of size N_s is very small compared to the range of the hash function using the permutation table of size N_l. If the final modulo divisor is N_s, this will not be the case. Therefore, the sizes of the permutation tables should not differ too much.

3.4.2.3 Efficiency

The time needed to evaluate a combined hash function is roughly proportional to the number of combining hash functions. This does not mean that an application that uses a long-period hash function consisting of N combining hash functions will be N times slower than the same application using a traditional hash function. In most applications, the evaluation of the hash function is only a small part of the total computation time. Also note that combined hash functions typically have a smaller memory footprint, which improves the cache efficiency of lookups in the permutation tables.

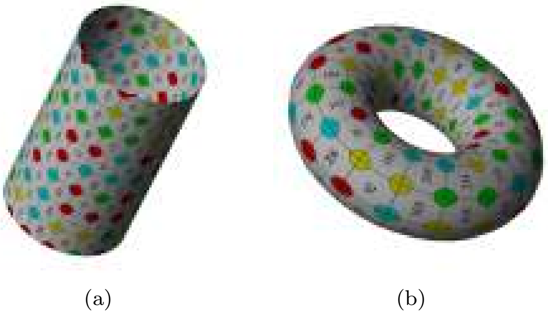
3.4.2.4 Design

The design of a combined hash function is determined by several factors: the required range of the hash function, the period of the hash function, the memory footprint of the combined permutation tables, and the time needed to evaluate the hash function. We recommend the strategy outlined below to design a long-period hash function.

First, determine the required range of the hash function. This fixes the final modulo divisor and thus the size of one permutation table. If the range of the hash function is too small, or if the range should be adjustable, then use a multiple of the range of the hash function and apply an additional final modulo divisor.

Next, choose a number of permutation table sizes for the rest of the combining hash function. The sizes should not differ too much, in order to ensure a uniform distribution. The sizes should also be relatively prime in order to maximize the period. An easy way to choose the permutation table sizes is to take the primes closest to the range of the hash function. If the primes are not a factor of the range, then the period of the combined hash function will be the product of the permutation table sizes.

The number of permutation tables will determine the time needed to evaluate the hash function, and the joint size of the permutation tables will determine the memory footprint of the hash function. Period length can be traded for evaluation time and memory footprint.



(a) (b)

Figure 3.8: Tilings of surfaces with (a) cylindrical and (b) toroidal topology.

3.4.3 Hash Functions for Direct Stochastic Tiling Algorithms

Direct stochastic tiling algorithms use a hash function to associate a random color with each integer lattice point. This colored lattice is then transformed to a tiling. Properties of the direct stochastic tiling algorithms such as efficiency and periodicity are inherited from the hash function.

Traditional hash functions based on permutation tables are simple and efficient. They are often used when speed is crucial or when a long period is less important. Permutation table sizes of 256 are common.

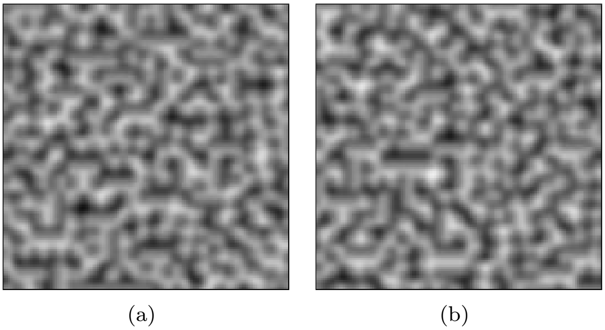
When speed is less important or when a long period is crucial, long-period hash functions based on permutation tables can be used. A long-period hash function for direct stochastic tiling algorithms can be designed as follows. The number of colors used in most tilings is 2, 3, 4, 6, or 8. The range of the hash function is therefore set to 24, the least common multiple of these number of colors. By applying an additional modulo divisor, the range of the hash function can be adjusted to 2, 3, 4, 6 and 8. For the sizes of the other permutation tables, the primes 17, 19, 23, 29, 31 and 37 are selected. The period of the combined hash function is $17 \times 19 \times 23 \times 24 \times 29 \times 31 \times 37 = 5,930,659,848$. Note that this is more than the 32-bit integer range. The size of the combined permutation table is only $17 + 19 + 23 + 24 + 29 + 31 + 37 = 180$.

Note that periodicity is not necessarily a bad thing. Periodicity allows to correctly handle boundary conditions when tiling surfaces with cylindrical or toroidal topology. This is illustrated in figure 3.8.

3.4.4 Hash Functions for Procedural Texturing

The most famous texture basis function is Perlin's noise function [Perlin, 1985, 2002]. Long-period hash functions based on permutation tables can be used to robustly implement this texture basis function.

A long-period hash function for Perlin's noise function can be designed as



(a) (b)

Figure 3.9: Perlin noise generated (a) with the traditional hash function and (b) with the long-period hash function. The two images have the same appearance.

follows. Perlin's noise function uses the lower 4 bits of the hash function (16 values) to choose amongst one of 12 vectors at each integer lattice point. The required range of the hash function is therefore 16. For the sizes of the other permutation tables the primes 11, 13, 17 and 19 are selected. The period of the combined hash function is $11 \times 13 \times 16 \times 17 \times 19 = 739,024$. The size of the combined permutation table is $11 + 13 + 16 + 17 + 19 = 76$. Compared to Perlin's implementation, the period is increased with a factor of almost $3,000$, the memory footprint is decreased with a factor of almost 3.5, and the evaluation time is increased with a factor of about 5. The total evaluation time of the modified noise function is increased with a factor of about 2.5. Figure 3.9 shows the original and the modified implementation. As expected, the two images have the same appearance. The long-period hash function does not introduce unwanted artifacts and preserves the typical look of Perlin's noise function.

3.5 Conclusion

Tile-based methods in computer graphics require efficient algorithms for generating stochastic tilings. In this chapter we have presented scanline stochastic tiling algorithms and direct stochastic tiling algorithms for Wang tiles and corner tiles. We have also studied the hash functions on which the direct stochastic tiling algorithms are based.

Direct stochastic tiling algorithms are more elegant and at least as efficient as scanline stochastic tiling algorithms. We therefore recommend to use direct stochastic tiling algorithms if possible, even for applications that do not require local evaluation of the tiling.

Stochastic tiling algorithms for corner tiles are usually more efficient and more elegant than stochastic tiling algorithms for Wang tiles. We therefore recommend to use corner tiles if possible.

One stochastic tiling algorithm that is still missing in this chapter is a direct stochastic tiling algorithm for compact sets of corner tiles. Although such an algorithm would be useful, we have not yet succeeded in developing one.

Direct stochastic tiling algorithms are based on hash functions. The period of the tiling is inherited from the period of the hash functions. Hash functions based on permutation tables allow to trade storage space and evaluation time for period length, and allow to construct hash functions with long periods. These hash functions are not only useful in tiling algorithms but also in procedural modeling and texturing techniques.

Chapter 4

Tile-Based Methods for Generating Poisson Disk Distributions

4.1 Introduction

In computer graphics, Wang tiles and corner tiles are used to facilitate the synthesis of complex signals. Poisson disk distributions are complex point distributions that are difficult to generate in real time. This chapter presents several methods for constructing Poisson disk distributions over a set of Wang tiles or corner tiles. With a single set of precomputed tiles, high-quality Poisson disk distributions of arbitrary size can be generated very efficient, simply by producing a stochastic tiling.

Overview

This chapter is organized as follows. In section 4.2 we introduce Poisson disk distributions. Sections 4.3, 4.4 and 4.5 present edge-based Poisson disk tiles, template Poisson disk tiles and corner-based Poisson disk tiles, three tile-based methods for generating Poisson disk distributions. In section 4.6 we investigate Poisson sphere distributions, the three-dimensional equivalent of Poisson disk distributions. Section 4.7 discusses nonuniform Poisson disk distributions. In section 4.8 we conclude.

This chapter only discusses methods for constructing Poisson disk distributions over a set of Wang tiles or corner tiles. The Poisson disk distributions generated by these methods are analyzed in detail in chapter 6, applications of Poisson disk distributions are discussed in chapter 7, and efficient tiling algorithms for Wang tiles and corner tiles are presented in chapter 3.

4.2 Poisson Disk Distributions

In this section we define Poisson disk distributions, we sketch the history and background of Poisson disk distributions, we propose an intuitive radius specification scheme for Poisson disk distributions, and we introduce methods for generating Poisson disk distributions.

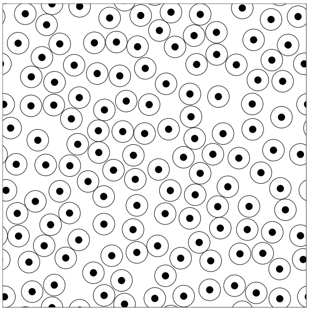


Figure 4.1: A Poisson disk distribution.

4.2.1 Definition

A Poisson distribution or random distribution is the simplest random point distribution. A Poisson distribution is a point distribution in which the points and the coordinates of the points have no relationship to each other. A Poisson distribution is obtained by generating uniformly distributed random numbers and using them as coordinates for the points. This distribution is called a Poisson distribution because the number of points in an area is distributed according to a Poisson probability distribution with mean equal to the area multiplied with the density.

A Poisson disk distribution is a two-dimensional Poisson distribution in which all points are separated from each other by a minimum distance. Half that distance is called the radius r of the distribution. If a disk of that radius is placed at each point, then no two disks overlap. This explains why this distribution is called a Poisson disk distribution. Figure 4.1 shows an example of a Poisson disk distribution.

4.2.2 History and Background

Poisson disk distributions were introduced in the field of computer graphics to solve the aliasing problem. Aliasing is a major source of artifacts in digitally synthesized images. This problem was first identified by Crow [1977]. Dippé and Wold [1985], Cook [1986] and Mitchell [1987] introduced nonuniform sampling to turn regular aliasing patterns into featureless noise, which is perceptually less objectable. The Poisson disk distribution was identified as one of the best sampling patterns. This work was based on studies by Yellot [1982, 1983], who found that the photoreceptors in the retina of the eye are distributed according to a Poisson disk distribution, an indication that this sampling pattern is effective for imaging.

Poisson disk distributions are traditionally generated using an expensive dart throwing algorithm [Cook, 1986]. Fast methods that generate approximate

Poisson disk distributions have been suggested by various authors [Dippé and Wold, 1985; Mitchell, 1987, 1991; Klassen, 2000]. The algorithm mostly used nowadays is due to McCool and Fiume [1992]. It generalizes over the dart throwing approach, and uses Lloyd's relaxation method [Lloyd, 1982] to optimize the generated distribution.

Because Poisson disk distributions are expensive to generate, Dippé and Wold [1985] suggested already in 1985 to replicate a precomputed tile with Poisson disk distributed points across the plane. Since then, several tile-based methods were proposed. Most of them use Wang tiles. The first tile based method, an extension of the dart throwing algorithm, was presented by Shade et al. [2000]. Hiller et al. [2001] used Lloyd's relaxation algorithm to construct a Poisson disk distribution over a set of Wang tiles. This method was later adopted by Cohen et al. [2003]. Ostromoukhov et al. [2004] presented an interesting technique to generate a distribution with blue noise properties over a given density, based on the Penrose tiles and Lloyd's relaxation method. Kopf et al. [2006] presented a method to generate Poisson disk distributions over a given density in real time, based on recursive Wang tiles that contain self-similar and progressive Poisson distributions.

Recently, Jones [2006] and Dunbar and Humphreys [2006] presented efficient implementations of the dart throwing algorithm.

Tools to analyze the spectral properties of point sets were introduced by Ulichney [1987], in the context of dithering.

4.2.3 Radius Specification

The radius of a Poisson disk distribution determines how well the points are distributed, and is therefore a measure of the quality of the Poisson disk distribution. The radius is typically expressed as an absolute number. However, this is not practical because the radius is dependent on the size of the domain of the point distribution and on the number of points in the distribution.

In 2005, we proposed a more intuitive radius specification scheme [Lagae and Dutré, 2005a]. Instead of using the absolute radius r, the radius is expressed as a relative radius ρ. The relative radius ρ is a fraction of the maximum radius r_{max} that can be achieved.

The densest packing of disks in the plane is a hexagonal lattice. Therefore, the point configuration with maximum disk radius r_{max} is a hexagonal lattice. The packing density η of a hexagonal lattice is [Steinhaus, 1999]

$$\eta = \frac{\pi}{2\sqrt{3}} \approx 0.9069. \tag{4.1}$$

The packing density is defined as the fraction of the area filled by the disks.

The maximum disk area of a Poisson disk distribution counting N points over the toroidal unit square is therefore η/N. The maximum possible disk radius r_{max} of this Poisson disk distribution is thus given by

$$r_{max} = \sqrt{\frac{1}{2\sqrt{3}N}}. \tag{4.2}$$

The Poisson disk radius r of a given point distribution is specified as a fraction ρ of the maximum disk radius

$$r = \rho\, r_{max}, \tag{4.3}$$

with $\rho \in [0, 1]$.

In contrast with the absolute radius, the relative radius is independent of the number of points and the size of the domain of the point distribution. The relative radius is therefore a good measure of how well the points are distributed.

The relative radius of a Poisson distribution is 0, because a Poisson distribution does not enforce a minimum distance between points. The relative radius of a hexagonal lattice is 1, because a hexagonal lattice is the densest packing of disks in the plane. The relative radius of a Poisson disk distribution should be relatively large, in order to ensure a good distribution of points, but not too large, because point distributions with a very large relative radius are too close to the hexagonal lattice, and are therefore too regular. Practice shows that the radius of most Poisson disk distribution is somewhere in between 0.65 and 0.85.

4.2.4 Generation

Poisson disk distributions are traditionally generated with dart throwing, relaxation dart throwing or Lloyd's relaxation method.

4.2.4.1 Dart Throwing

The dart throwing algorithm of Cook [1986] was the first algorithm to generate Poisson disk distributions. The algorithm generates points distributed according to a Poisson distribution, and rejects points that do not satisfy the minimum separation with already generated points. This process continues until no more points can be added. To correctly handle boundary conditions, the distributions generated by the dart throwing algorithm are usually toroidal.

This algorithm is expensive, and difficult to control. Instead of specifying the number of points, the radius of the distribution has to be provided, the final number of points in the distribution is difficult to predict, and if the process is stopped too soon, the density of the points is not uniform. The dart throwing algorithm is discussed in detail in section 6.3.

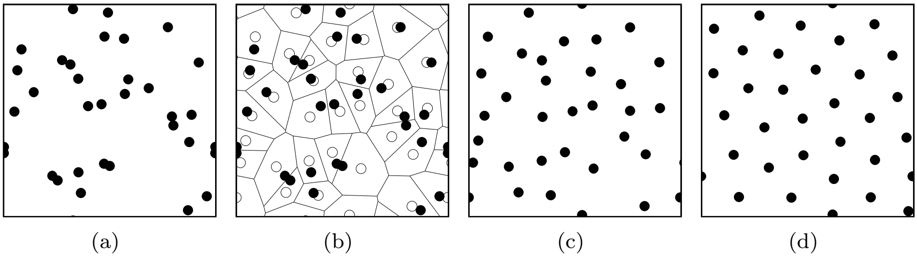


Figure 4.2: Lloyd's relaxation method. (a) The initial point set. (b) The Voronoi diagram of the initial point set. The centroids of the Voronoi cells are indicated by circles. (c) The points are moved to the centroid of their Voronoi cell. (d) This process is iterated. Note the increase in radius of the point set.

4.2.4.2 Relaxation Dart Throwing

McCool and Fiume [1992] proposed an improved version of the dart throwing algorithm, which we call relaxation dart throwing. Points are placed with a large radius initially, and once no more space has been found for a large number of attempts, the radius is reduced by some fraction.

This algorithm has several advantages compared to dart throwing. It is faster, it allows to specify the final size of the distributions rather than the radius, and termination is guaranteed. The relaxation dart throwing algorithm is discussed in detail in section 6.4.

4.2.4.3 Lloyd's Relaxation Scheme

After a Poisson disk distribution is generated, McCool and Fiume [1992] apply Lloyd's relaxation method [Lloyd, 1982] to optimize the radius of the Poisson disk distribution. Lloyd's relaxation method is an iterative process. In each iteration, the Voronoi diagram of the point set is computed, and each point is moved to the centroid of its Voronoi cell. This process is illustrated in figure 4.2. Lloyd's relaxation method is discussed in detail in section 6.5.

4.3 Edge-Based Poisson Disk Tiles

In 2005, we presented edge-based Poisson disk tiles, a method for constructing a Poisson disk distribution over a set of Wang tiles [Lagae and Dutré, 2005a].

Constructing a Poisson disk distribution over a set of Wang tiles is challenging. The difficulty is to generate a Poisson disk distribution in each tile of the tile set, such that every tiling results in a valid Poisson disk distribution. The minimum distance criterion of a Poisson disk distribution imposes severe constraints on the point distributions in the Wang tiles.

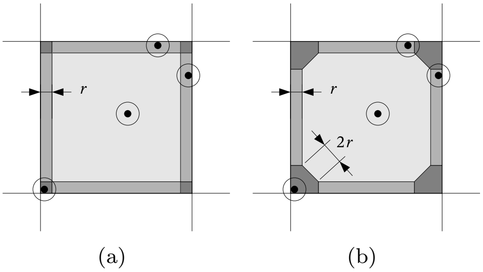


Figure 4.3: The Poisson disk tile regions and the modified Poisson disk tile regions. (a) The Poisson disk radius determines corner regions, edge regions and an interior region. (b) The corner regions are modified such that the distance between regions of the same type is at least $2r$.

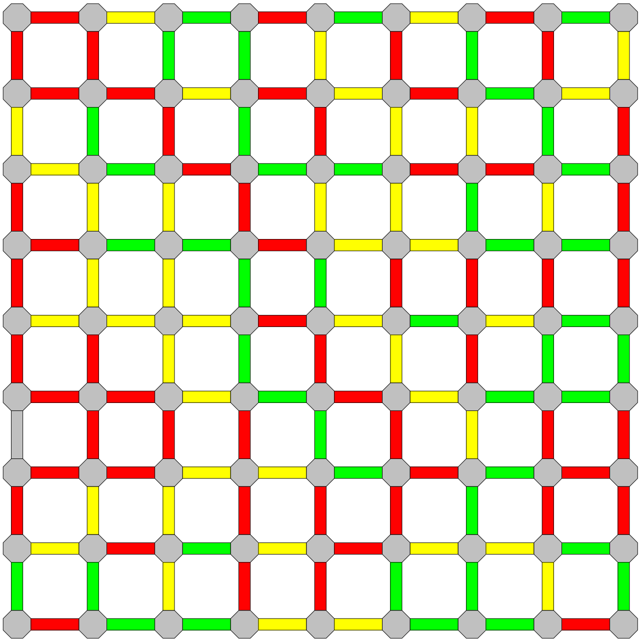


Figure 4.4: A tiling obtained by combining the modified Poisson disk tile regions with the complete Wang tile set over 3 colors. This tiling was generated from the tiling shown in figure 2.5.

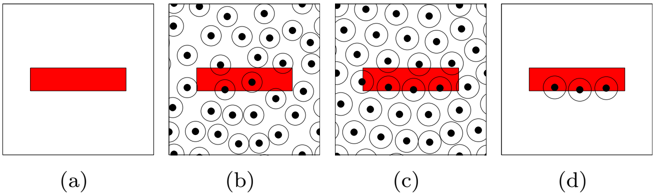


Figure 4.5: Construction of a Poisson disk distribution over an edge tile of an edge-based Poisson disk tile set. (a) The edge tile. (b) A toroidal Poisson disk distribution is generated. (c) The Poisson disk distribution is optimized using Lloyd's relaxation scheme. (d) The edge tile is cut out of the Poisson disk distribution.

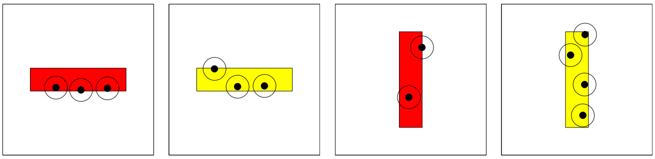


Figure 4.6: Poisson disk distributions constructed over edge tiles of an edge-based Poisson disk tile set.

A point in a tile closer to a corner than the Poisson disk radius affects points in three neighboring tiles. A point in a tile closer to an edge than the Poisson disk radius affects points in one neighboring tile. A point in a tile, further away from the tile boundary than the Poisson disk radius does not affect points in neighboring tiles. The regions obtained this way are called the Poisson disk tile regions. The Poisson disk radius determines corner regions, edge regions and an interior region. This is illustrated in figure 4.3(a).

To minimize the constraints between the different regions, the corner regions are enlarged such that the distance between edge regions is twice the Poisson disk radius. The regions obtained this way are called the modified Poisson disk tile regions. Within a single tile, points in edge regions now only affect points in corner regions, and not in other edge regions. This is illustrated in figure 4.3(b).

By combining the modified Poisson disk tile regions with the complete Wang tile set over C colors, a new tiling is obtained. This is illustrated in figure 4.4. This tiling uses three different kinds of tiles. Horizontal and vertical edge tiles, corner tiles[1] and interior tiles. Edge tiles correspond to the union of two modified edge regions. There are C horizontal and C vertical edge tiles, one for

[1] The term *corner tile* is used to indicate both a square tile with colored corners and the union of four Poisson disk regions. The context should make clear which meaning is intended.

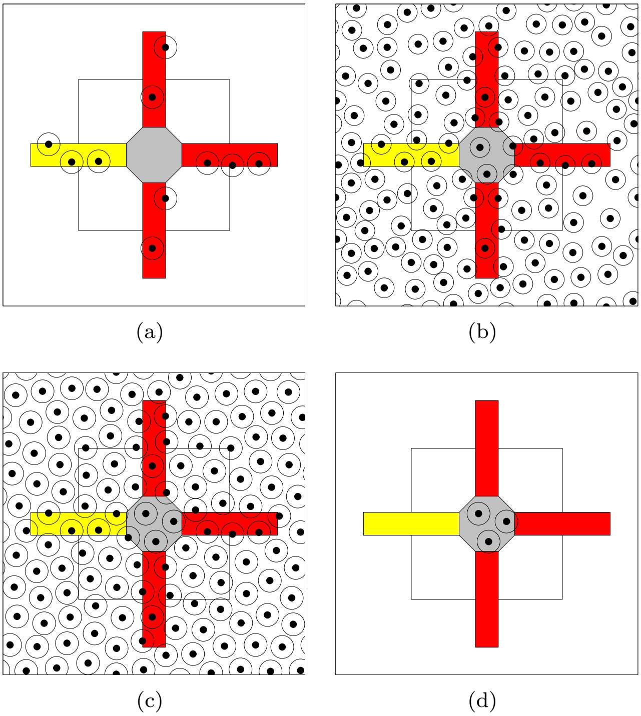


Figure 4.7: Construction of a Poisson disk distribution over a corner tile of an edge-based Poisson disk tile set. (a) The corner tile is assembled with the corresponding edge tiles. (b) A toroidal Poisson disk distribution is generated. (c) The Poisson disk distribution is optimized using Lloyd's relaxation scheme. (d) The corner tile is cut out of the Poisson disk distribution.

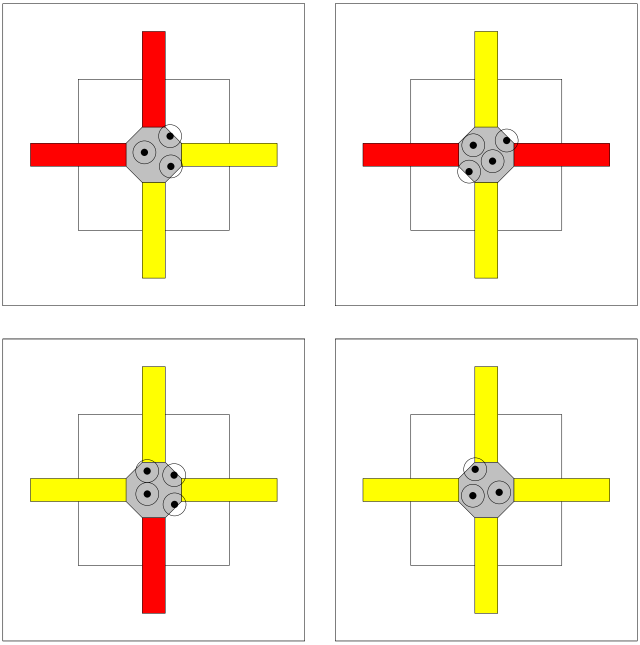


Figure 4.8: Poisson disk distributions constructed over corner tiles of an edge-based Poisson disk tile set.

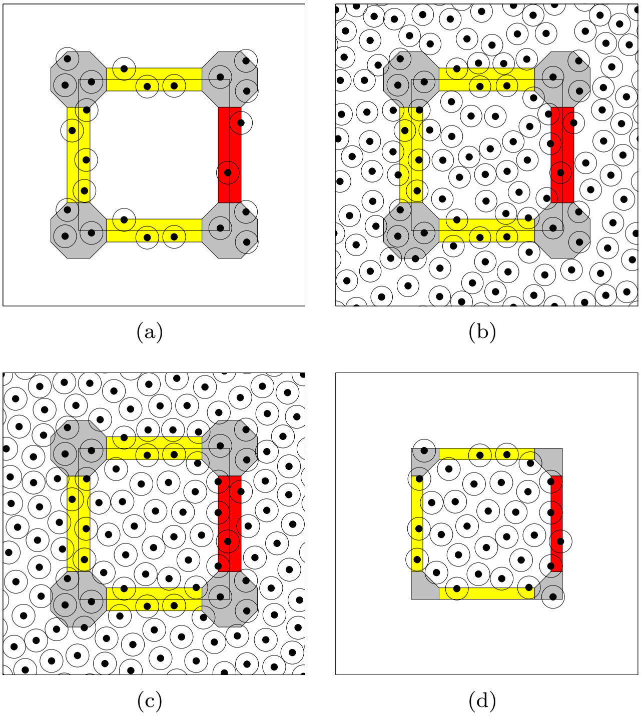


Figure 4.9: Construction of a Poisson disk distribution over a tile of an edge-based Poisson disk tile set. (a) The tile interior is assembled with the corresponding corner tiles and edge tiles. (b) A toroidal Poisson disk distribution is generated. (c) The Poisson disk distribution is optimized using Lloyd's relaxation scheme. (d) The tile is cut out of the Poisson disk distribution.

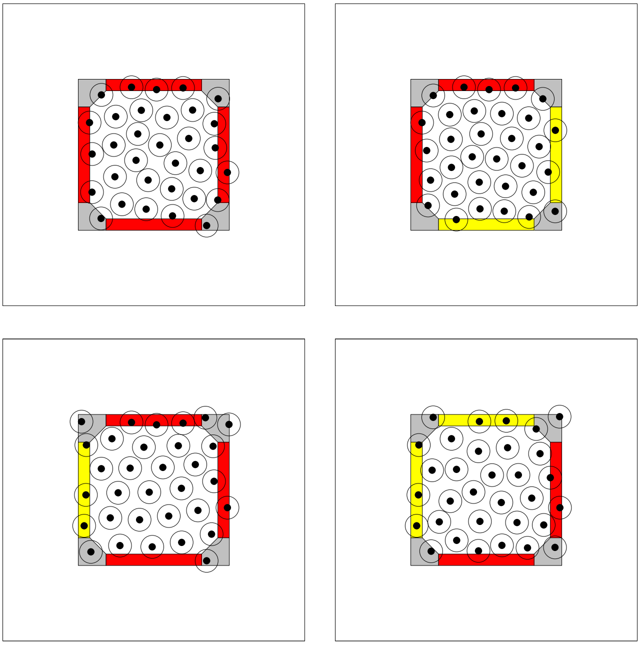


Figure 4.10: Poisson disk distributions constructed over tiles of an edge-based Poisson disk tile set.

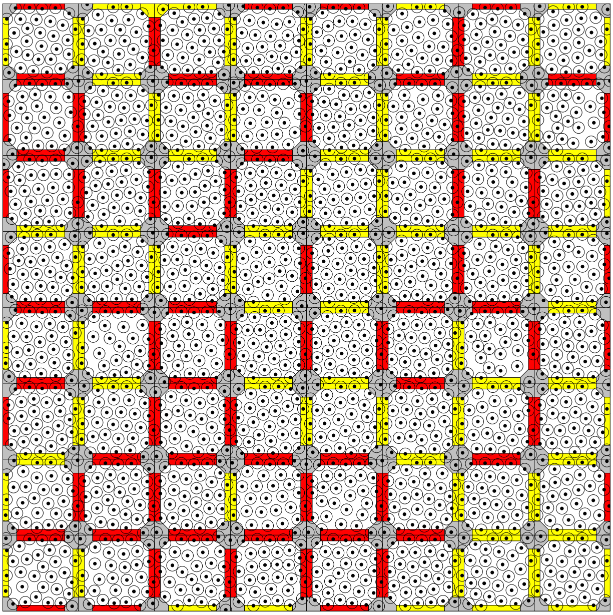


Figure 4.11: A tiling with a set of edge-based Poisson disk tiles.

each color. Corner tiles correspond to the union of four modified corner regions. There are C^4 corner tiles, one for each combination of four colors. Interior tiles correspond to the modified interior regions. There are C^{12} interior tiles, one for each combination of four corner tiles and four edge tiles. Note that there are C^{12} rather than C^4 interior tiles. This is because each corner tile encodes its four incident edges.

To construct a Poisson disk distribution over a set of Wang tiles, the number of colors of the Wang tile set C, the number of points per tile N, and the relative Poisson disk radius ρ are chosen. The absolute Poisson disk radius determines the size of the modified Poisson disk regions.

First, a Poisson disk distribution is constructed over the edge tiles. This is illustrated in figure 4.5. For each edge tile, a toroidal Poisson disk distribution of N points is generated using dart throwing or relaxation dart throwing (see figure 4.5(b)), optionally followed by Lloyd's relaxation method (see figure 4.5(c)). The edge tile is then cut out of the Poisson disk distribution (see figure 4.5(d)). If the desired Poisson disk radius is not reached, this process is repeated. Figure 4.6 shows Poisson disk distributions constructed over edge tiles.

Next, a Poisson disk distribution is constructed over the corner tiles. This is illustrated in figure 4.7. Each corner tile is assembled with the corresponding edge tiles (see figure 4.7(a)). A toroidal Poisson disk distribution is generated

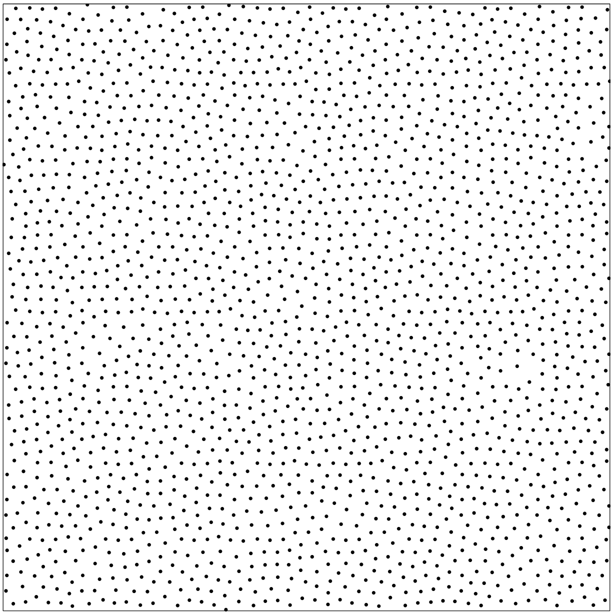


Figure 4.12: A Poisson disk distribution generated with a set of edge-based Poisson disk tiles. This Poisson disk distribution was generated from the tiling shown in figure 4.11.

using dart throwing or relaxation dart throwing (see figure 4.7(b)), optionally followed by Lloyd's relaxation method (see figure 4.7(c)). The corner tile is then cut out of the Poisson disk distribution (see figure 4.7(d)). If the desired Poisson disk radius is not reached, this process is repeated. No new points are added to the edge tiles. During relaxation, the points in the edge tiles are fixed, and other points are prohibited to enter the edge tiles. This is done by clipping the displacement vectors of points that are about to enter the edge tiles. Figure 4.8 shows Poisson disk distributions constructed over corner tiles.

Finally, a Poisson disk distribution is constructed over the interior tiles. This is illustrated in figure 4.9. Each interior tile is assembled with the corresponding edge tiles and corner tiles (see figure 4.9(a)). A toroidal Poisson disk distribution that brings the number of points inside the tile to N is generated using dart throwing or relaxation dart throwing (see figure 4.9(b)), optionally followed by Lloyd's relaxation method (see figure 4.9(c)). The tile is then cut out of the Poisson disk distribution (see figure 4.9(d)). If the desired Poisson disk radius is not reached, this process is repeated. No new points are added to the corner tiles and the edge tiles. During relaxation, the points in the edge tiles and the corner are fixed, and other points are prohibited to enter the edge tiles and the corner tiles. Figure 4.10 shows Poisson disk distributions constructed over tiles.

An edge-based Poisson disk tile set based on a complete Wang tile set over C colors consists of C^{12} tiles. The only practical choice for C is 2, which results in an edge-based Poisson disk tile set counting $4,096$ tiles. The reason that an edge-based Poisson disk tile set consists of C^{12} tiles rather than C^4 tiles, is that edge-based Poisson disk tiles keep into account the color of all edges incident on the tile corners, which is needed to fix the point configuration around corners. This is a concrete instance of the corner problem (see section 2.6).

Figure 4.11 shows a tiling with a set of edge-based Poisson disk tiles, and figure 4.12 shows the resulting Poisson disk distribution. Although edge-based Poisson disk tiles are not Wang tiles, tiling algorithms for Wang tiles can still be used, because there is a straightforward mapping between an edge-based Poisson disk tiling and the Wang tiling it is constructed from.

The time needed to generate a Poisson disk tile set ranges from several minutes to several hours, depending on N and ρ. However, the construction of a tile set has to be done only once. With a single set of tiles, an infinite number of Poisson disk distributions can be generated.

The properties of the Poisson disk distributions generated with edge-based Poisson disk tiles are discussed in detail in section 6.9.

4.4 Template Poisson Disk Tiles

In 2005, we presented template Poisson disk tiles [Lagae and Dutré, 2005b], a simple approach for generating tiled Poisson disk distributions. In contrast

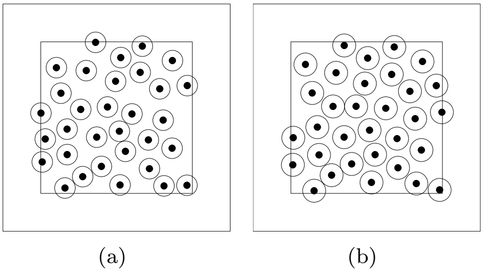


Figure 4.13: Construction of a master tile of a template Poisson disk tile set. (a) A toroidal Poisson disk distribution is generated. (b) The Poisson disk distribution is optimized using Lloyd's relaxation scheme.

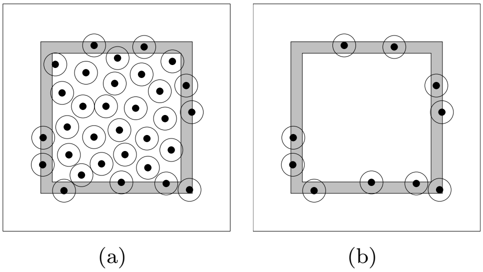


Figure 4.14: Construction of a tile template of a template Poisson disk tile set. (a) The master tile. (b) The tile template, constructed by discarding all points further from the tile border than the Poisson disk radius.

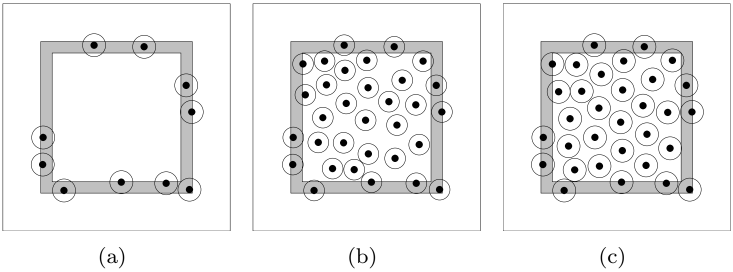


Figure 4.15: Construction of a tile of a template Poisson disk tile set. (a) The tile template. (b) A Poisson disk distribution is generated. (c) The Poisson disk distribution is optimized using Lloyd's relaxation scheme.

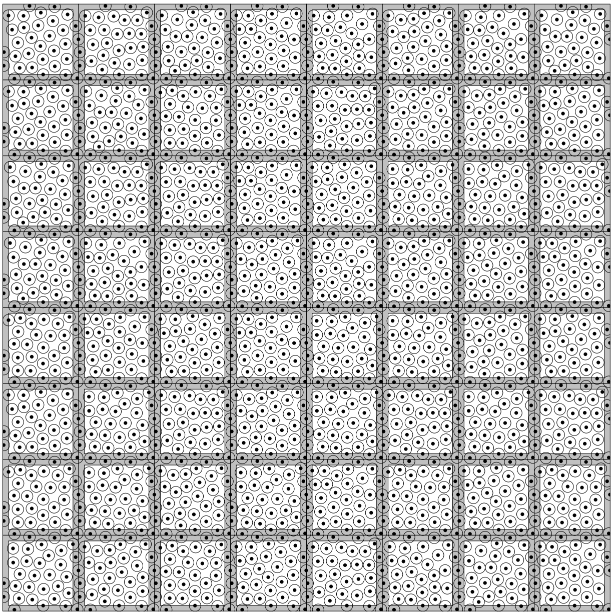


Figure 4.16: A tiling with a set of template Poisson disk tiles.

with edge-based Poisson disk tiles, the size of a set of template Poisson disk tiles is not constrained. This allows to investigate the effect of the size of the tile set on the quality of the resulting Poisson disk distributions (see section 6.10).

A set of template Poisson disk tiles is a set of unit square tiles that share the same toroidal boundary of points. Once a set of template Poisson disk tiles is constructed, a tiled Poisson disk distribution can be generated by placing tiles chosen randomly from the tile set next to each other.

To construct a set of template Poisson disk tiles, the number of points per tile N, the number of tiles T, and the relative Poisson disk radius ρ are chosen.

First, a single tile called a master tile is created. A toroidal Poisson disk distribution is generated using dart throwing or relaxation dart throwing, optionally followed by Lloyd's relaxation. If the desired Poisson disk radius is not reached, this process is repeated. This is illustrated in figure 4.13.

A tiled Poisson disk distribution can be generated using only the master tile, because the distribution it contains is toroidal. However, there will be much repetition in the distribution, something that should be avoided.

Next, a tile template is created. This is illustrated in figure 4.14. To ensure that a Poisson disk distribution is continuous over tile edges, only points closer to an edge of the tile than the Poisson disk radius need to be considered. The tile template is created by discarding all points in the interior of the master

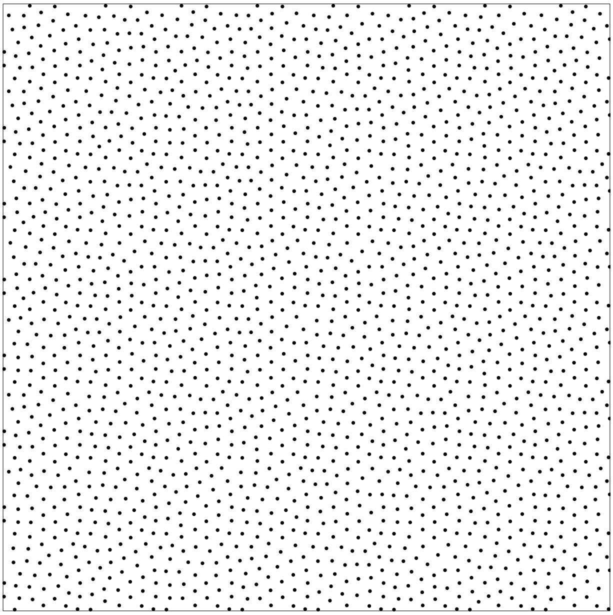


Figure 4.17: A Poisson disk distribution generated with a set of template Poisson disk tiles. This Poisson disk distribution was generated from the tiling shown in figure 4.16.

tile.

Finally, the remaining $T-1$ tiles in the tile set are created. This is illustrated in figure 4.15. Each tile is created by filling the tile template with a Poisson disk distribution that brings the total number of points in the tile to N. This distribution is generated using dart throwing or relaxation dart throwing, optionally followed by Lloyd's relaxation method. However, no new points are added to the tile template, and during relaxation, points are prohibited to enter the tile template. This is done by clipping displacement vectors of points that are about to enter the tile template.

Figure 4.16 shows a tiling with a set of template Poisson disk tiles, and figure 4.17 shows the resulting Poisson disk distribution.

A set of template Poisson disk tiles can be seen as a set of several instances of a single Wang tile or corner tile. This technique can therefore be used to bring more variation in a set of edge-based Poisson disk tiles (see section 4.3) or a set of corner-based Poisson disk tiles (see section 4.5).

The properties of the Poisson disk distributions generated with template Poisson disk tiles are discussed in detail in section 6.10.

4.5 Corner-Based Poisson Disk Tiles

Template Poisson disk tiles were developed to study the effect of the size of the tile set on the quality of the tiled Poisson disk distributions. In chapter 6 we show that a set of $4,096$ edge-based Poisson disk tiles is enough but too much for generating high-quality tiled Poisson disk distributions. Template Poisson disk tiles are also not suited because the tile template causes too much repetition in the generated Poisson disk distributions. In 2006, we presented corner tiles and corner-based Poisson disk tiles, a method for constructing a Poisson disk distribution over a set of corner tiles [Lagae and Dutré, 2006a].

By combining the modified Poisson disk tile regions with the complete corner tile set over C colors, a new tiling is obtained. This is illustrated in figure 4.18. This tiling uses three different kinds of tiles. Corner tiles, horizontal and vertical edge tiles, and interior tiles. Corner tiles correspond to the union of four modified corner regions. There are C corner tiles, one for each color. Edge tiles correspond to the union of two modified edge regions. There are C^2 horizontal and C^2 vertical edge tiles, one for each combination of two corner colors. Interior tiles correspond to the modified interior regions. There are C^4 interior tiles, one for each combination of four corner colors. Note that edge-based Poisson disk tiles use C^{12} rather than C^4 interior tiles.

To construct a Poisson disk distribution over a set of corner tiles, the number of colors of the corner tile set C, the number of points per tile N, and the relative Poisson disk radius ρ are chosen. The absolute Poisson disk radius determines the size of the modified Poisson disk regions.

First, a Poisson disk distribution is constructed over the corner tiles. This is

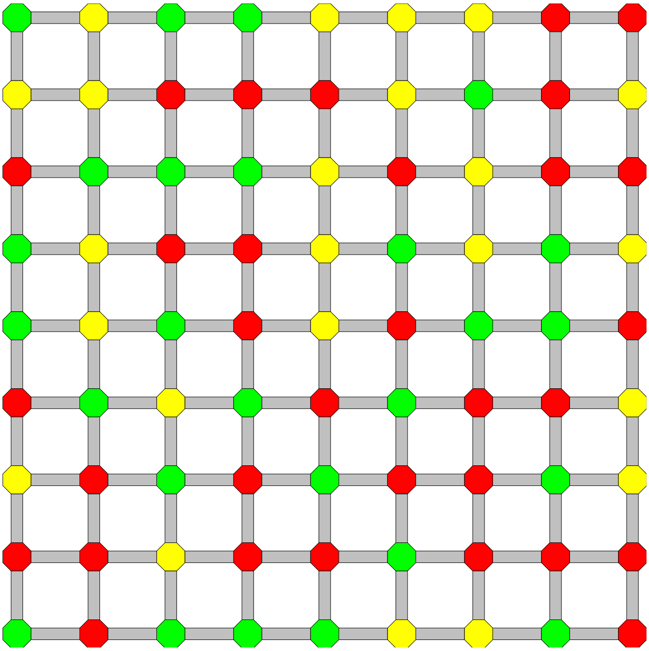


Figure 4.18: A tiling obtained by combining the modified Poisson disk tile regions with the complete corner tile set over 3 colors. This tiling was generated from the tiling shown in figure 2.6.

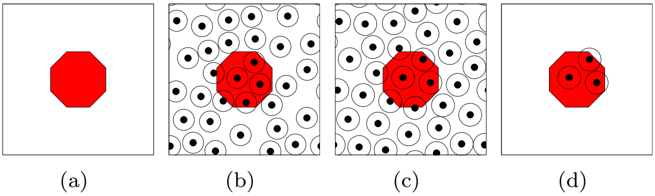


Figure 4.19: Construction of a Poisson disk distribution over a corner tile of a corner-based Poisson disk tile set. (a) The corner tile. (b) A toroidal Poisson disk distribution is generated. (c) The Poisson disk distribution is optimized using Lloyd's relaxation scheme. (d) The corner tile is cut out of the Poisson disk distribution.

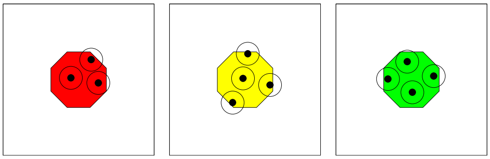


Figure 4.20: Poisson disk distributions constructed over corner tiles of a corner-based Poisson disk tile set.

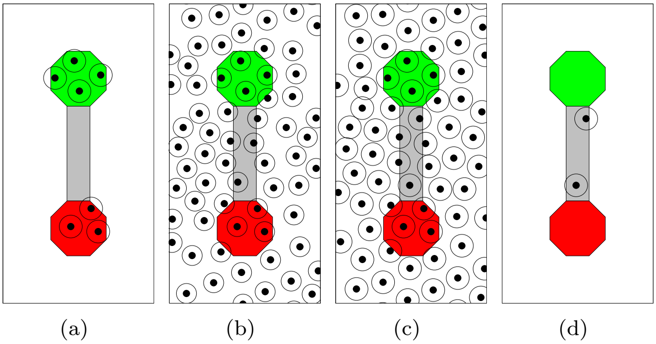


Figure 4.21: Construction of a Poisson disk distribution over a vertical edge tile of a corner-based Poisson disk tile set. (a) The edge tile is assembled with the corresponding corner tiles. (b) A toroidal Poisson disk distribution is generated. (c) The Poisson disk distribution is optimized using Lloyd's relaxation scheme. (d) The edge tile is cut out of the Poisson disk distribution.

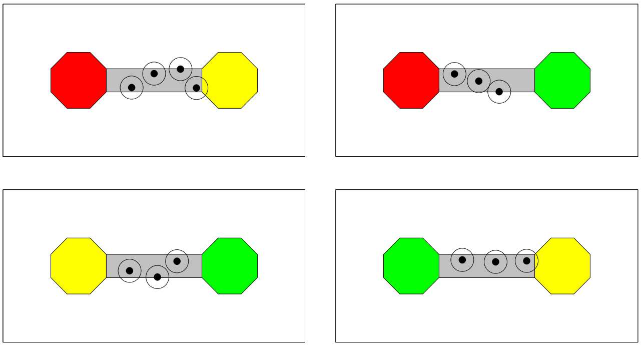


Figure 4.22: Poisson disk distributions constructed over horizontal edge tiles of a corner-based Poisson disk tile set.

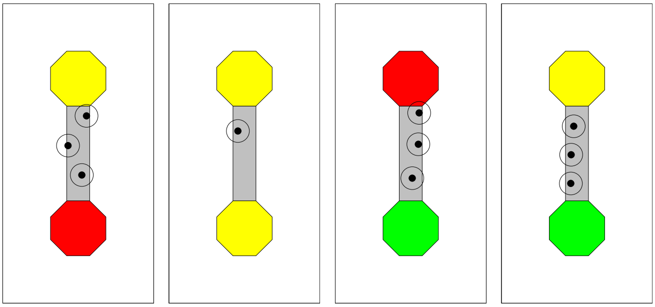


Figure 4.23: Poisson disk distributions constructed over vertical edge tiles of a corner-based Poisson disk tile set.

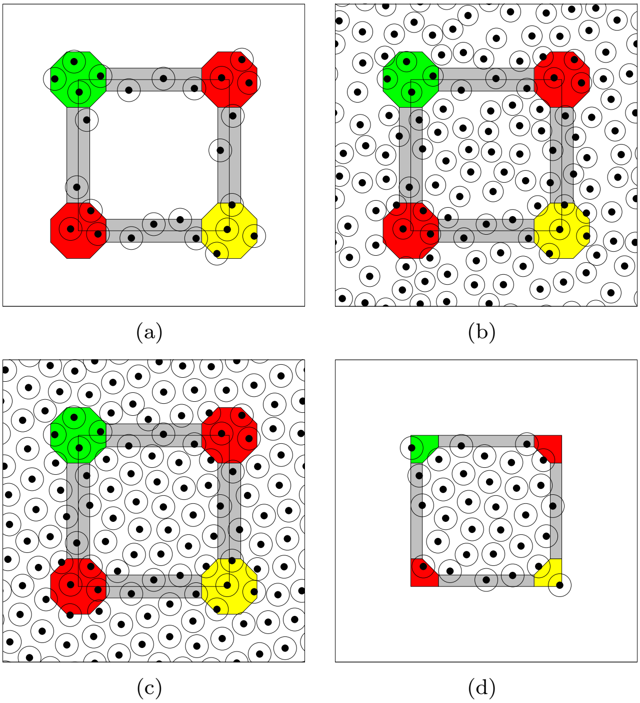


Figure 4.24: Construction of a Poisson disk distribution over a tile of a corner-based Poisson disk tile set. (a) The interior tile is assembled with the corresponding corner tiles and edge tiles. (b) A toroidal Poisson disk distribution is generated. (c) The Poisson disk distribution is optimized using Lloyd's relaxation scheme. (d) The tile is cut out of the Poisson disk distribution.

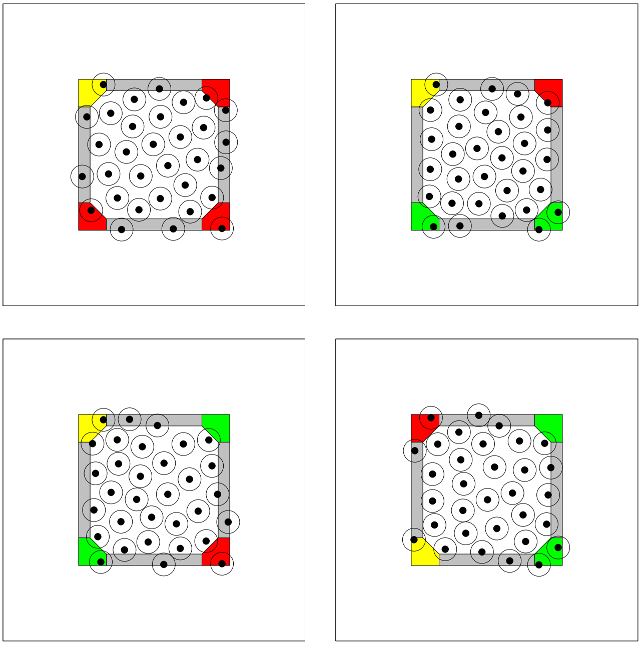


Figure 4.25: Poisson disk distributions constructed over tiles of a corner-based Poisson disk tile set.

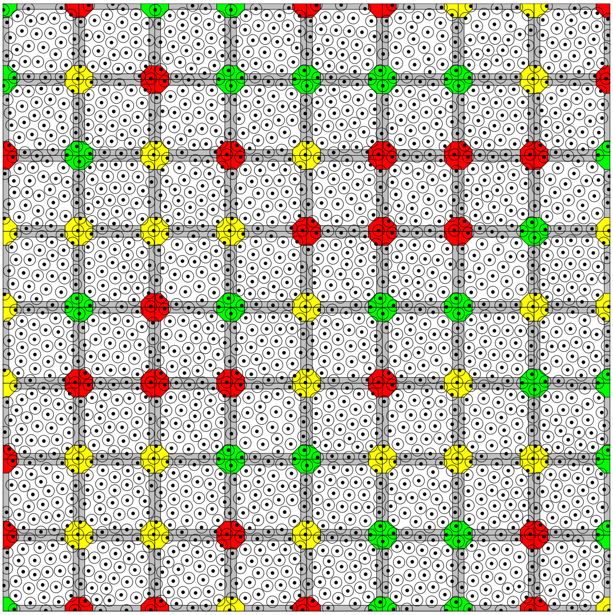


Figure 4.26: A tiling with a set of corner-based Poisson disk tiles.

illustrated in figure 4.19. For each corner tile, a toroidal Poisson disk distribution of N points is generated using dart throwing or relaxation dart throwing (see figure 4.19(b)), optionally followed by Lloyd's relaxation method (see figure 4.19(c)). The corner tile is then cut out of the Poisson disk distribution (see figure 4.19(d)). If the desired Poisson disk radius is not reached, this process is repeated. Figure 4.20 shows Poisson disk distributions constructed over corner tiles.

Next, a Poisson disk distribution is constructed over the edge tiles. This is illustrated in figure 4.21. Each edge tile is assembled with the corresponding corner tiles (see figure 4.21(a)). A toroidal Poisson disk distribution is generated using dart throwing or relaxation dart throwing (see figure 4.21(b)), optionally followed by Lloyd's relaxation method (see figure 4.21(c)). The edge tile is then cut out of the Poisson disk distribution (see figure 4.21(d)). If the desired Poisson disk radius is not reached, this process is repeated. No new points are added to the corner tiles. During relaxation, the points in the corner tiles are fixed, and other points are prohibited to enter the corner tiles. This is done by clipping the displacement vectors of points that are about to enter the corner tiles. Figures 4.22 and 4.23 show Poisson disk distributions constructed over horizontal and vertical edge tiles.

Finally, a Poisson disk distribution is constructed over the interior tiles. This is illustrated in figure 4.24. Each interior tile is assembled with the correspond-

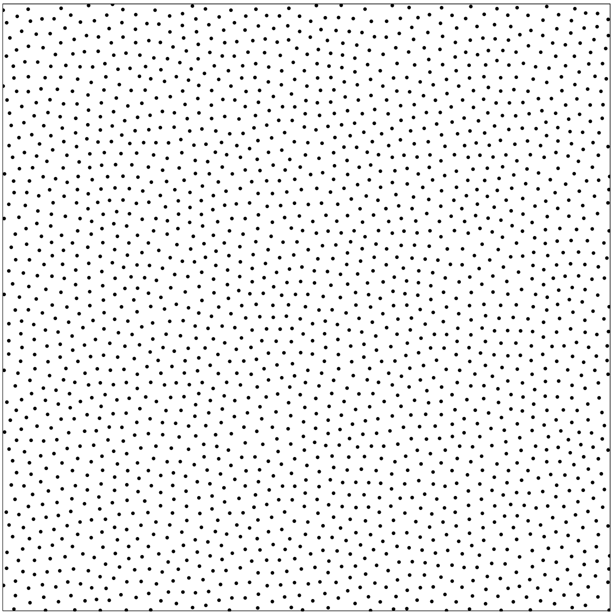


Figure 4.27: A Poisson disk distribution generated with a set of corner-based Poisson disk tiles. This Poisson disk distribution was generated from the tiling shown in figure 4.26.

ing corner tiles and edge tiles (see figure 4.24(a)). A toroidal Poisson disk distribution that brings the number of points inside the tile to N is generated using dart throwing or relaxation dart throwing (see figure 4.24(b)), optionally followed by Lloyd's relaxation method (see figure 4.24(c)). The tile is then cut out of the Poisson disk distribution (see figure 4.24(d)). If the desired Poisson disk radius is not reached, this process is repeated. No new points are added to the corner tiles and the edge tiles. During relaxation, the points in the corner tiles and the edge are fixed, and other points are prohibited to enter the corner tiles and the edge tiles. Figure 4.25 shows Poisson disk distributions constructed over corner tiles.

A corner-based Poisson disk tile set based on a complete Wang tile set over C colors consists of C^4 tiles. For 2, 3, 4, 5, 6, 7 and 8 colors, a set of corner-based Poisson disk tiles counts 16, 81, 256, 625, $1,296$, $2,401$ and $4,096$. Corner-based Poisson disk tile sets are significantly smaller than edge-based Poisson disk tile sets.

Figure 4.26 shows a tiling with a set of edge-based Poisson disk tiles, and figure 4.27 shows the resulting Poisson disk distribution.

The time needed to generate a Poisson disk tile set ranges from several minutes to several hours, depending on the parameters. However, the construction of a tile set has to be done only once. With a single set of tiles, an infinite number of Poisson disk distributions can be generated.

The properties of the Poisson disk distributions generated with corner-based Poisson disk tiles are discussed in detail in section 6.11. Although the difference between corner-based Poisson disk tiles and edge-based Poisson disk tiles is relatively small, we will show that the increase in quality of the resulting Poisson disk distributions is large.

4.6 A Tile-Based Method for Generating Poisson Sphere Distributions

Poisson disk distributions have many applications in computer graphics. Several of these applications, such as geometric object distribution (see section 7.5) and the two-dimensional procedural object distribution texture basis function (see section 7.6), have three-dimensional counterparts. These applications require Poisson sphere distributions, the three-dimensional equivalent of Poisson disk distributions.

In this section we present Poisson sphere distributions and three-dimensional corner tiles, and we introduce corner-based Poisson sphere tiles, a tile-based method for efficiently generating Poisson sphere distributions.

4.6.1 Poisson Sphere Distributions

Poisson sphere distributions are very similar to Poisson disk distributions, and most concepts of Poisson disk distributions easily generalize to Poisson sphere distribution. In this subsection we define Poisson sphere distributions, we extend the relative radius specification scheme to Poisson sphere distributions, and we discuss methods for generating Poisson sphere distributions.

4.6.1.1 Definition

A Poisson sphere distribution is a three-dimensional Poisson distribution in which all points are separated from each other by a minimum distance. Half that distance is called the radius r of the distribution. If a sphere of that radius is placed at each point, then no two spheres intersect.

4.6.1.2 Radius Specification

The relative radius specification scheme for Poisson disk distributions has a direct three-dimensional equivalent.

The packing density η of the densest packing of spheres is given by

$$\eta = \frac{\pi}{3\sqrt{2}} \approx 0.7405. \tag{4.4}$$

The packing density is defined as the fraction of the volume filled by the spheres.

The problem of finding the densest packing of spheres, also known as the Kepler Problem, was only solved recently [Cipra, 1998].

The maximum sphere volume of a Poisson sphere distribution counting N points over the toroidal unit cube is therefore η/N. The maximum possible sphere radius r_{max} of this Poisson sphere distribution is thus given by

$$r_{max} = \sqrt[3]{\frac{1}{4\sqrt{2}N}}. \tag{4.5}$$

The Poisson sphere radius r of a given point distribution is specified as a fraction ρ of the maximum disk radius

$$r = \rho\, r_{max}, \tag{4.6}$$

where $\rho \in [0, 1]$.

As in two dimensions, the relative radius is also a measure of how well the points are distributed. Good Poisson sphere distributions should have a relative radius that is relatively high.

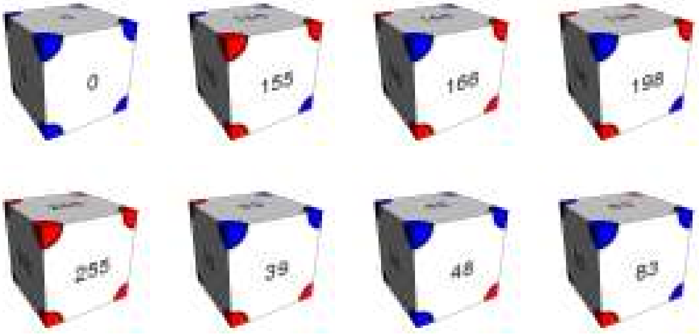


Figure 4.28: Several tiles of the complete 3D corner tile set over 2 colors.

4.6.1.3 Generation

Poisson disk distributions are traditionally generated with dart throwing, relaxation dart throwing or Lloyd's relaxation method. These algorithms easily generalize to three dimensions. Dart throwing, relaxation dart throwing and Lloyd's relaxation method for Poisson sphere distributions are discussed in more detail in section 6.15.

4.6.2 Three-Dimensional Corner Tiles

Three-dimensional corner tiles are a simple extension of two-dimensional corner tiles.

Three-dimensional corner tiles are unit cube tiles with colored corners. The corners of a three-dimensional corner tile are named after its coordinates. A complete tile set contains a tile for every possible combination of eight corner colors. A complete set of three-dimensional corner tiles over C colors counts C^8 tiles. Figure 4.28 shows several tiles of the complete set of three-dimensional corner tiles over two colors. A complete set of three-dimensional corner tiles over 2, 3 and 4 colors consist of 256, $6,561$ and $65,536$ tiles.

A tiling is constructed by placing the tiles next to each other such that adjoining corners have matching colors. Each tile in the tile set can be used arbitrarily many times. The tiles are placed with their corners on the integer lattice points. Figure 4.29 shows a tiling with the complete set of three-dimensional corner tiles over two colors.

The enumeration scheme for corner tiles (see section 2.8) and the direct stochastic tiling algorithm for corner tiles (see section 3.3) easily generalize to three dimensions.

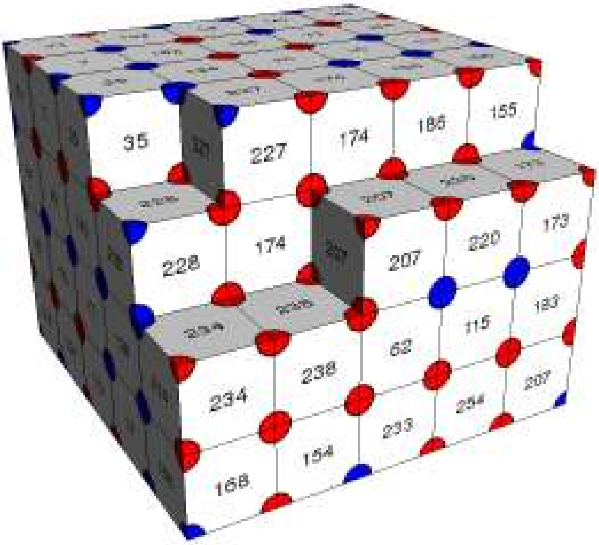


Figure 4.29: A tiling with the complete 3D corner tile set over 2 colors.

4.6.3 Corner-Based Poisson Sphere Tiles

In 2006, we presented corner-based Poisson sphere tiles, a tile-based method for efficiently generating Poisson sphere distributions [Lagae and Dutré, 2006e]. Corner-based Poisson sphere tiles are an extension of corner-based Poisson disk tiles (see section 4.5) to three dimensions.

A point in a tile closer to a corner than the Poisson sphere radius affects points in seven neighboring tiles. A point in a tile closer to an edge than the Poisson sphere radius affects points in three neighboring tiles. A point in a tile closer to a face than the Poisson sphere radius affects points in one neighboring tile. A point in a tile, further away from the tile boundary than the Poisson sphere radius does not affect points in neighboring tiles. The regions obtained this way are called the Poisson sphere tile regions. The Poisson sphere radius determines corner regions, edge regions, face regions and an interior region. This is illustrated in figure 4.30.

To minimize the constraints between the different regions, the corner regions are enlarged such that the distance between edge regions is twice the Poisson sphere radius. Now points in different edge regions do not affect each other. The edge regions are enlarged such that the distance between face regions is twice the Poisson sphere radius. Now points in different face regions do not affect each other. The regions obtained this way are called the modified Poisson sphere tile regions. This is illustrated in figure 4.31.

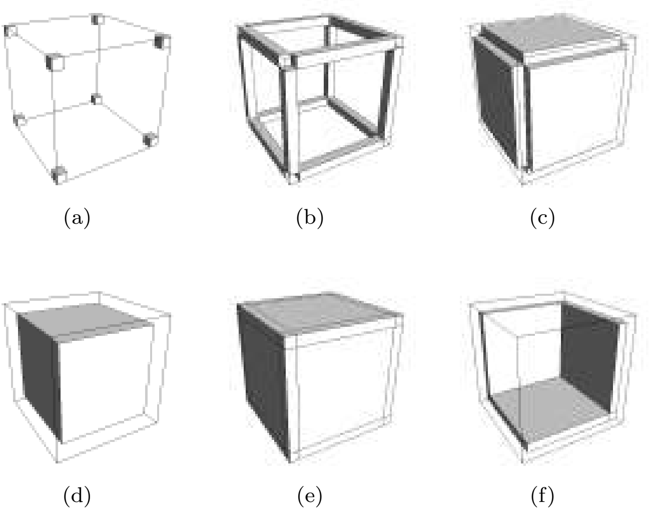


Figure 4.30: The Poisson sphere tile regions. (a) The corner regions. (b) The edge regions. (c) The face regions. (d) The interior region. (e) The assembled tile. (f) The partially assembled tile.

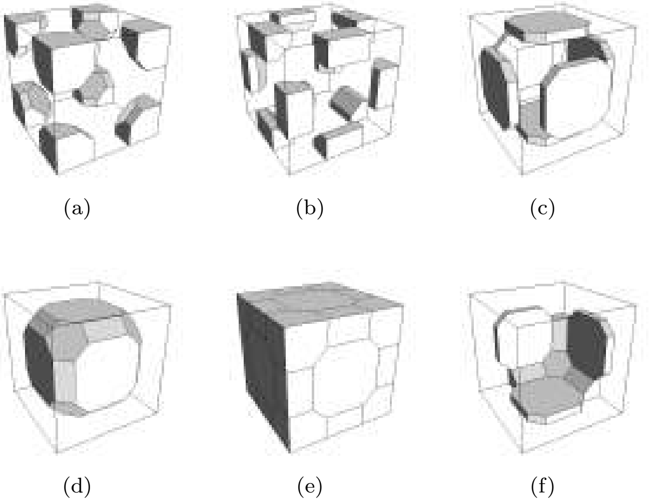


Figure 4.31: The modified Poisson sphere tile regions. (a) The modified corner regions. (b) The modified edge regions. (c) The modified face regions. (d) The modified interior region. (e) The assembled tile. (f) The partially assembled tile.

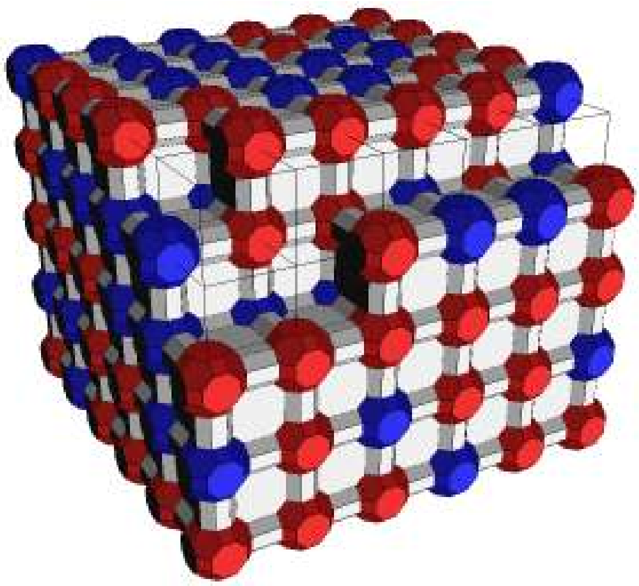


Figure 4.32: A tiling obtained by combining the modified Poisson sphere tile regions with the complete 3D corner tile set over 2 colors. This tiling was generated from the tiling shown in figure 4.29.

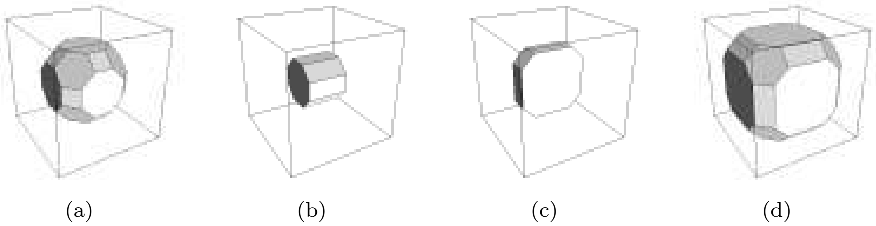
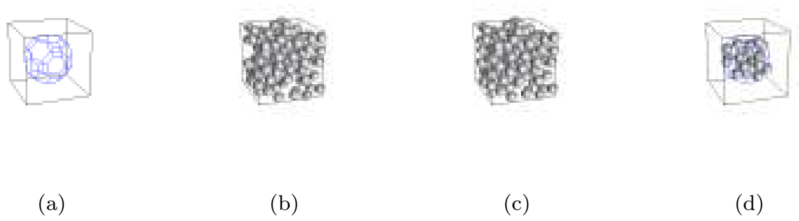
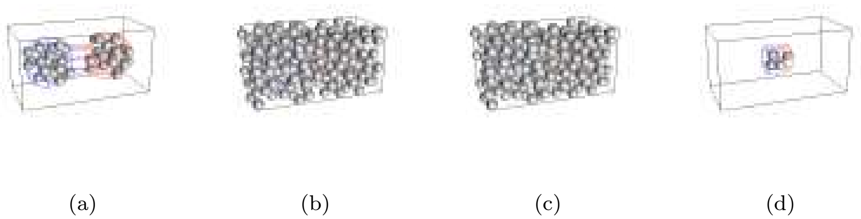


Figure 4.33: The four kinds of tiles in the tiling obtained by combining the modified Poisson sphere tile regions with the complete 3D corner tile set over 2 colors. (a) Corner tiles. (b) Edge tiles. (c) Face tiles. (d) Interior tiles.



(a) (b) (c) (d)

Figure 4.34: Construction of a Poisson sphere distribution over a corner tile of a corner-based Poisson sphere tile set. (a) The corner tile. (b) A toroidal Poisson sphere distribution is generated. (c) The Poisson sphere distribution is optimized using Lloyd's relaxation scheme. (d) The corner tile is cut out of the Poisson disk distribution.



(a) (b) (c) (d)

Figure 4.35: Construction of a Poisson sphere distribution over an edge tile of a corner-based Poisson sphere tile set. (a) The edge tile is assembled with the corresponding corner tiles. (b) A toroidal Poisson sphere distribution is generated. (c) The Poisson sphere distribution is optimized using Lloyd's relaxation scheme. (d) The edge tile is cut out of the Poisson sphere distribution.

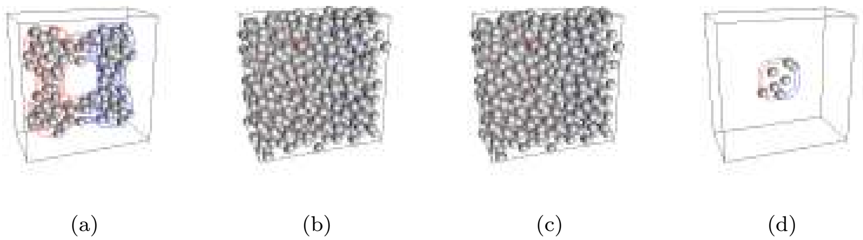


Figure 4.36: Construction of a Poisson sphere distribution over a face tile of a corner-based Poisson sphere tile set. (a) The face tile is assembled with the corresponding corner tiles and edge tiles. (b) A toroidal Poisson sphere distribution is generated. (c) The Poisson sphere distribution is optimized using Lloyd's relaxation scheme. (d) The face tile is cut out of the Poisson sphere distribution.

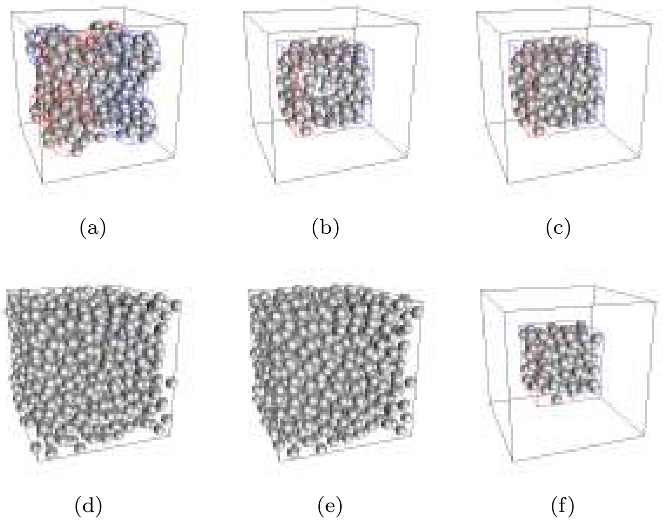


Figure 4.37: Construction of a Poisson sphere distribution over a tile of a corner-based Poisson sphere tile set. (a) The interior tile is assembled with the corresponding corner tiles, edge tiles and face tiles. (b) All points further from the tile than twice the Poisson sphere radius are discarded. (c) A toroidal Poisson sphere distribution is generated in the interior. (d) A toroidal Poisson sphere distribution is generated in the exterior. (e) The Poisson sphere distribution is optimized using Lloyd's relaxation scheme. (f) The tile is cut out of the Poisson sphere distribution.

By combining the modified Poisson sphere tile regions with the complete corner tile set over C colors, a new tiling is obtained. This is illustrated in figure 4.32. This tiling uses four different kinds of tiles. Corner tiles, edge tiles, face tiles and interior tiles. Corner tiles correspond to the union of eight modified corner regions. There are C corner tiles, one for each color. Edge tiles correspond to the union of four modified edge regions. There are C^2 edge tiles for each orientation, one for each combination of two corner colors. Face tiles correspond to the union of two modified face regions. There are C^4 face tiles for each orientation, one for each combination of four corner colors. Interior tiles correspond to the modified interior regions. There are C^8 interior tiles, one for each combination of eight corner colors. These four kinds of tiles are shown in figure 4.33. Note that the corner tile is a great rhombicuboctahedron, an Archimedean solid.

To construct a Poisson sphere distribution over a set of corner tiles, the number of colors of the corner tile set C, the number of points per tile N, and the relative Poisson sphere radius ρ are chosen. The absolute Poisson sphere radius determines the size of the modified Poisson sphere regions.

First, a Poisson sphere distribution is constructed over the corner tiles. This is illustrated in figure 4.34. For each corner tile, a toroidal Poisson sphere distribution of N points is generated using dart throwing or relaxation dart throwing (see figure 4.34(b)), optionally followed by Lloyd's relaxation method (see figure 4.34(c)). The corner tile is then cut out of the Poisson sphere distribution (see figure 4.34(d)). If the desired Poisson sphere radius is not reached, this process is repeated.

Next, a Poisson sphere distribution is constructed over the edge tiles. This is illustrated in figure 4.35. Each edge tile is assembled with the corresponding corner tiles (see figure 4.35(a)). A toroidal Poisson sphere distribution is generated using dart throwing or relaxation dart throwing (see figure 4.35(b)), optionally followed by Lloyd's relaxation method (see figure 4.35(c)). The edge tile is then cut out of the Poisson sphere distribution (see figure 4.35(d)). If the desired Poisson sphere radius is not reached, this process is repeated. No new points are added to the corner tiles. During relaxation, the points in the corner tiles are fixed, and other points are prohibited to enter the corner tiles. This is done by clipping the displacement vectors of points that are about to enter the corner tiles.

Next, a Poisson sphere distribution is constructed over the face tiles in the same way. This is illustrated in figure 4.36.

Finally, a Poisson sphere distribution is constructed over the interior tiles. This is illustrated in figure 4.37. Each interior tile is assembled with the corresponding corner tiles, edge tiles and face tiles. (see figure 4.37(a)). All points further away from the tile boundary than the Poisson sphere radius are discarded (see figure 4.37(b)). A Poisson sphere distribution is generated using dart throwing or relaxation dart throwing, both in the inside (see figure 4.37(c)) and in the outside (see figure 4.37(d)), optionally followed by Lloyd's relaxation method (see figure 4.37(e)). The tile is then cut out of the Poisson sphere distribution (see figure 4.37(f)). If the desired Poisson sphere radius is not reached, this process is repeated.

A corner-based Poisson sphere tile set based on a complete corner tile set over C colors consists of C^8 tiles. The only practical choice for C is 2, which results in a corner-based Poisson sphere tile set counting 256 tiles.

The time needed to generate a Poisson sphere tile set ranges from several minutes to several hours, depending on the parameters. However, the construction of a tile set has to be done only once. With a single set of tiles, an infinite number of Poisson sphere distributions can be generated.

The properties of the Poisson sphere distributions generated with corner-based Poisson sphere tiles are discussed in detail in section 6.15.

4.7 A Tile-Based Method for Generating Nonuniform Poisson Disk Distributions

Poisson disk distributions have many applications in computer graphics. Several of these applications, such as sampling (see section 7.2) and primitive distribution for illustration (see section 7.3), would also benefit from Poisson disk distributions with nonuniform density. In this section we investigate nonuniform Poisson disk distributions. We define nonuniform Poisson disk distributions and we present a tile-based method for efficiently generating nonuniform Poisson disk distributions.

4.7.1 Nonuniform Poisson Disk Distributions

Poisson disk distributions with nonuniform density are not very well studied. There is no established terminology nor definition. Nonuniform Poisson disk distributions are sometimes called adaptive Poisson disk distributions [McCool and Fiume, 1992] or distributions with blue-noise properties [Ostromoukhov et al., 2004; Kopf et al., 2006]. We define a nonuniform Poisson disk distribution as a two-dimensional Poisson distribution in which the point density is proportional to a given density function, and the points are separated from each other by a minimum distance inversely proportional to the given density function.

Poisson disk distributions with nonuniform density can be generated with a technique similar to Lloyd's relaxation method [Du et al., 1999]. However, the challenge is to generate nonuniform Poisson disk distributions in real time. McCool and Fiume [1992] proposed a technique to generate adaptive Poisson disk distributions by thresholding a precomputed hierarchical Poisson disk distribution. Ostromoukhov et al. [2004] proposed a method to generate point distributions with blue-noise properties by moving the vertices of a recursively

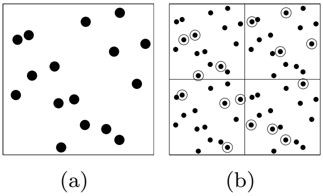


Figure 4.38: A self-similar point distribution. (a) The point distribution. (b) The point distribution scaled down by a factor of two, superimposed on each quadrant. The points of the original point distribution are circled.

subdivided Penrose tiling according to precomputed relaxation vectors. Concurrently with our work, Kopf et al. [2006] presented a technique to generate point distributions with blue-noise properties based on recursive Wang tiles that contain self-similar and progressive Poisson disk distributions.

However, non of these methods adequately solves the problem of generating Poisson disk distributions with nonuniform density. The method of McCool and Fiume is limited by the precomputed hierarchical Poisson disk distribution, and the techniques of Ostromoukhov et al. and Kopf et al. seem to have problems with guaranteeing the minimum radius.

4.7.2 A Self-Similar Hierarchical Tile

In 2006, we presented a tile-based method for generating nonuniform well-distributed point distributions [Lagae and Dutré, 2006c].

Like previous techniques, our method does not adequately solve the problem of generating nonuniform Poisson disk distributions. It produces deterministic point sequences rather than Poisson disk distributions. Therefore, we use the term well-distributed rather than Poisson disk. However, we hope that the technique we present provides additional insight into the problem.

Our method is based on a single precomputed tile. The tile contains a well-distributed point distribution that is both self-similar and hierarchical. A tile with a self-similar point distribution allows to increase the density of points in large steps by recursively subdividing the tile. A tile with a hierarchical point distribution allows to smoothly adjust the density of points. A tile containing a point distribution with both properties can be used to efficiently generate well-distributed point distributions according to a given density function.

4.7.2.1 Self-Similar Point Distributions

A self-similar point distribution can be defined in several ways. In general, a point distribution is self-similar if it looks roughly the same on any scale. For this method, we use the following more concrete definition. A point distribution

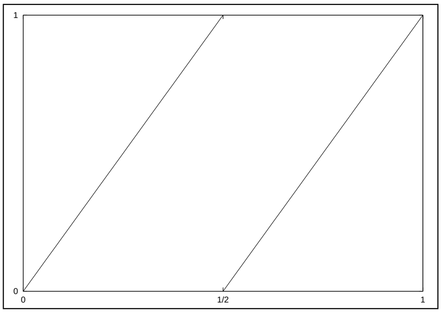


Figure 4.39: The function characterizing the self-similar tile.

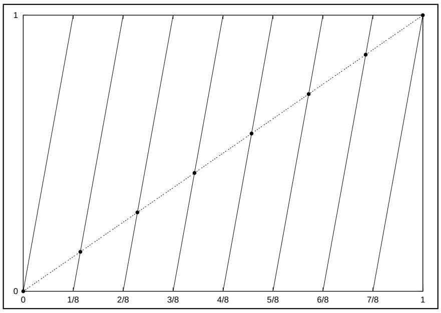


Figure 4.40: Three repeated applications of the function characterizing the self-similar tile.

defined over a square domain is self-similar if the point distribution obtained by superimposing the point distribution scaled down by a factor of two on any of the quadrants of the domain is the original point distribution scaled down by a factor of two. Figure 4.38 shows an example of a self-similar point distribution.

For each point in a self-similar point distribution, there is another point in the distribution that, when the point distribution is scaled down and put in the appropriate quadrant, will coincide with the point. This property can be used to construct a self-similar point distribution.

Without loss of generality, assume that the domain of the point distribution is $[0,1[^2$. Suppose the self-similar point distribution is constructed starting with a single point (x,y). Due to the above property, a second point

$$\begin{cases} (2x,2y) & \text{if } 0 \leq x < \frac{1}{2} \text{ and } 0 \leq y < \frac{1}{2} \\ (2x,2y-1) & \text{if } 0 \leq x < \frac{1}{2} \text{ and } \frac{1}{2} \leq y < 1 \\ (2x-1,2y) & \text{if } \frac{1}{2} \leq x < 1 \text{ and } 0 \leq y < \frac{1}{2} \\ (2x-1,2y-1) & \text{if } \frac{1}{2} \leq x < 1 \text{ and } \frac{1}{2} \leq y < 1 \end{cases} \tag{4.7}$$

must be added. Or, more compactly, whenever a point (x,y) is added to the tile, the point $(f(x),f(y))$, with

$$f(x) = \begin{cases} 2x & \text{if } 0 \leq x < \frac{1}{2} \\ 2x-1 & \text{if } \frac{1}{2} \leq x < 1 \end{cases}, \tag{4.8}$$

must also be added. The function f is shown in figure 4.39.

However, now a third point $(f(f(x)),f(f(y)))$ must be added as well. In order to generate a point distribution with N points, the $(N+1)$th point $(f^{(N)}(x),f^{(N)}(y))$ has to be equal to the first point (x,y). Here $f^{(N)}$ is used to denote N applications of the function f. In other words, x and y are fixed points of the function $f^{(N)}$.

The function $f^{(N)}$ is a piecewise linear function, given by

$$f^{(N)}(x) = \begin{cases} 2^N x - i & \text{if } \frac{i}{2^N} \leq x < \frac{i+1}{2^N} \end{cases}, \text{ for } i = 0 \ldots 2^N - 1. \tag{4.9}$$

The function $f^{(3)}$ is shown in figure 4.40. The $2^N - 1$ fixed points of $f^{(N)}$ are given by

$$\frac{i}{2^N - 1}, \text{ for } i = 0 \ldots 2^N - 2. \tag{4.10}$$

Note that 1 is not counted as a fixed point because the domain of f is $[0,1[$.

Thus, a self-similar point distribution of N points can be constructed by choosing two arbitrary fixed points x and y of the function $f^{(N)}$ and forming the deterministic sequence

$$\left\{(f^{(i)}(x),f^{(i)}(y))\right\}_{i=0}^{N-1}. \tag{4.11}$$

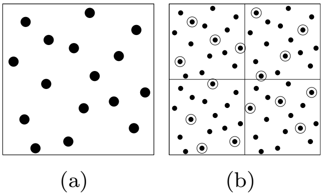


Figure 4.41: A well-distributed self-similar point distribution. (a) The point distribution. (b) The point distribution scaled down by a factor of two, superimposed on each quadrant. The points of the original point distribution are circled.

The self-similar point distribution shown in figure 4.38 counts 16 points and was generated using the fixed points with index $35,569$ and $12,988$.

It is interesting to note that the criterion of self-similarity results in a deterministic point sequence. Besides the choice of fixed points, no degrees of freedom are left.

Also note that this method can be extended in several ways. For example, point distributions in three or more dimensions could be generated, or three-way subdivision could be used instead of two-way subdivision.

4.7.2.2 Well-Distributed Self-Similar Point Distributions

The points of a self-similar point distribution are in general not well-distributed. However, for a given number of points, there are a lot of self-similar point distributions, and it is relatively easy to construct them. Therefore, instead of explicitly constructing a well-distributed self-similar point distribution, a well-distributed self-similar point distribution is obtained by generating self-similar point distributions and checking if the points are well-distributed.

No matter which criterion is used to determine whether a point distribution is well-distributed, it is reasonable to assume that each quadrant of the point distribution should contains an equal number of points. Note that this limits the number of points to multiples of 4. Remember that a self-similar point distribution is constructed by combining two sequences $\left\{f^{(i)}(x)\right\}_{i=0}^{N-1}$ and $\left\{f^{(i)}(y)\right\}_{i=0}^{N-1}$, with x and y fixed points of $f^{(N)}$. A sequence is called balanced if it contains as many elements in $[0, \frac{1}{2}[$ as in $[\frac{1}{2}, 1[$. A point distribution is called balanced if each quadrant contains an equal amount of points. Only two balanced sequences can produce a balanced point distribution.

The algorithm to construct a well-distributed self-similar point distribution counting N points iterates over all fixed points x of $f^{(N)}$. The algorithm constructs the sequence $\left\{f^{(i)}(x)\right\}_{i=0}^{N-1}$ and checks whether it is balanced. If the sequence is balanced, the algorithm iterates over all fixed points y of $f^{(N)}$, and

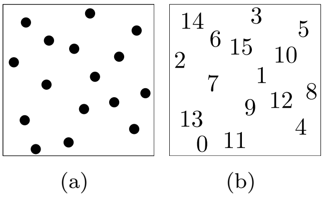


Figure 4.42: A hierarchical well-distributed point distribution. (a) The point distribution. (b) The rank assigned to each point.

constructs the sequence $\left\{f^{(i)}(y)\right\}_{i=0}^{N-1}$. If this sequence is also balanced, then the point distribution is constructed and the algorithm checks if the distribution is balanced. If the point distribution is balanced, a criterion is evaluated that determines how well the points are distributed. We use the relative Poisson disk radius to determine the quality of a point distribution, although other measures can also be used.

The time needed to construct and evaluate all 16-point self-similar point distributions is about 8 minutes on a regular desktop computer. The best point distribution has a relative radius of 0.6966, but is very symmetric. The second-best point distribution, with a relative radius of 0.6528, is shown in figure 4.41. It was generated using the fixed points with index 14217 and 2895. Approximately 19.64% of the 16-point sequences is balanced.

The time needed to construct and evaluate all 20-point self-similar point distributions is about three and a half day. The best point distributions have a relative radius of 0.8324, 0.7141 and 0.6713, but again are very symmetric. The fourth-best point distribution, with a relative radius of 0.6588, is not symmetric. Approximately 17.62% of the 20-point sequences is balanced. For 24 and 32 points, respectively 16.12% and 14.00% of the sequences is balanced.

The running times illustrate that this algorithm is only practical for a small number of points per tile. However, a small number of points per tile seems to be sufficient (see chapter 6). Moreover, there is still room for improvement. It is not necessary to construct all the self-similar point distributions, and the algorithm does not take into account symmetry. For example, the fact that cyclic permutations of sequences result in the same point distributions is not exploited.

4.7.2.3 Hierarchical Well-Distributed Point Distributions

A hierarchical (or progressive) well-distributed point distribution of N points is a well-distributed point distribution in which a rank is associated with each point such that the first i points of the distribution are well-distributed, for any $i \in \{1, 2, \ldots, N\}$. Figure 4.42 shows a hierarchical point distribution, and figure 4.43 shows well-distributed point distributions of increasing density

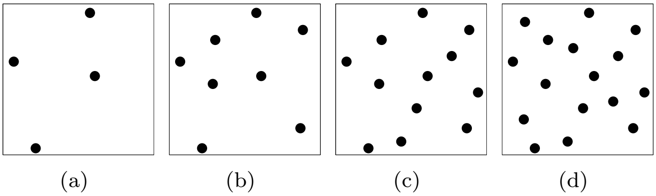


Figure 4.43: Well-distributed point distributions of increasing density generated using a hierarchical well-distributed point distribution. The first (a) 4, (b) 8, (c) 12 and (d) 16 points of a well-distributed hierarchical point distribution. Note that each point distribution is well-distributed.

generated with the hierarchical point distribution. Note that the hierarchical point distribution is toroidal. This ensures that the distributions obtained by tiling are still well-distributed.

A hierarchical distribution allows the use of thresholding for smoothly adjusting the density of points [McCool and Fiume, 1992]. A point is included based on its rank and the value of the density function at that point. The process of thresholding is also used in stochastic screen half toning for electronic imaging devices [Mitsa and Parker, 1991].

Although methods for generating hierarchical Poisson disk distribution exist [McCool and Fiume, 1992], it is also possible to assign a rank to each point after the point distribution has been generated.

We use the following algorithm. An arbitrary point is assigned rank 0. The next rank is assigned to the point that, when added to the point distribution, results in the most well-distributed point set, according to the criterion used in the previous section. Although this algorithm is greedy, it works well in practice. Figure 4.42 shows a ranking computed using this algorithm.

4.7.2.4 Generating Non-Uniform Well-Distributed Point Distributions

Our method for generating nonuniform well-distributed point distributions in real time is based on a single precomputed tile that contains a well-distributed point distribution that is both self-similar and hierarchical.

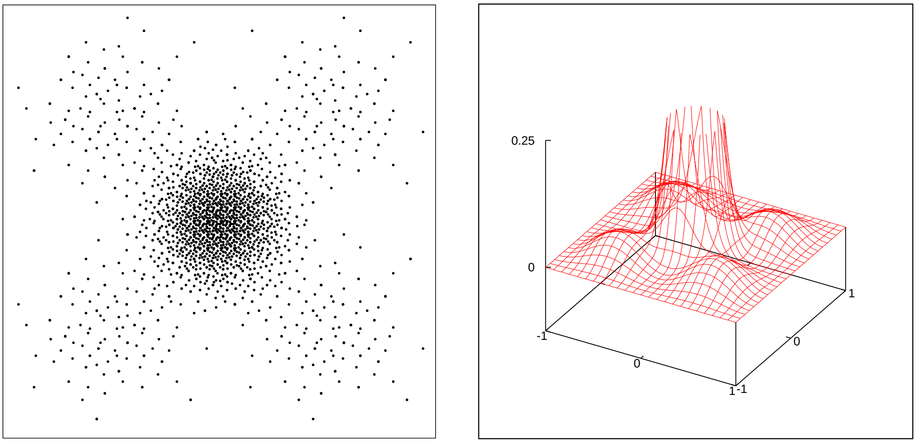
A nonuniform point distribution is generated according to a given density function by iterating over all points of the tile. A point is included in the distribution based on it's rank and the value of the density function at that point. If a density value is encountered that exceeds the maximum density of the tile, the tile is subdivided and the process is repeated.

Figure 4.44 shows a nonuniform well-distributed point distribution generated with the tile shown in figure 4.42. The coordinates of the points of the tile are shown in table 4.1.

A disadvantage of the method is that only a single tile is used. This re-

rank	coordinates	
	rational numbers	floating point numbers
0	$(14217, 2895)$	$(0.21693751, 0.04417487)$
1	$(39876, 34215)$	$(0.60846876, 0.52208743)$
2	$(4719, 40470)$	$(0.07200732, 0.61753262)$
3	$(37752, 61620)$	$(0.57605859, 0.94026093)$
4	$(56868, 11580)$	$(0.86775006, 0.17669947)$
5	$(57933, 54210)$	$(0.88400092, 0.82719158)$
6	$(19938, 49875)$	$(0.30423438, 0.76104372)$
7	$(18876, 30810)$	$(0.28802930, 0.47013046)$
8	$(61734, 27105)$	$(0.94200046, 0.41359579)$
9	$(35127, 20235)$	$(0.53600366, 0.30876631)$
10	$(50331, 42885)$	$(0.76800183, 0.65438315)$
11	$(28434, 5790)$	$(0.43387503, 0.08834974)$
12	$(48201, 23160)$	$(0.73550011, 0.35339895)$
13	$(9438, 15405)$	$(0.14401465, 0.23506523)$
14	$(9969, 57705)$	$(0.15211719, 0.88052186)$
15	$(30867, 46320)$	$(0.47100023, 0.70679789)$

Table 4.1: The coordinates of the points of a 16-point self-similar hierarchical tile. The tile was constructed by hierarchically sorting the sequence produced by the fixed points with index 14, 217 and 2, 895, and is shown in figure 4.41. The nominator of the rational numbers is 65, 535 $(2^{16} - 1)$.



(a)

(b)

Figure 4.44: A well-distributed point distribution with nonuniform density generated with the self-similar hierarchical tile shown in figure 4.42 according to the density function $\exp^{-20(x^2+y^2)} + 0.05 \sin^2(\pi x) \sin^2(\pi y)$ in $[-1, +1]^2$. (a) The generated point distribution. (b) A plot of the density function.

sults in noticeable periodicity in the generated point distributions, especially in regions of constant density. This problem could be solved by constructing multiple compatible tiles, performing Lloyd's relaxation method after the point set has been generated, or storing precomputed relaxation vectors [Ostromoukhov et al., 2004].

Concurrently with this work, Kopf et al. [2006] have developed a method for generating point distributions with blue noise properties in real time. Their work is based on recursive Wang tiles tiles that are self-similar and progressive. However, the method for constructing the tiles is very different from this method. Also, Kopf et al. use multiple compatible tiles to avoid periodicity in the generated point distributions.

The method for constructing a self-similar point distribution is somewhat similar to the methods for constructing quasi-random sequences [Niederreiter, 1992] and embedded lattice rules [Sloan and Joe, 1994] used in quasi-Monte Carlo integration. It could be interesting to further investigate this similarity.

4.8 Conclusion

In this chapter we have introduced Poisson disk distributions and we have proposed an intuitive scheme for specifying the radius of a Poisson disk distribution. We have presented edge-based Poisson disk tiles, template Poisson disk tiles, and corner-based Poisson disk tiles, three tile-based methods for generating Poisson disk distributions. We have introduced Poisson sphere distributions and three-dimensional corner tiles, and we have presented corner-based Poisson sphere tiles, a tile-based method for constructing Poisson sphere distributions. We have discussed nonuniform Poisson disk distributions and we have proposed a tile-based method for generating nonuniform well-distribution point distributions.

Chapter 5

Tile-Based Methods for Texture Synthesis

5.1 Introduction

In computer graphics, Wang tiles and corner tiles are used to facilitate the synthesis of complex signals. A texture is a complex signal that is difficult to synthesize efficiently. This chapter presents a method for synthesizing a texture over a set of Wang tiles or corner tiles, and a tile-based texture mapping algorithm for efficiently generating an arbitrary large non-repeating texture using a set of precomputed tiles.

Overview

This chapter is organized as follows. In section 5.2 we introduce texture mapping and texture synthesis. Section 5.3 presents a method for synthesizing a texture over a set of Wang tiles or corner tiles. In section 5.4 we propose a tile-based texture mapping algorithm. Section 5.5 discusses the tile packing problem. In section 5.6 we conclude.

This chapter only discusses methods for synthesizing textures over a set of Wang tiles or corner tiles. Efficient tiling algorithms for Wang tiles and corner tiles are presented in chapter 3.

5.2 Texture Mapping and Texture Synthesis

Texture mapping was introduced in 1974 by Catmull as a method for increasing the visual complexity of computer-generated images without adding geometric detail. A texture map, or simply a texture, is mapped onto the surface of a shape to add color or detail to the shape.

A texture can be acquired in several ways. Possibilities include painting a texture, taking a digital photograph of a texture, and generating a texture procedurally. Procedural texture synthesis is discussed in detail in section 7.6. Texture synthesis is an alternative way to obtain textures. Texture synthesis

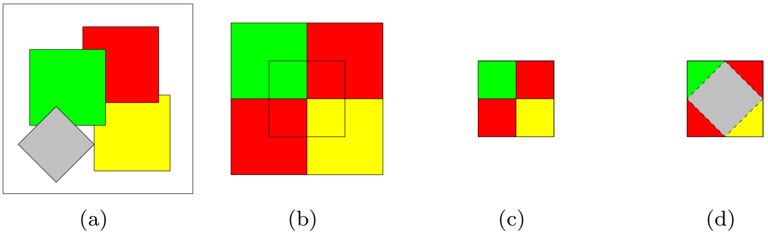


Figure 5.1: Construction of a texture tile set based on corner tiles from an example texture. (a) For each color, a square patch is chosen in the example texture (the red, green and yellow patch). (b) The patches are assembled according to the corner colors of the tile. (c) The tile is cut out. (d) The seam is covered with a new irregular patch from the example texture (the gray patch).

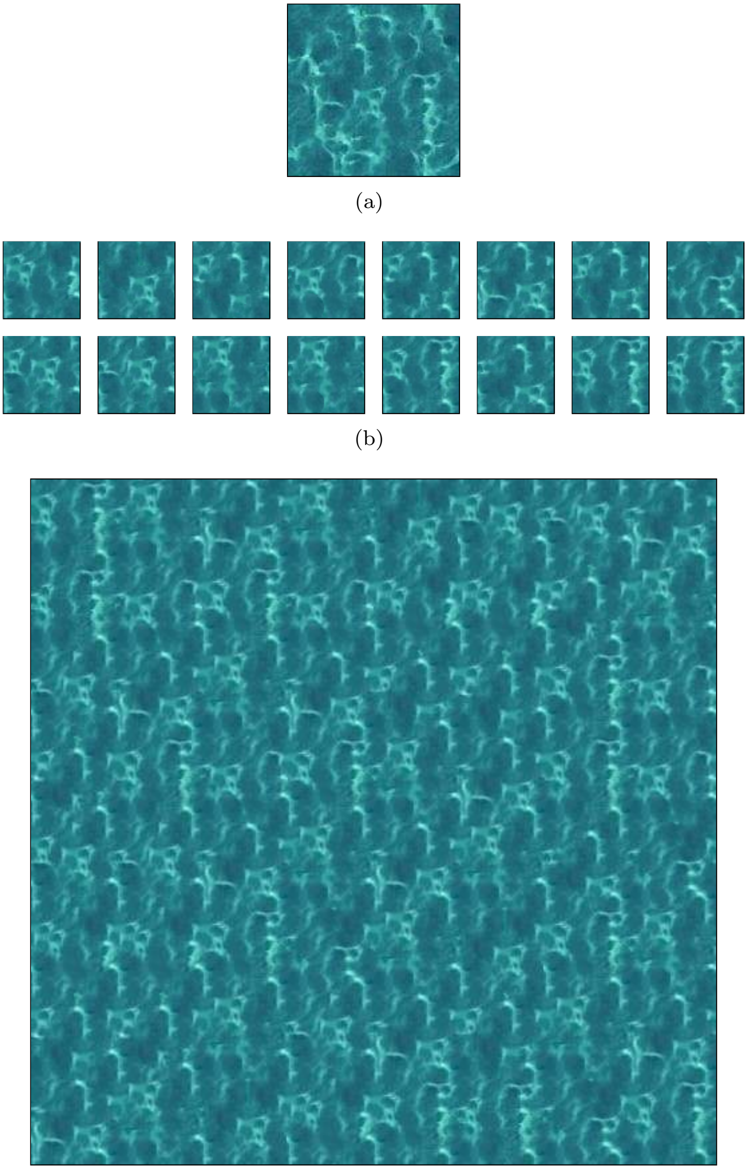
creates from an example texture a new, usually larger texture that appears to be generated by the same underlying process.

Texture synthesis has become a popular area of research within computer graphics, and a complete survey of related work is beyond the scope of this dissertation. For an overview, we refer to Liu et al. [2004] and Kwatra et al. [2005]. Most techniques for texture synthesis are region growing methods, such as pixel-based texture synthesis [Bonet, 1997; Efros and Leung, 1999; Wei and Levoy, 2000] and patch-based texture synthesis [Efros and Freeman, 2001; Liang et al., 2001; Cohen et al., 2003; Kwatra et al., 2003], or global methods [Heeger and Bergen, 1995; Kwatra et al., 2005]. Tile-based texture synthesis can be classified as a patch-based method.

5.3 Tile-Based Texture Synthesis

Stam [1997] was the first to consider Wang tiles in the context of texturing. Stam used Wang tiles to generate non-repeating procedural textures of arbitrary size. A method similar in spirit was presented by Neyret and Cani [1999]. They used triangular tiles with edge and corner colors to generate pattern-based textures over a triangle mesh. Cohen et al. [2003] combined Wang tiles with texture synthesis, and presented a method for synthesizing an example texture over a set of Wang tiles. A variation on the technique of Cohen et al. was proposed by Burke. The method of Cohen et al. was also used by Wei [2004], who presented a texture mapping algorithm based on Wang tiles. Fu and Leung [2005] extended texture tiling to arbitrary topological surfaces. The method of Cohen et al. was extended to corner tiles by Ng et al. [2005].

We use the method of Ng et al. for synthesizing an example texture over a set of corner tiles. This is illustrated in figure 5.1. For each corner color, a square



(a)

(b)

(c)

Figure 5.2: Texture synthesis with texture tiles based on corner tiles. (a) The example texture. (b) A set of texture tiles based on corner tiles constructed from the example texture. (c) A new texture synthesized with the texture tile set.

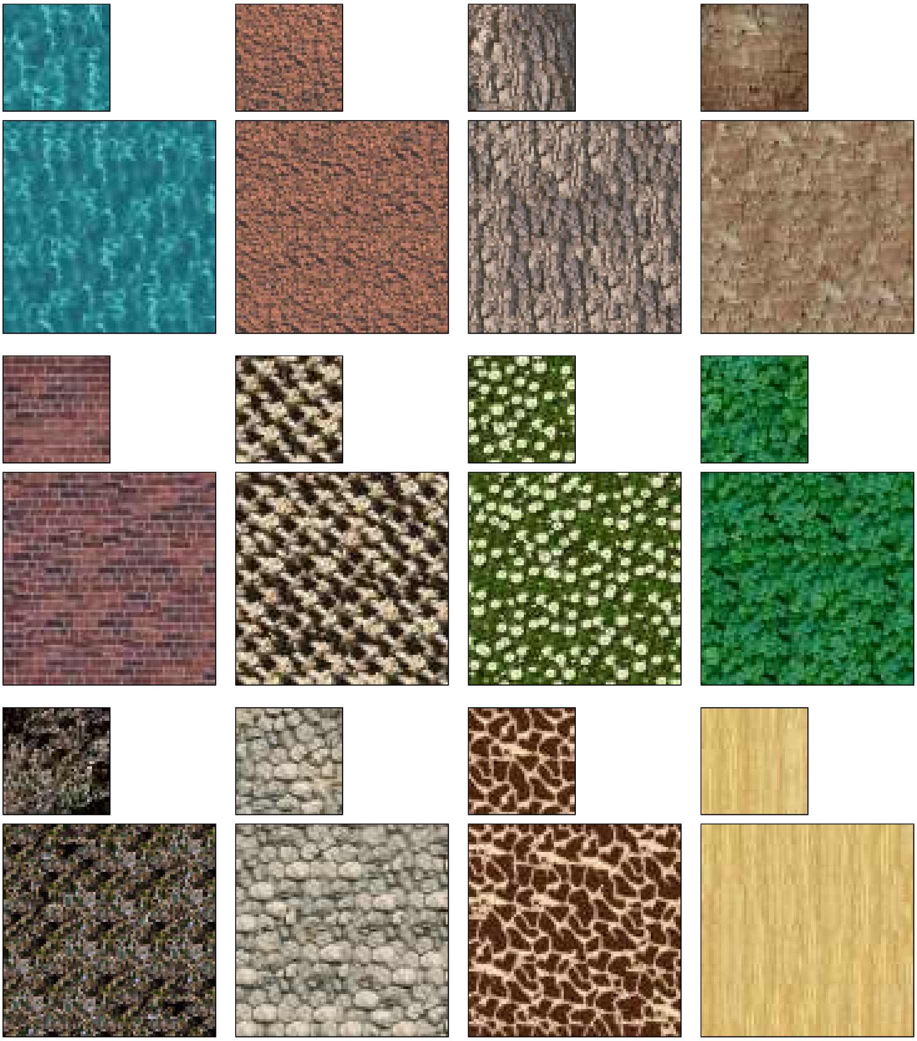


Figure 5.3: Textures synthesized with texture tiles based on corner tiles. These textures are synthesized by tiling 4 × 4 tiles from a complete texture tile set based on corner tiles over two or three colors.

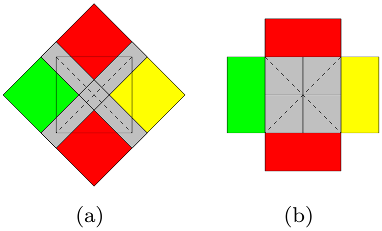


Figure 5.4: Patch combination strategies for texture tiles based on Wang tiles. (a) The method of Cohen et al. (b) A variant introduced by Burke.

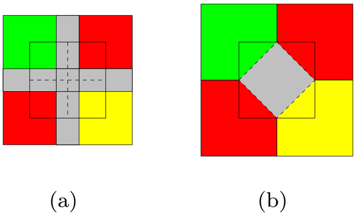


Figure 5.5: Patch combination strategies for texture tiles based on corner tiles. (a) A straightforward extension of the method of Cohen et al. for Wang tiles to corner tiles. (b) The method of Ng et al. Note that this is the only method that adds a new texture patch to each tile.

patch is chosen at random from the example texture (see figure 5.1(a)). Each tile of the tile set is constructed by combining the four patches corresponding to the corner colors (see figure 5.1(b)), and cutting out the tile (see figure 5.1(c)). This leaves a cross shaped seam that is covered with a new diamond-shaped irregular patch from the example texture (see figure 5.1(d)). This patch is optimized using the graph cut method [Kwatra et al., 2003], and is restricted to lie in the circle inscribed in the tile. After an example texture is synthesized over a set of corner tiles, a new texture can be synthesized by generating a tiling. The process of tile-based texture synthesis is illustrated in figure 5.2. The method of Ng et al. is simple and works well. Figure 5.3 shows several textures synthesized with corner-based texture tiles. The quality of the synthesized textures is similar to that of other patch-based techniques.

Cohen et al. [2003] were the first to synthesize an example texture over a set of Wang tiles. Several variations on the method of Cohen et al. have been proposed, and the technique of Ng et al. for corner tiles is based on the method of Cohen et al. These methods only differ in how the patches are placed on the tile. Figures 5.4 and 5.5 show several patch combination strategies for Wang tiles and corner tiles.

The advantage of corner tiles over Wang tiles is less pronounced in texture tile construction. Unwanted artifacts in the synthesized textures are typically located where patches meet. This is at the corners for Wang tiles and in the middle of the edges for corner tiles. In this respect, Wang tiles and corner tiles are similar. However, textures synthesized with corner tiles are usually more similar to the example texture than textures synthesized with Wang tiles. This is because the center of each corner tile is covered with a new irregular patch from the example texture. Therefore, each corner tile contains potentially unique texture samples from the example texture. Other patch combination strategies use for each tile only the patches corresponding to the corner or edge colors. We refer to Ng et al. [2005] for a more detailed comparison.

An important advantage of tile-based texture synthesis is that the process of texture synthesis is broken up into two parts. Once an example texture is synthesized over a set of tiles, arbitrary large textures can be synthesized very efficiently simply by generating stochastic tilings.

5.4 Tile-Based Texture Mapping

Interactive applications in computer graphics harness the power of graphics hardware to guarantee interactive frame rates. Texture mapping is a fundamental feature for these applications. However, texture memory is a scarce resource on graphics hardware, and storing and managing large textures is challenging. All too often, tileable textures are used to save texture memory. However, this results in visually disturbing repetition. In 2004, Wei presented a texture mapping algorithm based on Wang tiles to overcome this problem. In

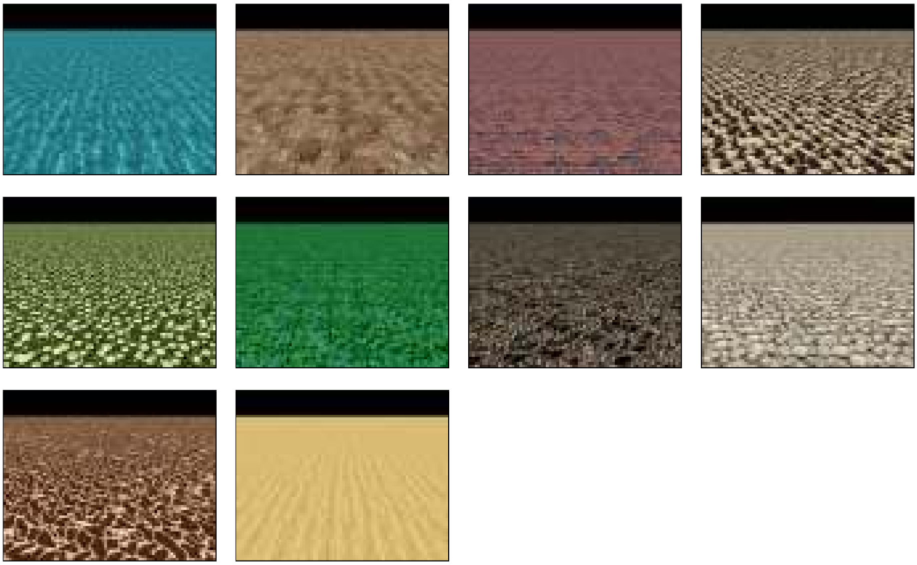


Figure 5.6: Tile-based texture mapping using corner tiles. Screenshots from our interactive tile-based texture mapping application based on corner tiles. Texture filtering does not introduce unwanted artifacts because a tile packing was used.

2006, we presented a cleaner and more efficient variant version of this algorithm based on corner tiles [Lagae and Dutré, 2006a]. In this section we discuss these tile-based texture mapping algorithms in detail.

Tile-based texture mapping uses an example texture synthesized over a set of Wang tiles or corner tiles to simulate an arbitrary large non-periodic texture. This is more complicated than it seems at first sight, because graphics hardware is very specialized and the graphics processing unit (GPU) is a stream processing architecture.

The number of texture units on a GPU is typically small, and a GPU generally prefers square textures. Therefore, all tiles of the tile set must be packed into a single square texture. Tile-based texture mapping algorithms typically use complete tile sets. This is because the C^4 tiles of a complete tile set over C colors can easily be arranged into a square texture using a $C^2 \times C^2$ configuration. However, in order to avoid unwanted discontinuity artifacts introduced by texture filtering, this configuration must also be a valid tiling. This is because texture sampling uses texels from adjacent tiles. Note that the borders of a texture are treated toroidally. For more details, we refer to Wei [2004].

An arrangement of the C^4 tiles of a complete set of Wang tiles or corner tiles over C colors into a $C^2 \times C^2$ toroidal configuration such that adjoining edges or corners have matching colors is called a tile packing. Tile packings are discussed in detail in section 5.5.

The tile-based texture mapping algorithm runs as a fragment program on the GPU. This fragment program transforms texture coordinates in the arbitrary large non-periodic texture to texture coordinates in the texture containing the texture tiles. A tiling is imposed on the arbitrary large non-periodic texture. For each incoming fragment, the tile coordinates and the coordinates within the tile are computed. A direct stochastic tiling algorithm is used to determine the tile at these coordinates. The tile packing provides the location of that tile in the texture containing the texture tiles. This location is combined with the coordinates of the fragment within the tile, and the texture lookup is performed. For more implementation details we refer to Wei [2004] and Lefebvre and Neyret [2003].

A Wang tile packing can be formulated as a closed-form expression [Wei, 2004]. This expression is evaluated directly in the fragment program. However, this is not the case for a corner tile packing (see section 5.5). Therefore, the corner tile packing is stored explicitely, as a constant array in the fragment program, or as an additional texture. The permutation table used by the hash function of the direct stochastic tiling algorithm is stored in the same way.

To avoid the corner problem, the tile-based texture mapping algorithm of Wei requires a second Wang tile packing that contains all possible corner configurations of the Wang tile set. This additional texture is used for texture lookups close to tile corners. Because corner tiles are not subject to the corner problem, only the texture containing the tile packing is needed. Compared to the original method of Wei, our tile-based texture mapping algorithm based on

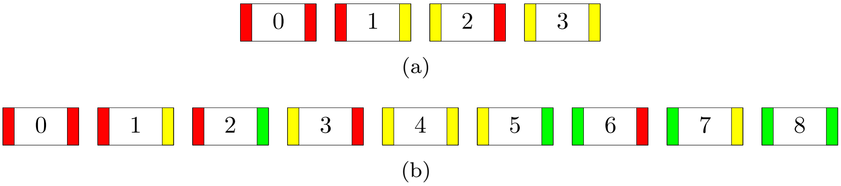


Figure 5.7: The complete 1D Wang or corner tile sets over (a) 2 and (b) 3 colors.

corner tiles reduces the required texture memory by a factor of two and saves one texture unit. This is an important advantage, as reducing texture memory usage is the main goal of tile-based texture mapping. Our algorithm also runs faster, because the tiling algorithm for corner tiles is simpler and more efficient than equivalent algorithms for Wang tiles. Corner tiles reduce the cost of tile-based texture mapping almost to that of regular texture mapping. This is a significant saving for interactive applications.

Our tile-based texture mapping algorithm based on corner tiles runs at several hundred frames per second on a NVidia GeForce 7800 GTX graphics card. Figure 5.6 shows several results.

5.5 The Tile Packing Problem

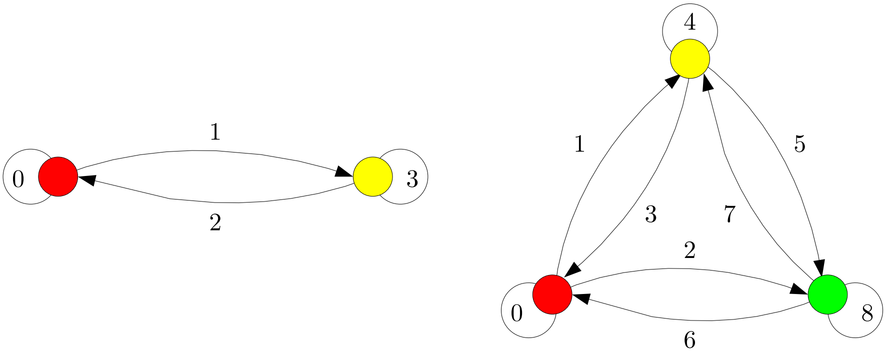
A tile packing is an essential ingredient of the tile-based texture mapping algorithms discussed in section 5.4. A tile packing is used to avoid unwanted texture filtering artifacts. In this section we discuss the problem of computing a tile packing of a complete set of Wang tiles or corner tiles. We also show that the tile packing problem is an interesting combinatorial problem.

The tile packing problem consists of arranging the C^4 tiles of a complete set of Wang tiles or corner tiles over C colors into a $C^2 \times C^2$ toroidal configuration such that adjoining edges or corners have matching colors.

In 2006, we studied the corner tile packing problem in detail [Lagae and Dutré, 2006a,f].

5.5.1 The One-Dimensional Tile Packing Problem

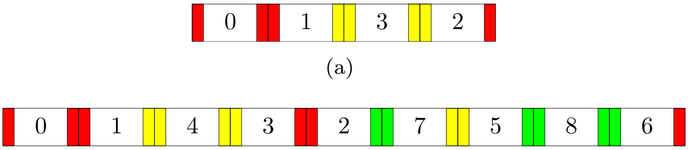
In one dimension, Wang tiles and corner tiles can be seen as dominoes. A complete set of Wang tiles or corner tiles over C colors counts C^2 tiles. Figure 5.7 shows the complete set of tiles over 2 and 3 colors. The one-dimensional tile packing problem consists of arranging a complete set of C^2 tiles into a single circular train. Domino problems like this one are well known in the field of recreational mathematics. A solution based on graph theory is given in the classic work *Mathematical Recreations and Essays* [Ball, 1926].



(a)

(b)

Figure 5.8: Graphs representing the complete 1D Wang or corner tile set over (a) 2 and (b) 3 colors.



(a)

(b)

Figure 5.9: Tile packings of the complete 1D Wang or corner tile set over 2 and 3 colors.

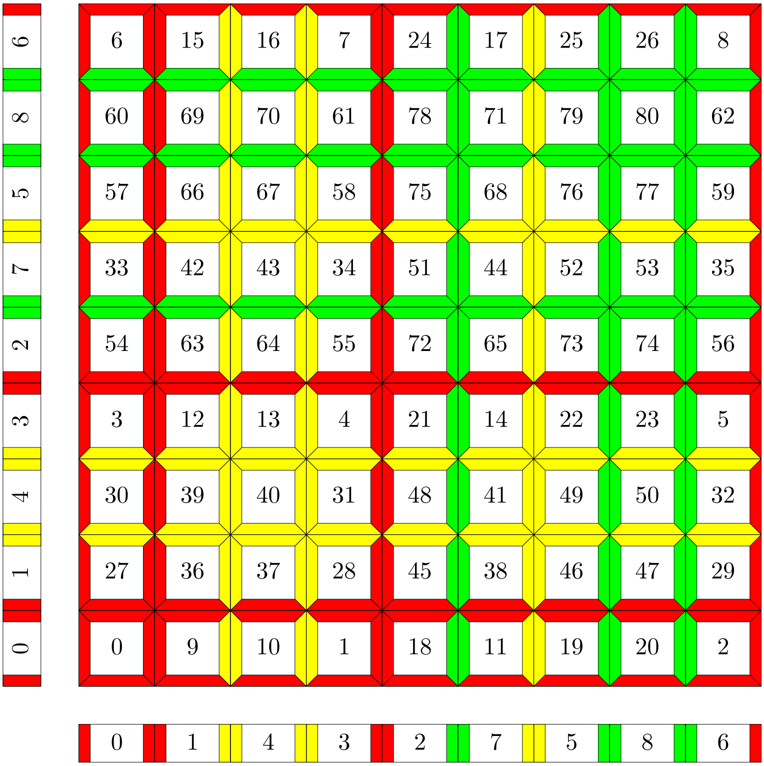


Figure 5.10: The construction of an Eulerian tile packing of the complete Wang tile set over 3 colors.

The tile set is represented as a directed graph with a vertex for each color, and an edge connecting two vertices for each tile in the tile set. Figure 5.8 shows the graphs for the complete tile sets over 2 and 3 colors. A solution for the tile packing problem is given by an Eulerian circuit, a graph cycle that uses each graph edge exactly once. A complete tile set results in a complete directed graph, which always has an Eulerian circuit. Figure 5.9 shows tile packings of the complete tile sets over 2 and 3 colors. A tile packing obtained with this method is called an Eulerian Wang tile packing.

Wei [2004] presented a closed-form expression that gives the position of a specific tile in a one-dimensional Eulerian tile packing.

5.5.2 The Wang Tile Packing Problem

Wei [2004] observed that Wang tiles are separable and that a solution for the two-dimensional Wang tile packing problem is given by the outer product of two one-dimensional tile packings. This is illustrated in figure 5.10.

A one-dimensional tile packing of a complete set over C colors consist of C^2 tiles. The outer product of two such tile packings produces a matrix of C^4 tiles. Because adjoining edges have matching colors, and each tile occurs exactly once, this is a tile packing of the C^4 tiles of a complete Wang tile set over C colors. This construction method generalizes to tile packings of Wang tiles in any dimension.

A closed-form expression that gives the position of a specific tile in an Eulerian Wang tile packing is obtained by applying the closed-form expression for the one-dimensional case for each dimension.

5.5.3 The Corner Tile Packing Problem

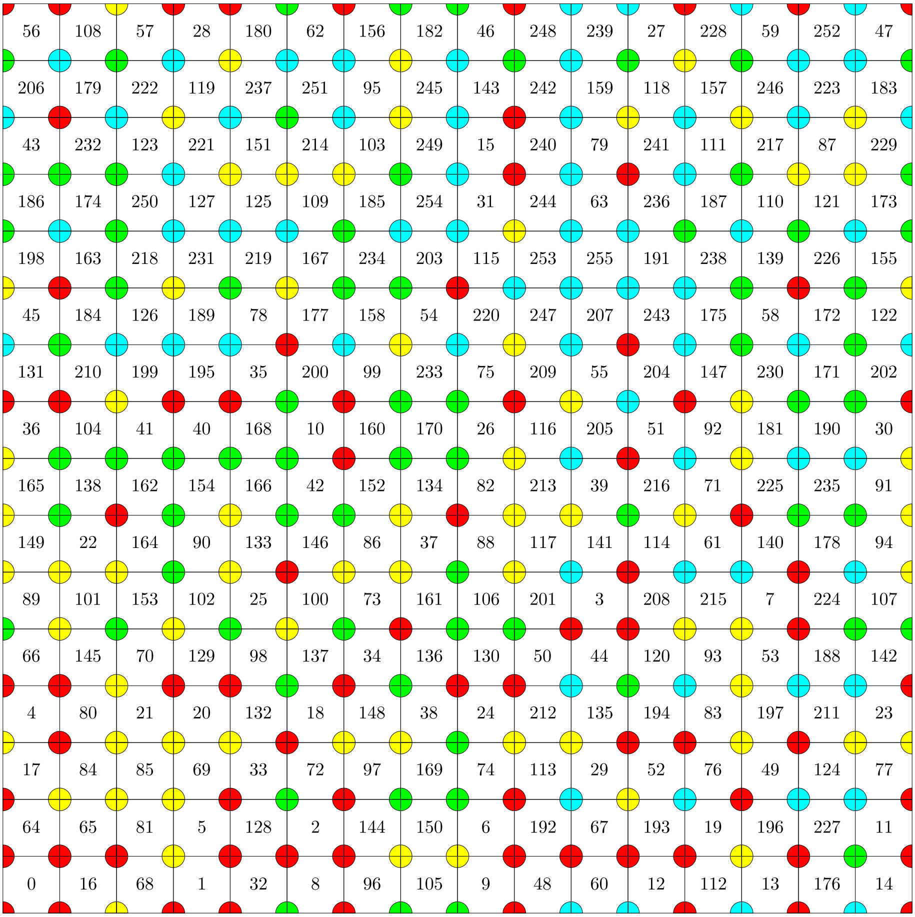
Although tiles with colored edges and problems similar to the Wang tile packing problem were studied before in the field of recreational mathematics [MacMahon, 1921], corner tiles and the corner tile packing problem have not been examined. The method for constructing a Wang tile packing also does not seem to extend to corner tiles.

When we started exploring the corner tile packing problem, it resisted all attempts to solve it. It was not clear whether the corner tile packing problem even had a solution. We therefore decided to tackle the problem using combinatorial search methods.

A simple exhaustive search or generate-and-test method is not practical. For C colors the tiles can be arranged in $C^4!$ ways. For 2, 3 and 4 colors, this equals approximately 2.09×10^{13}, 5.80×10^{120} and 8.58×10^{506}. Instead we use a backtracking method, that places one tile at a time until a dead end is reached, at which point previous steps are retraced. Backtracking greatly reduces the amount of work in an exhaustive search, and is often used to solve hard combinatorial problems such as the knights tour problem and the queens problem [Ball, 1926]. The algorithm can also be used to search for solutions to the Wang tile packing problem.

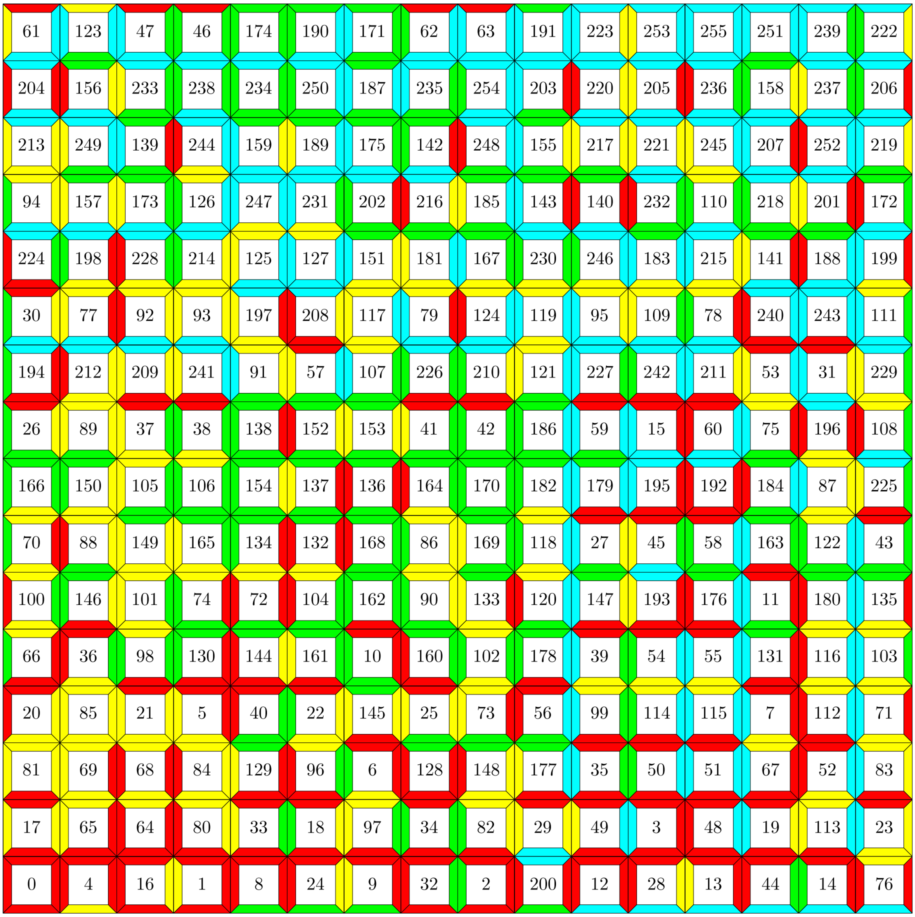
Although backtracking is relatively fast compared to simple exhaustive search and generate-and-test methods, the time needed to solve the tile packing problem is still large. Therefore, our backtracking algorithm also supports parallelization, checkpointing and progress estimation. For implementation details, we refer to [Lagae and Dutré, 2006f].

With the backtracking algorithm, we are able to compute Wang and corner tile packings for 2, 3 and 4 colors. For 2 colors, all solutions of the Wang and corner tile packing problem are obtained almost immediately on a regular desktop computer. The corner tile packing problem has 32 solutions and the Wang tile packing problem has $203,520$ solutions. This supports the claim that in some way, corner tile packings are more difficult to construct than Wang tile packings. For 3 colors, the first solution of the Wang and corner tile packing problems is obtained almost immediately, but computing or counting



56	108	57	28	180	62	156	182	46	248	239	27	228	59	252	47
206	179	222	119	237	251	95	245	143	242	159	118	157	246	223	183
43	232	123	221	151	214	103	249	15	240	79	241	111	217	87	229
186	174	250	127	125	109	185	254	31	244	63	236	187	110	121	173
198	163	218	231	219	167	234	203	115	253	255	191	238	139	226	155
45	184	126	189	78	177	158	54	220	247	207	243	175	58	172	122
131	210	199	195	35	200	99	233	75	209	55	204	147	230	171	202
36	104	41	40	168	10	160	170	26	116	205	51	92	181	190	30
165	138	162	154	166	42	152	134	82	213	39	216	71	225	235	91
149	22	164	90	133	146	86	37	88	117	141	114	61	140	178	94
89	101	153	102	25	100	73	161	106	201	3	208	215	7	224	107
66	145	70	129	98	137	34	136	130	50	44	120	93	53	188	142
4	80	21	20	132	18	148	38	24	212	135	194	83	197	211	23
17	84	85	69	33	72	97	169	74	113	29	52	76	49	124	77
64	65	81	5	128	2	144	150	6	192	67	193	19	196	227	11
0	16	68	1	32	8	96	105	9	48	60	12	112	13	176	14

Figure 5.11: A recursive tile packing of the complete corner tile set over 4 colors.



61	123	47	46	174	190	171	62	63	191	223	253	255	251	239	222
204	156	233	238	234	250	187	235	254	203	220	205	236	158	237	206
213	249	139	244	159	189	175	142	248	155	217	221	245	207	252	219
94	157	173	126	247	231	202	216	185	143	140	232	110	218	201	172
224	198	228	214	125	127	151	181	167	230	246	183	215	141	188	199
30	77	92	93	197	208	117	79	124	119	95	109	78	240	243	111
194	212	209	241	91	57	107	226	210	121	227	242	211	53	31	229
26	89	37	38	138	152	153	41	42	186	59	15	60	75	196	108
166	150	105	106	154	137	136	164	170	182	179	195	192	184	87	225
70	88	149	165	134	132	168	86	169	118	27	45	58	163	122	43
100	146	101	74	72	104	162	90	133	120	147	193	176	11	180	135
66	36	98	130	144	161	10	160	102	178	39	54	55	131	116	103
20	85	21	5	40	22	145	25	73	56	99	114	115	7	112	71
81	69	68	84	129	96	6	128	148	177	35	50	51	67	52	83
17	65	64	80	33	18	97	34	82	29	49	3	48	19	113	23
0	4	16	1	8	24	9	32	2	200	12	28	13	44	14	76

Figure 5.12: A recursive tile packing of the complete Wang tile set over 4 colors.

all solutions seems to be hopeless. For 4 colors, computing a corner tile packing took 280 days of CPU time, and it took roughly 23 years of CPU time to find the first solution of the Wang tile packing problem. These last results were obtained using a parallel version of our backtracking algorithm, running on a cluster with almost 400 2.4 GHz CPU's.

A solution for C colors can often be found faster by starting from a solution of $C - 1$ colors. That way, a recursive tile packing is obtained. Figures 5.12 and 5.11 show recursive Wang and corner tile packings for 4 colors.

The tile packing problem has many symmetries. New solutions can be obtained from existing ones using translation (the tile packing is toroidal), rotation, reflection, and permutation of the colors. The 32 solutions of the 2 color corner tile packing problem reduce to a single fundamental solution.

There is still room for improving the backtracking algorithm. The many symmetries of the tile packing problem are currently not exploited. To solve the corner tile packing problem we have also experimented with a backtracking algorithm that places colored pegs rather than tiles. This algorithm seems to be faster.

The tile packing problem is an interesting combinatorial puzzle. We were able to obtain Wang and corner tile packings for up to 4 colors. However, several problems, such as counting the number of solutions of the tile packing problem, and finding a constructive method and a closed-form expression for the corner tile packing problem, remain unsolved.

5.5.4 Puzzles Derived from the Tile Packing Problem

The work most closely related to the tile packing problem in the field of recreational mathematics is that of MacMahon [1921]. He describes sets of pieces of different geometrical forms (including equilateral triangles, squares and pentagons) with colored edges that are tiled into another geometrical form. The profile of the adjoining edges is then altered to produce jigsaw puzzles. His work is unique in the fact that it details how the puzzles can be constructed and solved. In contrast with Wang and corner tiles, the pieces of MacMahon pieces may be rotated. MacMahon also does not consider pieces with colored corners.

Inspired by the work of MacMahon, we have created jigsaw puzzles from tile packings by altering the profile of the adjoining edges or corners. Figure 5.13 shows two examples. To create interesting puzzles, it is better to use tile packings constructed with the backtracking algorithm instead of Eulerian tile packings. We found it already hard to construct a tile packing of the complete set of corner tiles over 2 colors by hand, so these puzzles should be challenging, especially puzzles based on tile packings of 3 or 4 colors. To prevent the tiles from being rotated, a picture could be printed on the puzzle.

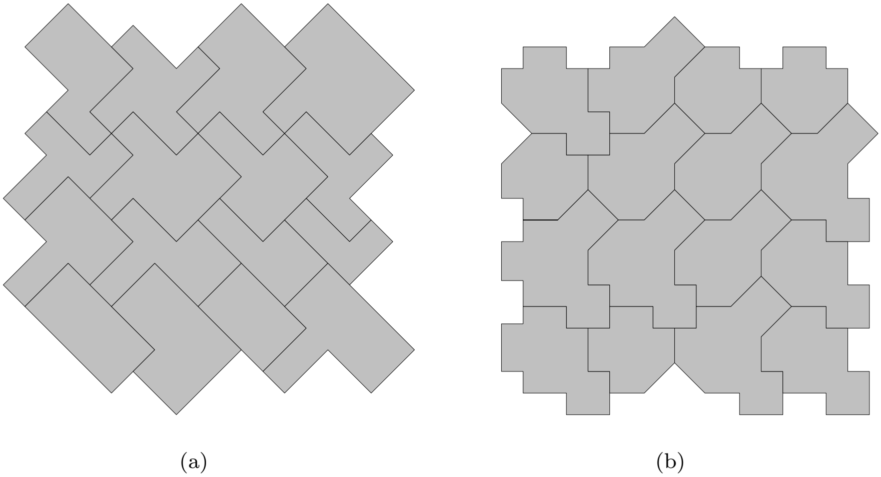


Figure 5.13: Jigsaw puzzles derived from tile packings of the complete (a) Wang and (b) corner tile set over 2 colors.

5.6 Conclusion

In this chapter we have investigated the use of Wang tiles and corner tiles in texture mapping and texture synthesis. We have introduced texture mapping and texture synthesis. We have discussed a method for synthesizing a texture over a set of Wang tiles or corner tiles. We have presented a tile-based texture mapping algorithm, and we have discussed the tile packing problem. For tile-based texture mapping, corner tiles are clearly superior to Wang tiles. For tile-based texture synthesis corner tiles are also better than Wang tiles, but the difference is less pronounced. Tile packings are more difficult to compute for corner tiles than for Wang tiles. However, a tile packing has to be computed only once.

Chapter 6

A Comparison of Methods for Generating Poisson Disk Distributions

6.1 Introduction

Poisson disk distributions have several applications in the field of computer graphics. Over the years, a large number of methods for generating Poisson disk distributions have been proposed. These methods, and the distributions they generate, often have very different characteristics. This makes it difficult to choose the right method for a given application. In this chapter, we present a detailed comparison of methods for generating Poisson disk distributions, including the tile-based methods presented in chapter 4.

Overview

This chapter is organized as follows. In section 6.2 we introduce the methodology used to analyze Poisson disk distributions. The next sections analyze several methods for generating Poisson disk distributions. The methods we study are dart throwing (section 6.3), relaxation dart throwing (section 6.4), Lloyd's relaxation method (section 6.5), Shade's Poisson disk tiles (section 6.6), tiled blue noise samples (section 6.7), fast hierarchical importance sampling with blue noise properties (section 6.8), edge-based Poisson disk tiles (section 6.9), template Poisson disk tiles (section 6.10), corner-based Poisson disk tiles (section 6.11), efficient generation of Poisson disk sampling patterns (section 6.12), a spatial data structure for fast Poisson disk sample generation (section 6.13), and recursive Wang tiles for real time blue noise (section 6.14). In section 6.15 we analyze several methods for generating Poisson sphere distributions. In section 6.16 we conclude.

6.2 Methodology

Poisson disk distributions have several applications in the field of computer graphics. These applications are often very different. Therefore, choosing a single method for analyzing Poisson disk distributions is difficult. Instead, we use a set of methods that cover a wide range of applications. These methods are radius analysis, spectral analysis, sampling performance, and timings. In this section, we discuss these methods for analyzing Poisson disk distributions in detail.

6.2.1 Radius Analysis

The most obvious way for analyzing a Poisson disk distribution is examining the radius of the distribution. Because the radius of a Poisson disk distribution determines how well the points are distributed, it is an important measure for the quality of the distribution.

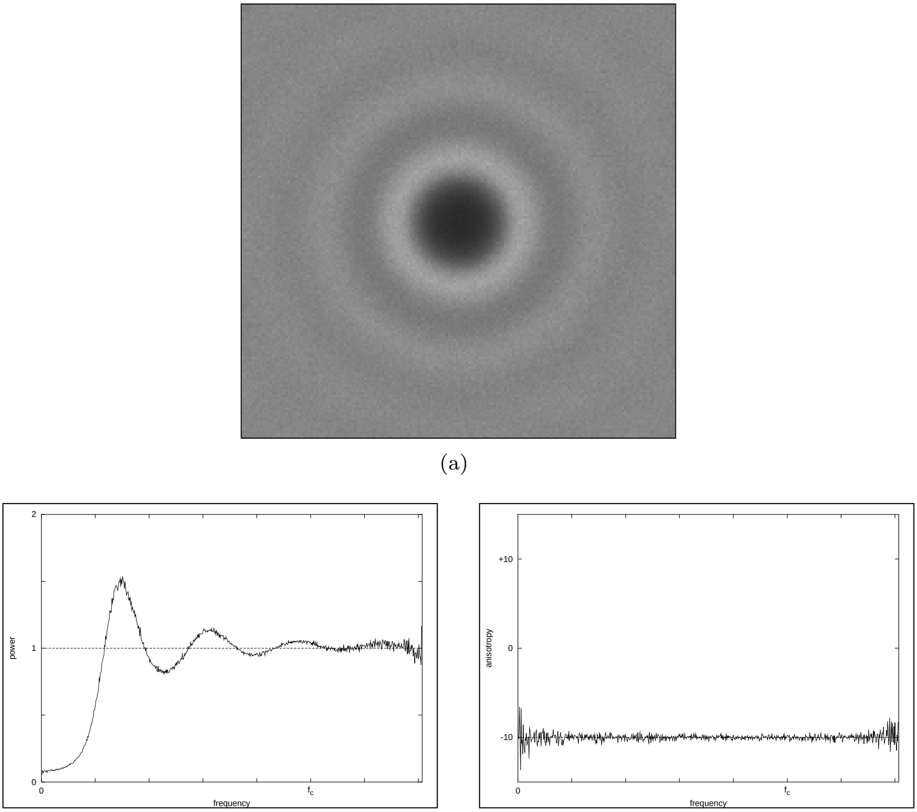
We use the relative radius specification scheme introduced in section 4.2.3. The absolute radius r of a Poisson disk distribution is expressed as a relative radius ρ. The relative radius ρ is a fraction of the maximum radius r_{max} that can be achieved. The relative radius should be large ($\rho \geq 0.65$), but not too large ($\rho \leq 0.85$), because regular configurations must be avoided.

6.2.2 Spectral Analysis

In computer graphics, continuous functions such as images are stored as a collection of samples. This discrete representation of functions causes unwanted side effects such as jaggies, motion strobing, moiré patterns and popping, also known as aliasing [Crow, 1977]. Aliasing can be traded for noise by using stochastic sampling patterns rather than regular patterns [Dippé and Wold, 1985; Cook, 1986; Mitchell, 1987]. In computer graphics, noise is preferred over aliasing because the human visual system is quite tolerant to noise, while structured aliasing artifacts are spotted easily. For more information about digital signal processing in computer graphics, we refer to Glassner [1995].

In the early eighties, Yellot [1982, 1983] observed hat the photoreceptors in the retina are placed according to a Poisson disk distribution. This was an indication that the Poisson disk sampling pattern is effective for imaging. Yellot showed that the least noticeable form of aliasing occurs if the power spectrum of the sampling pattern is noisy and lacks concentrated spikes of energy, and contains no low-frequency energy. The absence of concentrated spikes prevents structured aliasing while the absence of low-frequency energy pushes aliasing noise to less conspicuous higher frequencies. A power spectrum with these properties is called a blue noise power spectrum.

Analyzing a Poisson disk distribution in the frequency domain is also interesting because it might reveal information that is not obvious in the spatial

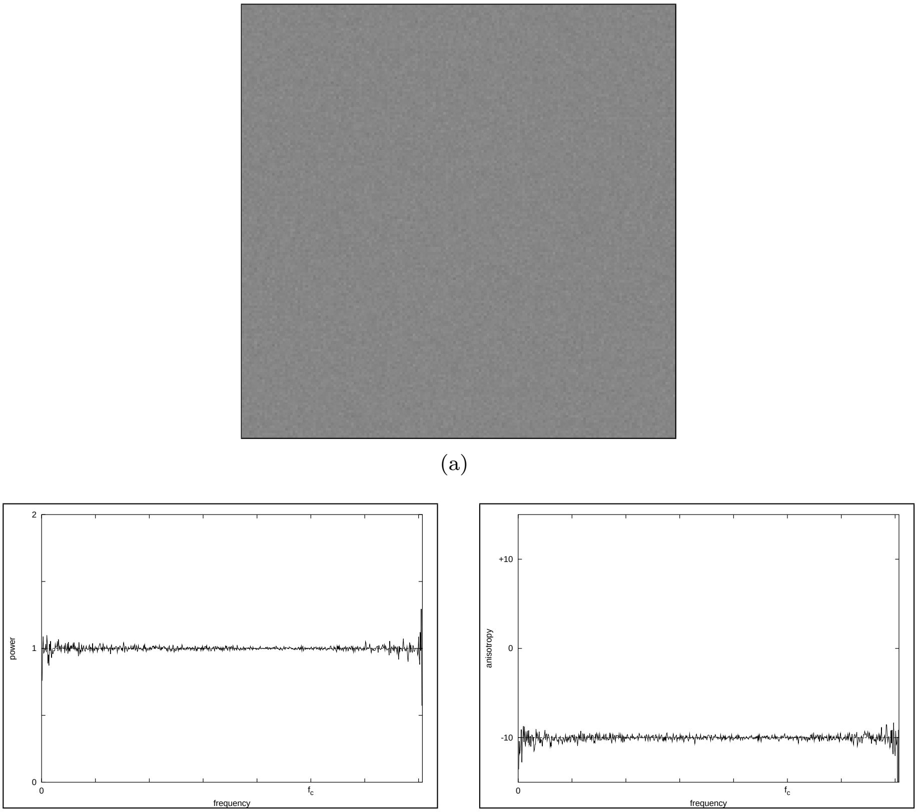


(a)

(b)

(c)

Figure 6.1: The typical power spectrum of Poisson disk distributions. (a) The power spectrum. (b) The radially averaged power spectrum. (c) The anisotropy.



(a)

(b)

(c)

Figure 6.2: The typical power spectrum of Poisson distributions. (a) The power spectrum. (b) The radially averaged power spectrum. (c) The anisotropy.

domain.

To estimate the power spectrum of a Poisson disk distribution we use the method of averaging periodograms of Bartlett [1955]. This technique was first used by Ulichney [1987] to study dither patterns. The periodogram of a Poisson disk distribution of N points $\{x_0, x_1, \ldots, x_{N-1}\} \subset [0,1[^2$ is the magnitude squared of the Fourier transform of the distribution

$$\left| \mathcal{F} \frac{1}{N} \sum_{j=0}^{N-1} \delta\left(x - x_j\right) \right|^2, \tag{6.1}$$

where $\mathcal{F}$ denotes the Fourier transform and δ is Dirac's delta function. By averaging K periodograms, an estimate $\hat{P}(f)$ of the power spectrum $P(f)$ is obtained with variance

$$var\left\{\hat{P}(f)\right\} \approx \frac{1}{K} P^2(f). \tag{6.2}$$

Thus, the spectral properties of a method for generating Poisson disk distributions are obtained by averaging periodograms from Poisson disk distributions generated with that method. It is important to note that a periodogram is associated with a single Poisson disk distribution, while the power spectrum estimate is associated with a specific method for generating Poisson disk distributions, for example a specific algorithm with a fixed set of parameters.

The periodogram of a Poisson disk distribution is radially symmetric. Therefore, two one-dimensional statistics are derived from the power spectrum. The first one is the radially averaged power spectrum

$$P_r\left(f_r\right) = \frac{1}{N_r\left(f_r\right)} \sum_{i=1}^{N_r(f_r)} \hat{P}\left(f\right), \tag{6.3}$$

which is obtained by averaging $\hat{P}(f)$ in concentric annuli of width Δ. Each annulus has a central radius f_r and contains $N_r\left(f_r\right)$ frequency samples. The second statistic is the anisotropy

$$A_r\left(f_r\right) = \frac{s^2\left(f_r\right)}{P_r^2\left(f_r\right)}, \tag{6.4}$$

where the sample variance of the frequency samples is defined as

$$s^2\left(f_r\right) = \frac{1}{N_r\left(f_r\right) - 1} \sum_{i=1}^{N_r(f_r)} \left(\hat{P}\left(f\right) - P_r\left(f_r\right)\right)^2. \tag{6.5}$$

The anisotropy is a measure for the radial symmetry of the power spectrum.

As in the work of Ulichney, the spectral estimates in this paper were produced by averaging $K = 10$ periodograms. Therefore, an anisotropy of $-10\, dB$ should be considered background noise. A reference line at $-10\,dB$ appears in all anisotropy plots.

All spectral estimates in this paper were obtained by analyzing distributions of $16,384$ (or as close as possible to $16,384$) points. The periodogram resolution used to analyze a distribution of N points is determined as

$$\frac{1}{0.75\,r_{max}}. \tag{6.6}$$

This ensures that the periodogram covers a domain corresponding to twice the principal frequency of a Poisson disk distribution with a relative radius of 0.75 (634×634 for $16,384$ points). This number is rounded to the next power of two ($1,024 \times 1,024$ for $16,384$ points). Thus, all power spectra and plots are at the same scale. The width of the annuli Δ is one sample.

All power spectrum images were tone mapped with a logarithmic tone mapper, using the same settings for all images. The radially averaged power spectrum was normalized. The high magnitude DC peak was removed from all plots.

Figure 6.1 shows the typical power spectrum of a Poisson disk distribution. The corresponding radially averaged power spectrum and anisotropy are also shown. The power spectrum reveals the typical blue noise properties. The central DC peak is surrounded by an annulus of low energy, followed by a sharp transition region, a low frequency cutoff at the principal frequency $1/2r$, and a flatter high frequency region. Figure 6.2 shows the typical power spectrum and radially averaged power spectrum and anisotropy of a Poisson distribution. The corresponding radially averaged power spectrum and anisotropy are also shown. As expected, the power spectrum is flat. In both cases, the anisotropy is low (close to $-10\,dB$), indicating good radial symmetry.

Comparing periodograms, power spectrum estimates and radially averaged power and anisotropy graphs of different sources is very difficult. Often the distinction between periodogram and power spectrum is not made, periodograms and power spectra are tone mapped with different settings, and power and anisotropy graphs are computed with wide annuli, effectively smoothing the graphs. Analyzing all methods within a single framework is one of the contributions of this dissertation.

6.2.3 Sampling Performance

Since Yellot [1983] presented theoretical evidence in favor of the Poisson disk distribution, Poisson disk distributions are frequently used in the context of sampling. Therefore, the sampling performance of a Poisson disk distribution is an important measure for the quality of the distribution.

The performance of a sampling pattern can be tested by sampling a function using the pattern, reconstructing the function, and inspecting the reconstructed function. When the sampling rate is too low to capture high frequency content

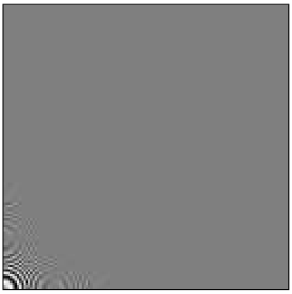
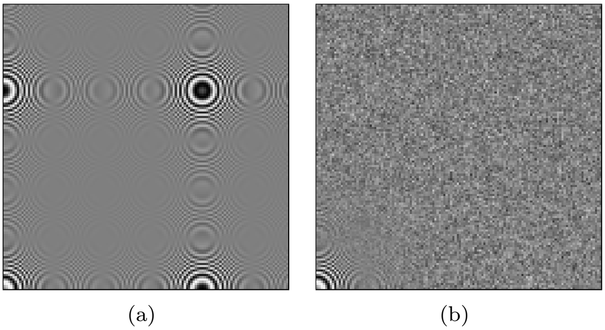


Figure 6.3: The zone plate function used for testing sampling patterns. This reference image was constructed by sampling the zone plate function using a stratified sampling pattern with one million samples per pixel. The image was reconstructed using a Mitchell filter of 4 × 4 pixels.



(a)

(b)

Figure 6.4: The zone plate function sampled with a regular and stratified sampling pattern. (a) A regular sampling pattern with one sample per pixel. (b) a stratified sampling pattern with one sample per pixel. Both images were reconstructed using a Mitchell filter of 4 × 4 pixels.

in the function, aliasing or noise will appear. In this dissertation, the performance of a sampling pattern is tested by sampling and reconstructing the function

$$f(x, y) = \frac{1}{2}\left(1 + \sin\left(x^2 + y^2\right)\right). \tag{6.7}$$

This function is also known as the zone plate function, because it resembles a zone plate device. The frequency of the zone plate function is proportional to the distance from the origin. Therefore, the performance of a sampling pattern is better as aliasing or noise occurs further away from the origin. Noise is preferred over structured aliasing.

The zone plate function is sampled over the domain $[0, 48[^2$, at sample locations determined by the Poisson disk distribution, and reconstructed at a resolution of 512 × 512, using a Mitchell filter of 4 × 4 pixels [Mitchell and Netravali, 1988]. The reconstructed function is visualized as an image, where 0 is mapped to black, 1 to white, and intermediate values are interpolated. The origin is at the lower left corner of the image, the x-axis is pointing to the right, and the y-axis is pointing up. A sampling rate of one sample per pixel is used. This corresponds to a Poisson disk distribution of 262, 144 points.

Figure 6.3 shows the zone plate function sampled using one million random samples per pixel and reconstructed using a Mitchell filter of 4×4 pixels. This image serves as a reference. However, note that some aliasing artifacts are still visible. This is not aliasing caused by undersampling, or prealiasing, but aliasing caused by reconstruction, or postaliasing. In practice, postaliasing is much less severe than prealiasing.

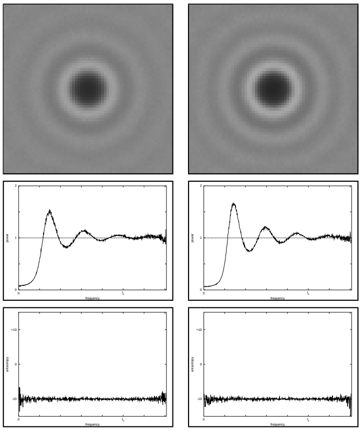
Figure 6.4 shows the zone plate function sampled using a regular sampling pattern and a stratified sampling pattern. The regular sampling pattern causes aliasing, which the stratified sampling pattern trades for noise. In computer graphics, the noisy image is preferred over the image with moiré patterns.

6.2.4 Timings

A final important measure for comparing methods for generating Poisson disk distributions is the time needed to generate a Poisson disk distribution. These timings are an indication of the relative computational cost of the different techniques. The timings were obtained on a regular desktop PC with an AMD Athlon(tm) 64 Processor 4000+ CPU running at 2.4 GHz.

6.3 Dart Throwing

In the mid eighties, Dippé and Wold [1985], Cook [1986] and Mitchell [1987] introduced nonuniform sampling and the Poisson disk distribution to solve the aliasing problem. Cook [1986] proposed the dart throwing algorithm for generating Poisson disk distributions.



(a) $\rho = 0.70$ (b) $\rho = 0.75$

Figure 6.5: Spectral analysis of Poisson disk distributions generated with dart throwing. The Poisson disk distributions have a relative radius of (a) $\rho = 0.70$ and (b) $\rho = 0.75$.

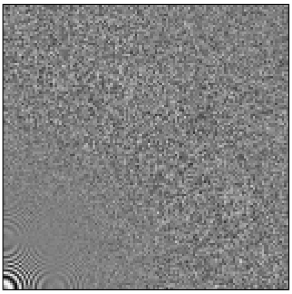


Figure 6.6: The zone plate function sampled with a Poisson disk distribution generated with dart throwing. The Poisson disk distribution has a relative radius of $\rho = 0.70$.

The dart throwing algorithm generates uniformly distributed points, and rejects points that do not satisfy the minimum separation with already generated points. This process continues until no more points can be added. To correctly handle boundary conditions, the Poisson disk distributions are typically generated over a toroidal domain.

Dart throwing is widely used and is easy to implement. However, the algorithm is also slow, and difficult to control. Instead of specifying the number of points, the radius of the distribution has to be provided, the final number of points in the distribution is difficult to predict, and if the process is stopped too soon, the density of the points is not uniform.

The dart throwing algorithm can be improved by combining it with the relative radius specification scheme (see section 4.2.3). The improved dart throwing algorithm takes as input the desired number of points and the relative radius of the Poisson disk distribution. The absolute radius is then calculated. This solves the radius specification problem. Experiments show that dart throwing is capable of generating Poisson disk distributions with a relative radius up to 0.75. Termination however, is still not guaranteed, and generating large Poisson disk distributions ($100,000$ points or more) is very difficult in practice.

Figure 6.5 shows the power spectrum of Poisson disk distributions generated with the dart throwing algorithm. The power spectrum is radially symmetric, the radially averaged power spectrum exhibits the typical blue noise properties, and the anisotropy is low.

Figure 6.6 shows the zone plate function sampled using a Poisson disk distribution generated with dart throwing. Undersampling causes noise rather than aliasing, because the Poisson disk distribution is a stochastic distribution. The zone plate function is reproduced significantly better at higher frequencies than when using a stratified sampling pattern. This illustrates that the Poisson disk distribution is a good sampling pattern.

The time required to generate a Poisson disk distribution of $16,384$ points with a relative radius of 0.70 and 0.75 is respectively 34.997 and 263.492 seconds.

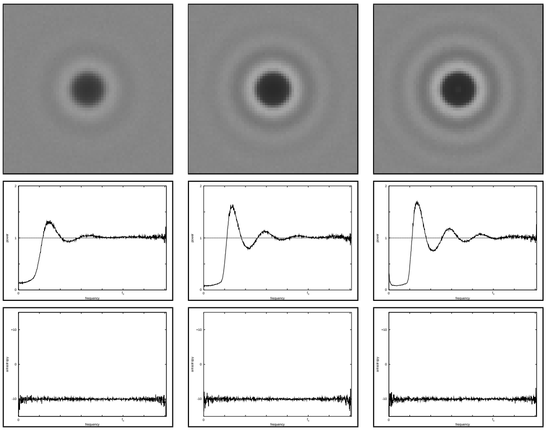
Although the traditional dart throwing algorithm is terribly inefficient, it is the most natural way for generating a Poisson disk distribution. Therefore, we will use the power spectra of figure 6.5 as a reference blue noise power spectrum.

In 2006, Jones and Dunbar and Humphreys presented efficient implementations of the dart throwing algorithm. These algorithms are discussed in section 6.12 and in section 6.13.

6.4 Relaxation Dart Throwing

In 1992, McCool and Fiume proposed an improved version of the dart throwing algorithm, which we call relaxation dart throwing.

Relaxation dart throwing is similar to dart throwing. However, points are



(a) 0.9, 100 (b) 0.99, 1,000 (c) 0.999, 10,000

Figure 6.7: Spectral analysis of Poisson disk distributions generated with relaxation dart throwing. The Poisson disk distributions were generated using an initial radius of 0.15, and the radius was reduced by a factor of (a) 0.9, (b) 0.99 and (c) 0.999 after (a) 100, (b) 1,000 and (c) 10,000 failed attempts.

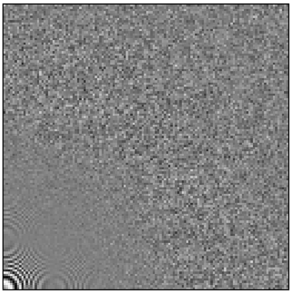


Figure 6.8: The zone plate function sampled with a Poisson disk distribution generated with relaxation dart throwing. The Poisson disk distribution was generated using an initial radius of 0.15, and the radius was reduced by a factor of 0.99 after 1,000 failed attempts.

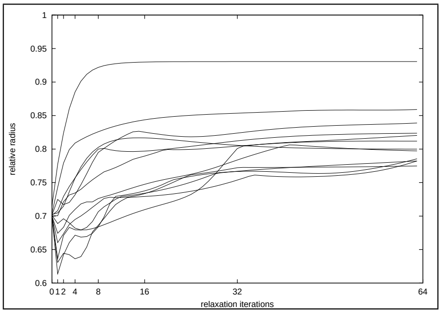


Figure 6.9: The relative radius versus the number of iterations of Lloyd's relaxation method applied to a Poisson disk distribution.

placed with a large radius initially, and once no more space has been found for a large number of attempts, the radius is reduced by some fraction. The Poisson disk distributions generated with relaxation dart throwing are usually toroidal, and also hierarchical.

This algorithm has several advantages compared to dart throwing: it is somewhat faster, it allows to specify the desired number of points rather than the radius, and termination is guaranteed.

The relative radius of Poisson disk distributions of $16,384$ points generated using an initial radius of 0.15, a radius reduction factor of 0.9, 0.99 and 0.999, and 100, $1,000$ and $10,000$ failed attempts is respectively 0.57,0.68 and 0.73.

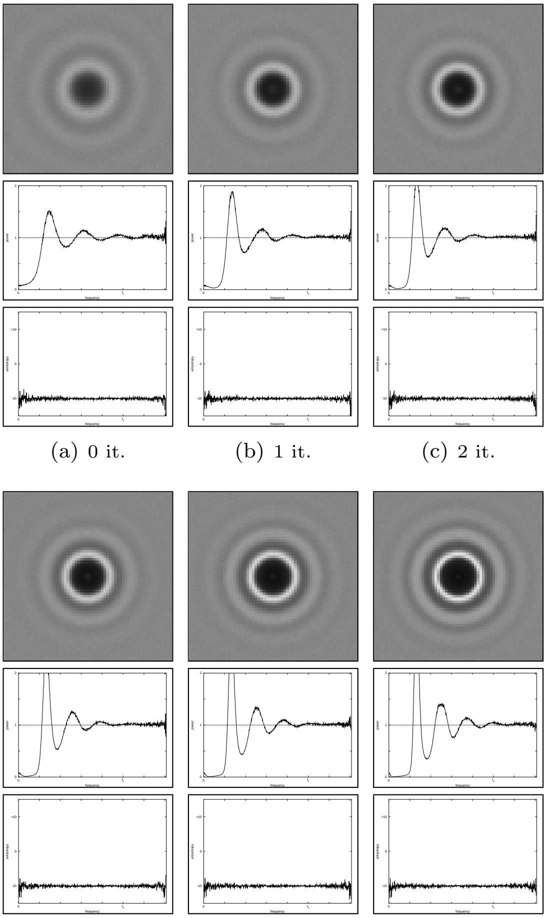
Figure 6.7 shows the power spectrum of Poisson disk distributions generated with the relaxation dart throwing. The spectral characteristics of Poisson disk distributions generated with dart throwing and relaxation dart throwing are very similar. The transition region is a bit steeper for Poisson disk distributions generated with relaxation dart throwing.

Figure 6.8 shows the zone plate function sampled using a Poisson disk distribution generated with relaxation dart throwing. The zone plate function is reproduced even better at higher frequencies than when using dart throwing. This might be explained by the fact that Poisson disk distributions generated with relaxation dart throwing are hierarchical.

The time required to generate Poisson disk distributions of $16,384$ points using an initial radius of 0.15, a radius reduction factor of 0.9, 0.99 and 0.999, and 100, $1,000$ and $10,000$ failed attempts is respectively 0.020, 0.289 and 11.733 seconds.

6.5 Lloyd's Relaxation Scheme

After a Poisson disk distribution is generated using relaxation dart throwing, McCool and Fiume [1992] apply Lloyd's relaxation method [Lloyd, 1982] to



(a) 0 it.

(b) 1 it.

(c) 2 it.

(d) 4 it.

(e) 8 it.

(f) 16 it.

Figure 6.10: Spectral analysis of Poisson disk distributions optimized with Lloyd's relaxation method. The Poisson disk distributions were generated with dart throwing and optimized with (a) 0, (b) 1, (c) 2, (d) 4, (e) 8 and (f) 16 iterations of Lloyd's relaxation method.

increase the radius of the Poisson disk distribution.

Lloyd's relaxation method is an iterative process: in each iteration, the Voronoi diagram of the point set is computed, and each point is moved to the centroid of its Voronoi cell. This process is illustrated in figure 4.2. To correctly handle boundary conditions, Lloyd's relaxation is performed over a toroidal domain.

Lloyd's relaxation can be used as a post-process for increasing the radius of a Poisson disk distribution generated with any technique, or as a standalone technique that generates a Poisson disk distribution from a Poisson distribution.

Figure 6.9 shows the radius of Poisson disk distributions generated with dart throwing versus the number of relaxation iterations. For distributions up to $16,384$ points, Lloyd's relaxation converges in about 16 to 32 iterations. Poisson disk distributions with less points seem to converge to a slightly higher radius, but in general all distributions settle for a relative radius of about 0.75 to 0.85. Lloyd's relaxation is the only technique that allows to generate Poisson disk distributions with such a high radius.

Figure 6.10 shows the power spectrum of Poisson disk distributions generated with dart throwing after several relaxation iterations. The peak at the principal frequency, and the smaller peaks at multiples of the principal frequency, increase in magnitude with each iteration, and the transition region becomes steeper with each iteration. Hiller et al. [2001] report that Lloyd's relaxation results in an increased anisotropy, but the anisotropy graphs do not support that claim, or the increase is not significant.

We have not succeeded in generating Poisson disk distributions that are large enough for sampling the zone plate function using Lloyd's relaxation method because constructing a Voronoi diagram of such large point distributions was too time consuming.

It is possible that Lloyd's relaxation, after many iterations, finds the global minimum (a hexagonal grid), and that the Poisson disk distribution regains periodicity. We have seen this happening for smaller distributions, but for larger distributions this does not seem to be a problem. Also, the problem can be avoided by monitoring the radius during relaxation.

The time required for 1, 2, 4, 8, 16 and 32 iterations, starting from a Poisson distribution of $16,384$ points, is respectively 21.308, 35.661, 64.397, 120.448, 234.140 and 458.460 seconds.

6.6 Shade's Poisson Disk Tiles

In 2000, Shade et al. presented the first tile-based approach for generating Poisson disk distributions. Their approach is an extension of the dart throwing algorithm based on Wang tiles.

Shade et al. construct a Poisson disk distribution over a set of eight Wang tiles using an algorithm similar to dart throwing. Before a point is added to a

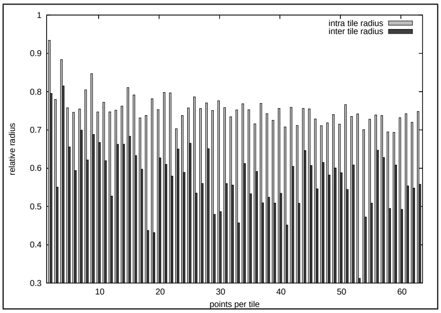


Figure 6.11: The intra and inter tile radius of the Poisson disk tile sets constructed with the method of Hiller et al.

tile, all possible neighboring tiles are checked, and the point is rejected if the minimum separation criterion is not met. Once a Poisson disk distribution is constructed over a set of Wang tiles, arbitrary large Poisson disk distributions can be generated in real time by tiling the plane with the Wang tiles using a stochastic tiling algorithm. For more details, we refer to Shade et al. [2000].

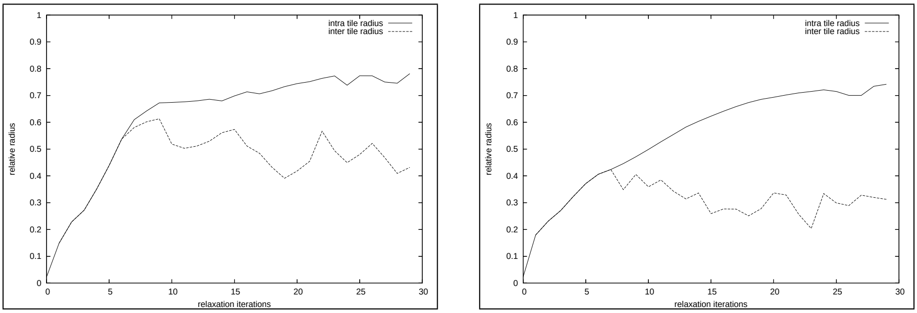
However, as noted by Cohen et al. [2003], this approach is flawed. The constraints of multiple tiles cause less points to be inserted near the edges and corners. This results in a noticeable lower density of points in those regions. The problem of multiple constraints is more severe than it might seem at first sight: placing a point near the corner of one tile makes it impossible, for at least one corner of every other tile in the tile set, to have a point nearby.

Although these problems limit the applicability of this method, the idea of using Wang tiles for generating Poisson disk distributions would later prove to be very valuable.

6.7 Tiled Blue Noise Samples

In 2001, Hiller et al. presented an approach based on Lloyd's relaxation algorithm rather than dart throwing to construct a Poisson disk distribution over a set of Wang tiles.

Hiller et al. construct a Poisson disk distribution over a set of eight Wang tiles using an algorithm similar to Lloyd's relaxation method. An initial point set is generated in the center of every tile. Each tile in the set is surrounded by all possible configurations of 8 tiles. For all of these configurations, a Voronoi diagram is constructed. Each Voronoi diagram determines a displacement vector for every point in the tile. All displacement vectors are averaged, and the points in the tile are moved accordingly. This process is iterated, until the point distributions stabilize. For more details, we refer to Hiller et al. [2001]. The method of Hiller et al. was later adopted by Cohen et al. [2003], in a paper



(a) $N = 19$

(b) $N = 53$

Figure 6.12: The intra and inter tile radius versus the number of relaxation iterations of the Poisson disk tile sets constructed with the method of Hiller et al. (a) The tile set with $N = 19$ points per tile. (b) The tile set with $N = 53$ points per tile.

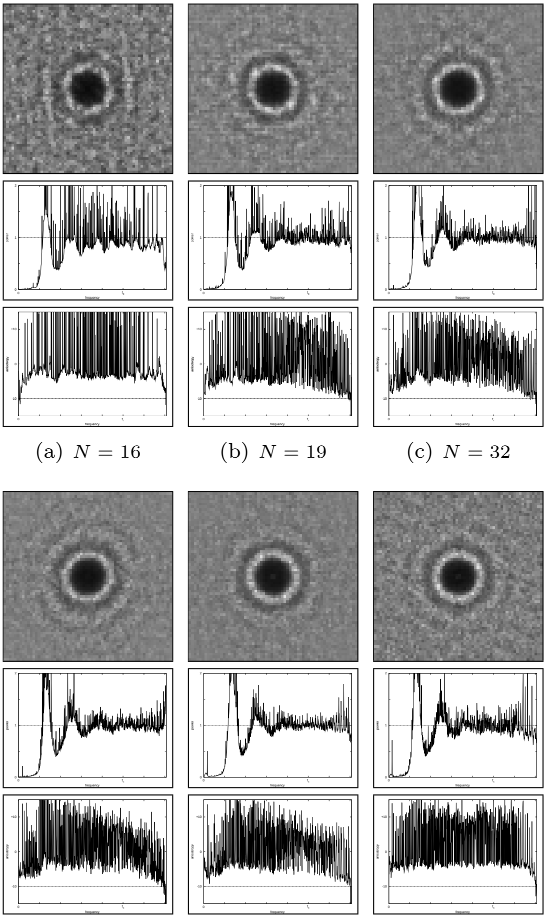
that popularized Wang tiles in the field of computer graphics.

To analyze this method, we reviewed the original data of Hiller et al. The data consists of 63 tile sets, with 1 to 63 points per tile. For each of these tile sets, the positions of the points from iteration 0 to 29 are given.

From the original data we computed radius statistics, which were not listed in the original paper. Figure 6.11 shows the intra and inter tile radius of the 63 tile sets. The intra tile radius, which is the radius of the Poisson disk distribution within a single tile, is relatively high ($\rho \approx 0.80$), as expected from an approach based on Lloyd's relaxation. However, the inter tile radius, which is the radius of the Poisson disk distribution taking into account neighboring tiles, and thus the final radius of the generated distribution, is relatively low (often $\rho \approx 0.40$ or even less). This shows that the low radius is due to difficulties at the tile edges, which indicates problems with the construction method. Figure 6.12 shows the intra and inter tile radius versus the number of relaxation iterations for two tile sets. As expected, the intra tile radius converges to a relative radius of about 0.80. The inter tile radius, however, does not seem to converge.

Figure 6.13 shows power spectra of Poisson disk distributions generated with the method of Hiller et al. Compared to the previous methods we have discussed, the power spectra are relatively bad. This is due to the construction method, but also because only eight tiles are used. In section 6.10, we will show that eight tiles is not enough for generating Poisson disk distributions with good spectral properties. However, due to the problems mentioned above, the algorithm is most likely not capable of handling larger tile sets. Even with eight tiles, the number of eight-tile configurations is that large that the displacement vectors tend to average each other out.

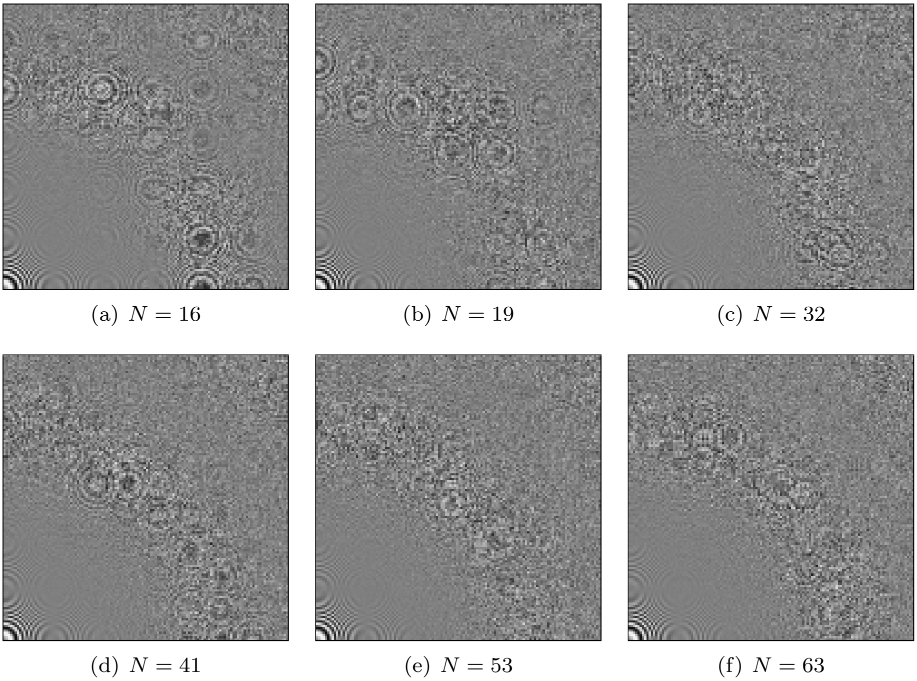
The power spectrum also reveals a regular pattern of spikes. The autocorre-



(a) $N = 16$ (b) $N = 19$ (c) $N = 32$

(d) $N = 41$ (e) $N = 53$ (f) $N = 63$

Figure 6.13: Spectral analysis of Poisson disk distributions generated with the method of Hiller et al. The Poisson disk distributions were generated by tiling (a) 32×32, (b) 29×29, (c) 22×22, (d) 20×20, (e) 18×18 and (f) 16×16 tiles of Poisson disk tile sets consisting of $T = 8$ tiles with (a) $N = 16$, (b) $N = 19$, (c) $N = 32$, (d) $N = 41$, (e) $N = 53$ and (f) $N = 63$ points per tile.


(a) $N = 16$

(b) $N = 19$

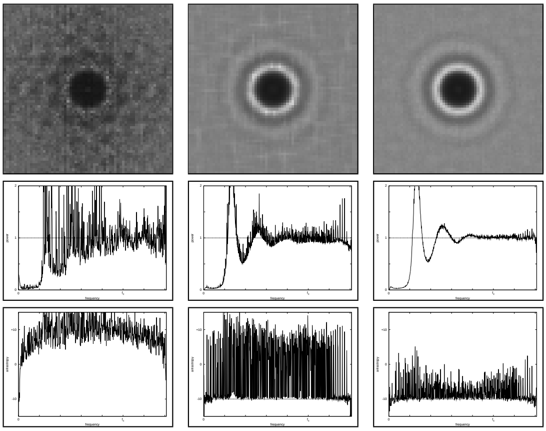
(c) $N = 32$

(d) $N = 41$

(e) $N = 53$

(f) $N = 63$

Figure 6.14: The zone plate function sampled with a Poisson disk distribution generated with the method of Hiller et al. The Poisson disk distributions were generated by tiling (a) 128 × 128, (b) 117 × 117, (c) 91 × 91, (d) 80 × 80, (e) 70 × 70 and (f) 64 × 64 tiles of Poisson disk tile sets consisting of $T = 8$ tiles with (a) $N = 16$, (b) $N = 19$, (c) $N = 32$, (d) $N = 41$, (e) $N = 53$ and (f) $N = 63$ points per tile.



(a) Ostromoukhov (b) edge-based (c) corner-based

Figure 6.15: Spectral analysis of Poisson disk distributions generated with the method of Ostromoukhov et al. (a) A periodogram of a Poisson disk distributions generated with the method of Ostromoukhov et al. (b) A periodogram of a Poisson disk distribution generated with edge-based Poisson disk tiles. (c) A periodogram of a Poisson disk distribution generated with corner-based Poisson disk tiles.

lation peaks for lags that are a multiple of the tile size. This is because only a limited number of tiles are used, and because tiles often have points in common. In the power spectrum, this results in a grid like pattern of peaks on frequencies that are a multiple of the reciprocal of the tile size. When using more tiles, or when tiles have less points in common, the spikes decrease in magnitude. However, because only a finite number of tiles is used, and because the tiles have to remain compatible, it is impossible to eliminate the spikes completely. Note that this effect is not limited to the method of Hiller et al. The regular pattern of spikes is visible in all methods that are based on tilings.

Figure 6.14 shows the zone plate function sampled using a Poisson disk distribution generated with the method of Hiller et al. The low-frequency region of the zone plate function is reproduced relatively good, however, the high-frequency region shows severe structured aliasing artifacts. This is also due to the construction method and because only eight tiles are used.

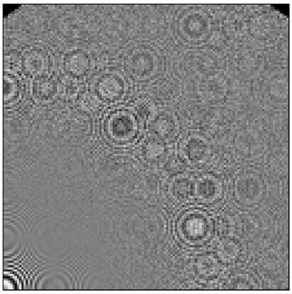


Figure 6.16: The zone plate function sampled with a Poisson disk distribution generated with the method of Ostromoukhov et al.

6.8 Fast Hierarchical Importance Sampling with Blue Noise Properties

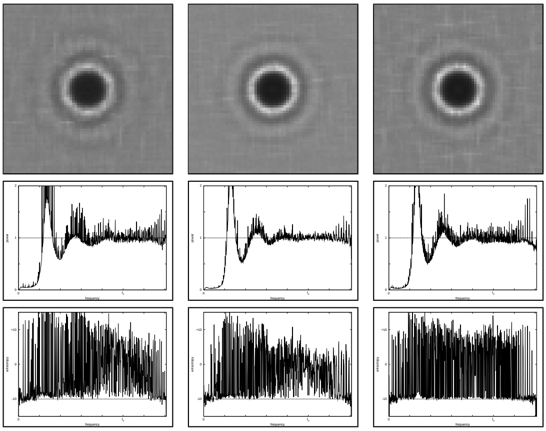
In 2004, Ostromoukhov et al. presented an interesting technique for generating point distributions with blue noise properties over a given density. Although the method of Ostromoukhov et al. is capable of generating nonuniform distributions, we only analyze it here in the context of uniform Poisson disk distributions.

The points are placed on the vertices of a Penrose tiling [Penrose, 1974; Grünbaum and Shepard, 1986], recursively subdivided according to a given density. The point distribution is optimized by moving the points according to precomputed displacement vectors obtained with Lloyd's relaxation method. For more details, we refer to Ostromoukhov et al. [2004].

The radius of the point distributions generated over a constant density is comparable with dart throwing. The relative radii we have measured varied between 0.65 and 0.75.

Because the method is deterministic, it is not possible to compute a power spectrum estimate. Instead, figure 6.15 shows the periodogram of a Poisson disk distribution generated with the approach of Ostromoukhov et al. For comparison, the periodogram of a Poisson disk distribution generated with edge-based Poisson disk tiles (see section 6.9) and corner-based Poisson disk tiles (see section 6.11) is included. The periodogram of the Poisson disk distribution generated with the approach of Ostromoukhov et al. shows severe artifacts. A star-like pattern of large spikes revealing the 10-fold symmetry of the underlying Penrose tiling is visible. The anisotropy of is also significantly higher.

Figure 6.16 shows the zone plate function sampled using a Poisson disk distribution generated with the method of Ostromoukhov et al. Again, the low-frequency region of the zone plate function is reproduced relatively good while the high-frequency region shows severe structured aliasing artifacts. The ex-



(a) $N = 32$ (b) $N = 48$ (c) $N = 64$

Figure 6.17: Spectral analysis of Poisson disk distributions generated with edge-based Poisson disk tiles. The Poisson disk distributions were generated by tiling (a) 23×23, (b) 18×18 and (c) 16×16 tiles of edge-based Poisson disk tile sets consisting of $4,096$ tiles with (a) $N = 32$, (b) $N = 48$ and (c) $N = 64$ points per tile, and a relative radius of $\rho = 0.75$.

ample code provided by Ostromoukhov et al. does not generate points in two regions. This explains the black triangular regions at the top of figure 6.16.

The method of Ostromoukhov et al. is very fast. The time needed to generate a Poisson disk distribution of $16,384$ points is 0.192 seconds.

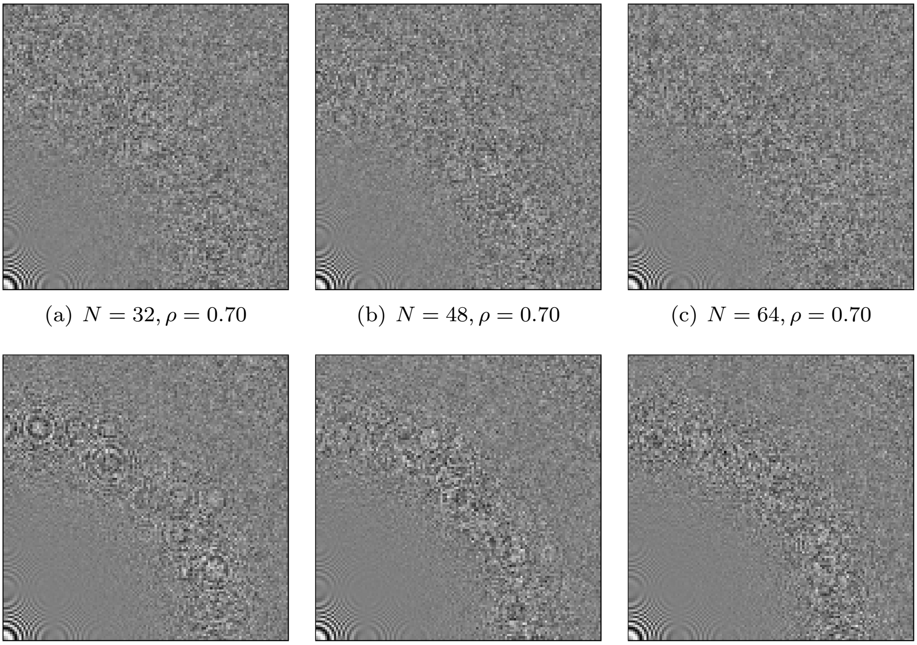
6.9 Edge-Based Poisson Disk Tiles

In 2005, we presented edge-based Poisson disk tiles, a method for constructing a Poisson disk distribution over a set of Wang tiles [Lagae and Dutré, 2005a]. Edge-based Poisson disk tiles are discussed in detail in section 4.3.

Edge-based Poisson disk tiles are capable of generating Poisson disk distributions with a relative radius up to 0.85. This is because edge-based Poisson disk distributions use Lloyd's relaxation method, and carefully handle points near the tile boundary.

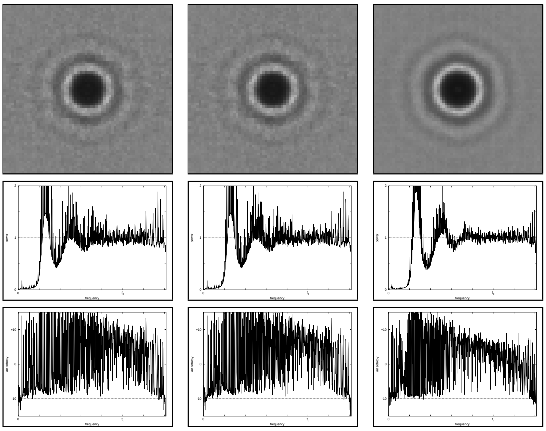
Figure 6.17 shows the power spectrum of tiled Poisson disk distributions generated with edge-based Poisson disk tiles. Compared to the method of Hiller et al. [2001] the power spectra are relatively good, they are more similar to the reference power spectrum and the peaks are smaller. However, there are some strange wavelike artifacts in the power spectra that we cannot explain.

Figure 6.18 shows the zone plate function sampled using Poisson disk distribution generated with edge-based Poisson disk tiles. Although a small amount


(a) $N = 32, \rho = 0.70$ (b) $N = 48, \rho = 0.70$ (c) $N = 64, \rho = 0.70$

(d) $N = 32, \rho = 0.80$ (e) $N = 48, \rho = 0.80$ (f) $N = 64, \rho = 0.80$

Figure 6.18: The zone plate function sampled with a Poisson disk distribution generated with edge-based Poisson disk tiles. The Poisson disk distributions were generated by tiling (a, d) 91×91, (b, e) 74×74 and (c, f) 64×64 tiles of edge-based Poisson disk tile sets consisting of $4,096$ tiles with (a, d) $N = 32$, (b, e) $N = 48$ and (c, f) $N = 64$ points per tile, and a relative radius of (a, b, c) $\rho = 0.70$ and (d, e, f) $\rho = 0.80$. Note that the bottom row exhibits more aliasing artifacts than the top row.



(a) $T\!=\!8,N\!=\!24$ (b) $T\!=\!32,N\!=\!48$ (c) $T\!=\!128,N\!=\!96$

Figure 6.19: Spectral analysis of Poisson disk distributions generated with template Poisson disk tiles. The Poisson disk distributions were generated by tiling (a) 26×26, (b) 18×18 and (c) 13×13 tiles of template Poisson disk tile sets consisting of (a) $T = 8$, (b) $T = 32$ and (c) $T = 128$ tiles with (a) $N = 24$, (b) $N = 48$ and (c) $N = 96$ points per tile, and a relative radius of $\rho = 0.75$.

of structured aliasing artifacts are visible, the zone plate function is reproduced relatively good. Poisson disk distribution with a relative radius of 0.80 reproduce the low-frequency region of the zone plate function better than Poisson disk distribution with a relative radius of 0.70, but are also more subject to aliasing. This is probably because a larger relative radius introduces more regularity in the Poisson disk distribution.

Tile-based methods like edge-based Poisson disk tiles are typically very fast. This is because generating a stochastic tiling can be done very efficiently.

6.10 Template Poisson Disk Tiles

In 2005, we presented template Poisson disk tiles [Lagae and Dutré, 2005b]. Template Poisson disk tiles are discussed in detail in section 4.4.

Template Poisson disk tiles were designed to study the effect of the size of the tile set on the quality of the tiled Poisson disk distributions. Figure 6.20 shows the power spectrum of tiled Poisson disk distributions generated with template Poisson disk tile sets, for a variety of parameters. In general, the power spectra for 16 and 24 points per tile and the power spectra for 1, 2, 4, 8 and 16 tiles are rather bad, while spectra for a larger tile set size and number of points per tile are rather good, and the power spectra for 128 and 256 tiles are not that different. From this experiment it can be concluded that roughly

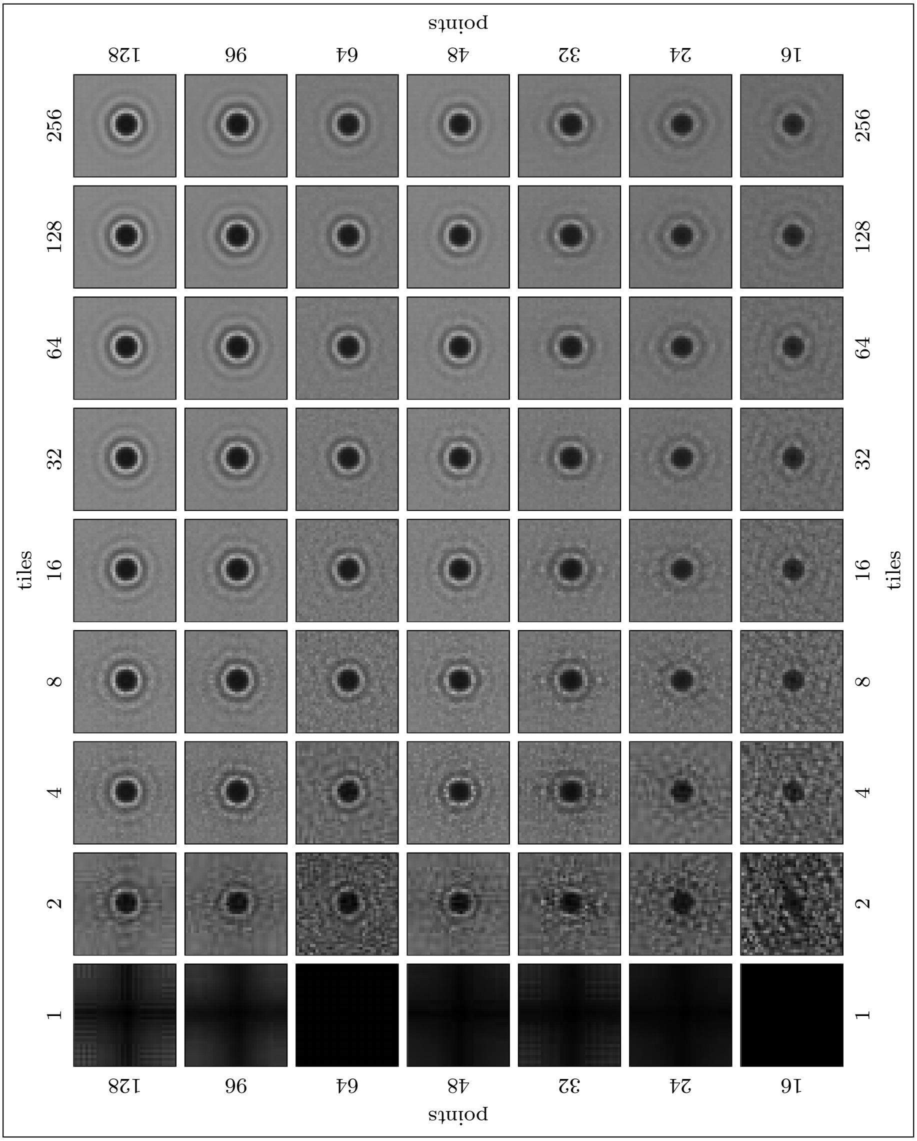


Figure 6.20: The power spectrum of Poisson disk distributions generated with template Poisson disk tiles for several tile set sizes and number of points per tile. (Please note the figure is rotated 90 degrees counterclockwise.)

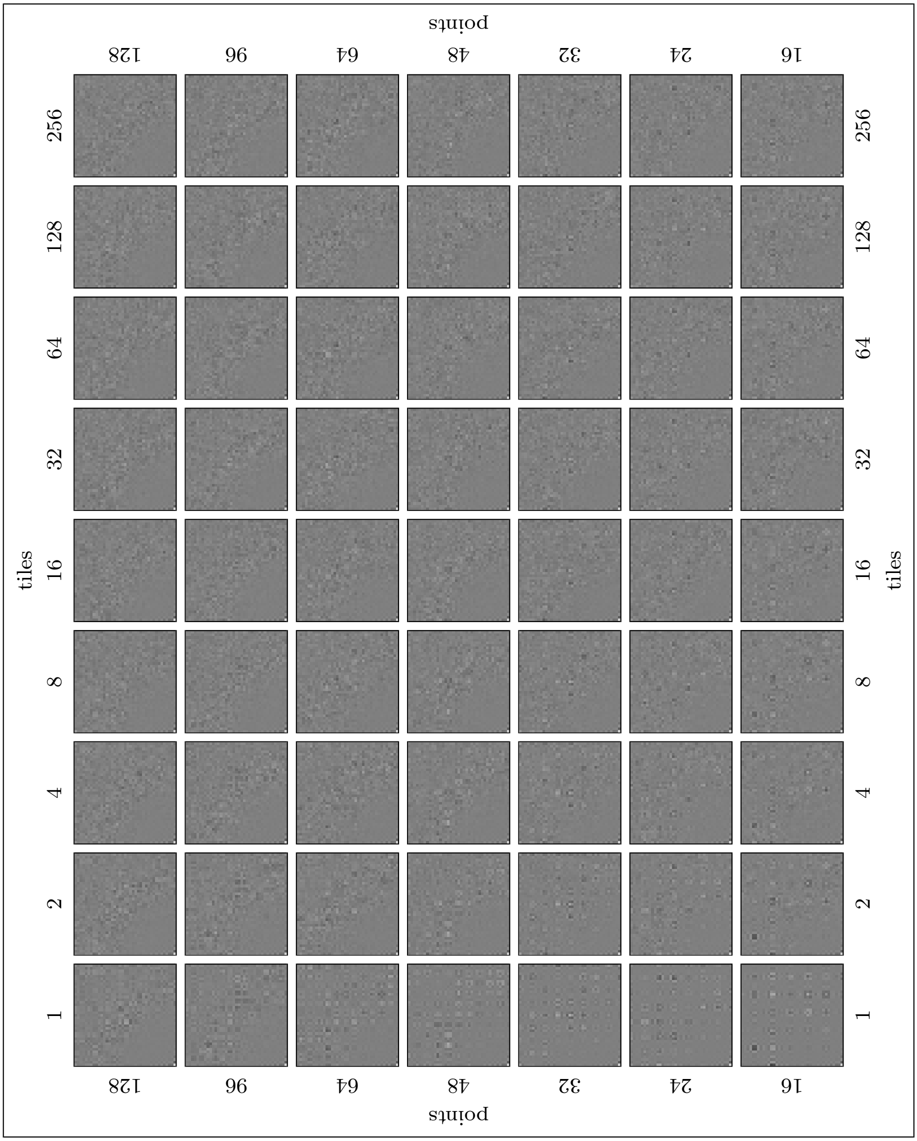


Figure 6.21: The zone plate function sampled with a Poisson disk distribution generated with template Poisson disk tiles. (Please note the figure is rotated 90 degrees counterclockwise.)

at least 32 points per tile and 32 tiles are needed to generate tiled Poisson disk distributions with good quality, and that it probably does not make much sense to use several thousands of tiles. We believe these results generalize to other kinds of Poisson disk tiles, and explain why the power spectrum of tiled blue noise samples is rather bad.

Figure 6.20 shows more detailed power spectra of tiled Poisson disk distributions generated with template Poisson disk tiles. Although the power spectra are not that bad, the square lattice of peaks is more pronounced. This is because all tiles in a set of template Poisson disk tiles have a number of points in common, namely the points in the tile template, which are replicated periodically when generating a tiled Poisson disk distribution. These peaks also turn up in the radially averaged power spectrum and anisotropy graphs.

Figure 6.21 shows the zone plate function sampled using Poisson disk distribution generated with template Poisson disk tiles for a variety of parameters. The reconstructed zone plate function shows severe structured aliasing artifacts. Although the reconstruction of the zone plate function improves when using more tiles or more points per tile, the periodically replicated tile template causes severe structured aliasing artifacts.

Template Poisson disk tiles are not useful in practice. However, template Poisson disk tiles allow to study the effect of the size of the tile set and the number of points per tile on the power spectrum, and help to better understand the spectral characteristics of other tile based approaches.

6.11 Corner-Based Poisson Disk Tiles

In 2006, we presented corner-based Poisson disk tiles, a method for constructing a Poisson disk distribution over a set of corner tiles [Lagae and Dutré, 2006a]. Corner-based Poisson disk tiles are discussed in detail in section 4.5.

In contrast with edge-based Poisson disk tiles, corner-based Poisson disk tile sets can be constructed for a variety of tile set sizes. For C colors, an edge-based Poisson disk tile set contains C^{12} tiles. The only practical choice for C is 2, which results in $4,096$ tiles. For C colors, a corner-based Poisson disk tile set counts only C^4 tiles, enabling tile sets of 16, 81, 256, 625, $1,296$, $2,401$ and $4,096$ tiles. This solves the problem of edge-based Poisson disk tiles having too much tiles, but also allows to trade spectral quality for tile set size.

Like edge-based Poisson disk tiles, corner-based Poisson disk tiles are capable of generating Poisson disk distributions with a relative radius up to 0.85. This is because corner-based Poisson disk distributions use Lloyd's relaxation method, and carefully handle points near the tile boundary.

Figure 6.22 shows the power spectrum of tiled Poisson disk distributions generated with corner-based Poisson disk tile sets, for a variety of parameters. These results confirm earlier findings. Starting from 3 colors (81 tiles) and 32 points per tile the spectra are rather good. The power spectra of corner-based

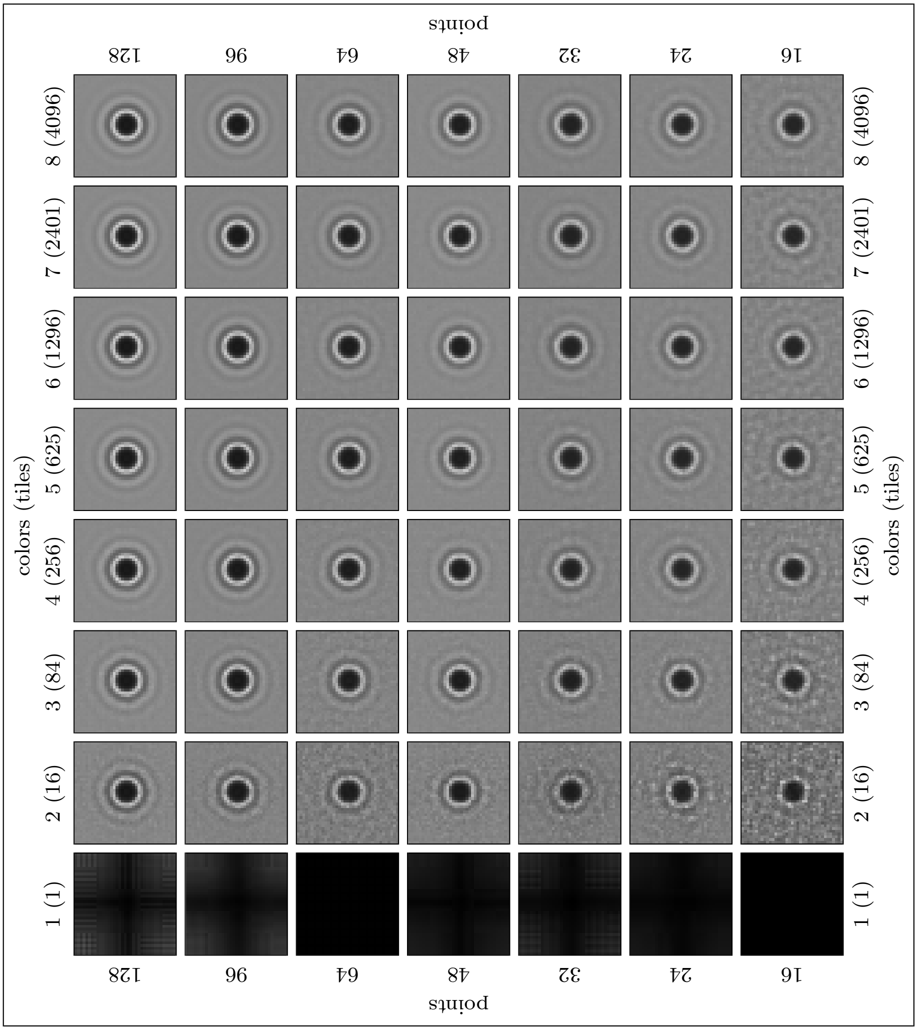
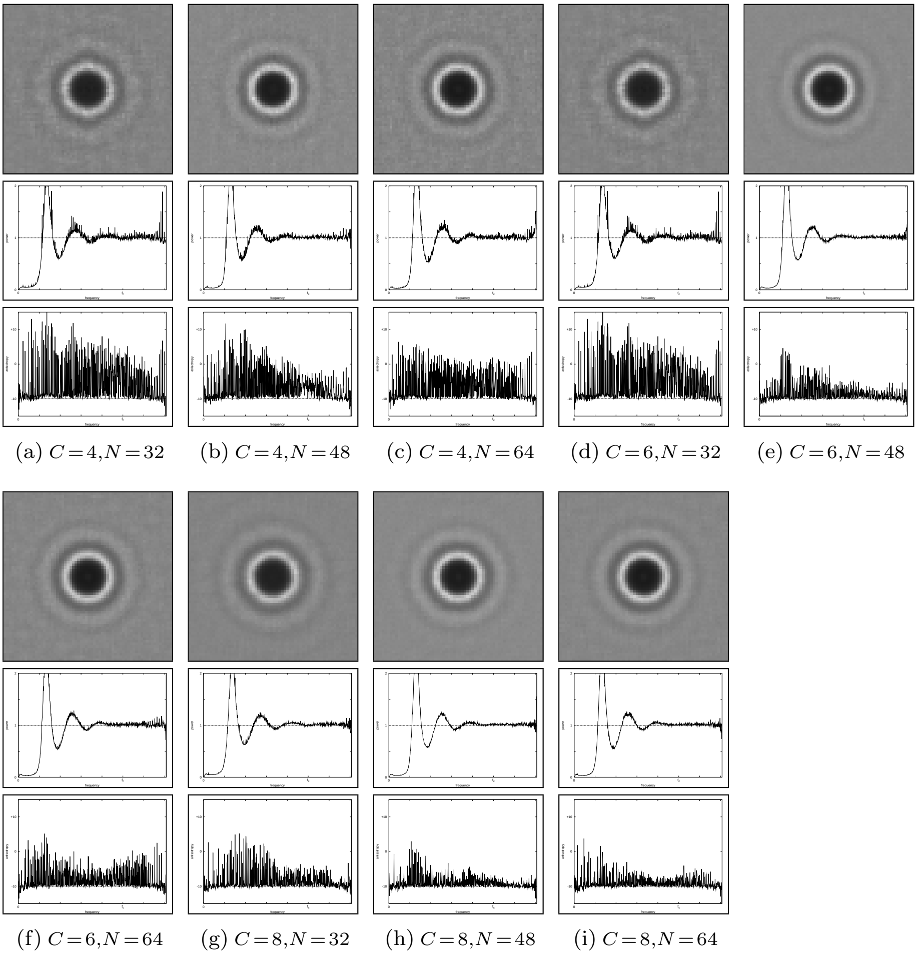


Figure 6.22: The power spectrum of Poisson disk distributions generated with corner-based Poisson disk tiles for several tile set sizes and number of points per tile. (Please note the figure is rotated 90 degrees counterclockwise.)



(a) $C = 4, N = 32$ (b) $C = 4, N = 48$ (c) $C = 4, N = 64$ (d) $C = 6, N = 32$ (e) $C = 6, N = 48$

(f) $C = 6, N = 64$ (g) $C = 8, N = 32$ (h) $C = 8, N = 48$ (i) $C = 8, N = 64$

Figure 6.23: Spectral analysis of Poisson disk distributions generated with corner-based Poisson disk tiles. The Poisson disk distributions were generated by tiling (a, d, g) 23×23, (b, e, h) 18×18 and (c, f, i) 16×16 tiles of corner-based Poisson disk tile sets over (a, b, c) $C = 4$, (d, e, f) $C = 6$ and (g, h, i) $C = 8$ colors with (a, d, g) $N = 32$, (b, e, h) $N = 48$ and (c, f, i) $N = 64$ points per tile, and a relative radius of $\rho = 0.75$. Corner tile sets over $C = 4$, $C = 6$ and $C = 8$ colors consist of respectively 256, 1,296 and 4,096 tiles.

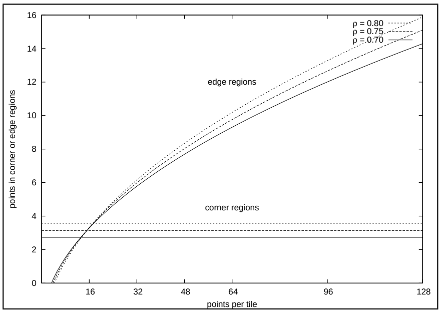


Figure 6.24: The number of points in corner and edge regions versus the number of points per tile.

Poisson disk tiles are significantly better than the spectra of edge-based Poisson disk tiles, even when using smaller tile sets.

Figure 6.23 shows that the peaks in power spectra of tiled Poisson disk distributions generated with corner-based Poisson disk tiles are much smaller than those generated with edge-based Poisson disk tiles. The radially averaged power spectrum is almost the same as that of non-tiled Poisson disk distributions. Although the anisotropy still contains peaks, it is much smaller than the anisotropy of other Poisson disk tiles.

Figure 6.24 plots the expected number of points in edge and corner regions. The expected number of points in corner regions is constant, and much smaller than the expected number of points in edge regions. Therefore, more variation in edge tiles will produce a better power spectrum than more variation in corner tiles. For $C = 2$ colors, a set of edge-based Poisson disk tiles consists of $4,096$ tiles and there are 2 different edge tiles (and 16 different corner tiles). For $C = 8$ colors, a set of corner-based Poisson disk tiles also counts $4,096$ tiles, but there are 64 different edge tiles (and 8 different corner tiles). This is an important reason why corner-based Poisson disk tiles produce better power spectra than edge-based Poisson disk tiles.

Figure 6.21 shows the zone plate function sampled using Poisson disk distribution generated with corner-based Poisson disk tiles for a variety of parameters. Corner-based Poisson disk tiles perform significantly better than other tile-based approaches, even when using small tile sets.

The time required to generate Poisson disk distributions of approximately $16,348$ points using a tile set with 4 colors and 32, 48 and 64 points per tile is respectively 0.088, 0.102 and 0.122 seconds. With 6 colors the time is respectively 0.238, 0.319 and 0.415 seconds, and with 8 colors, the time is respectively 0.629, 0.911 and 1.200 seconds. Most of this time is used for loading the tile set stored in a text file.

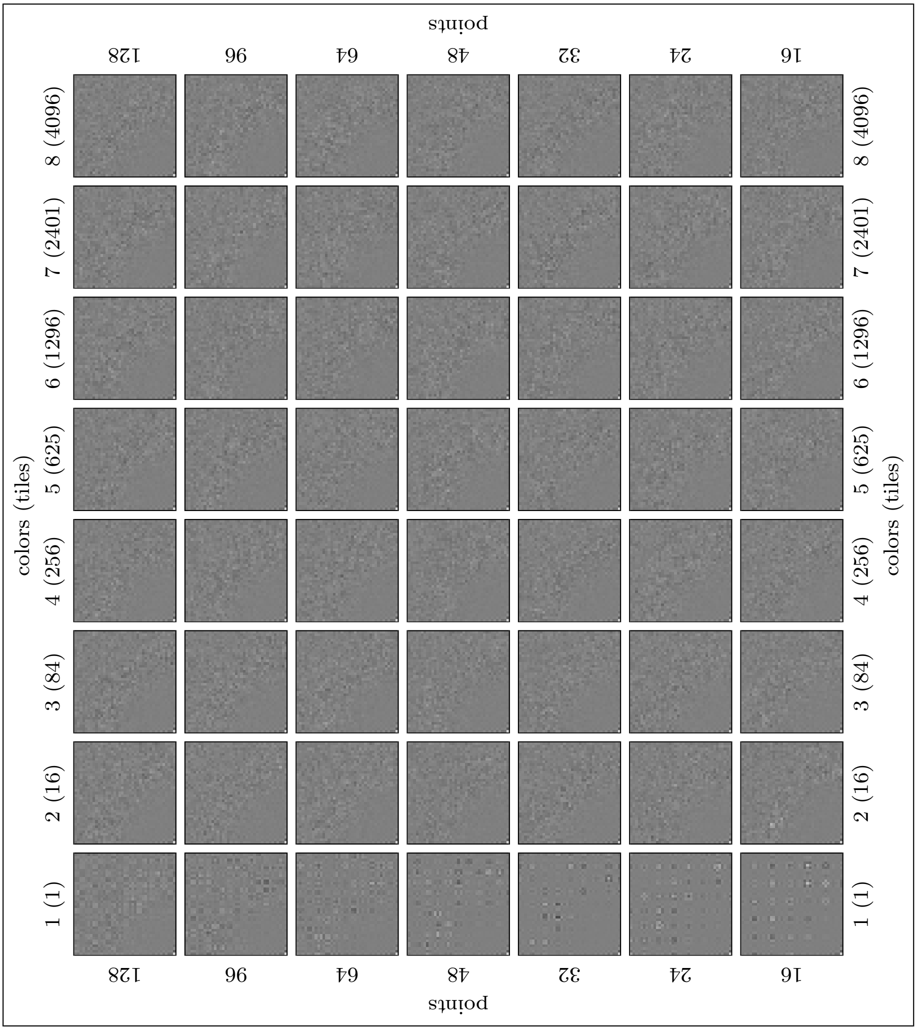
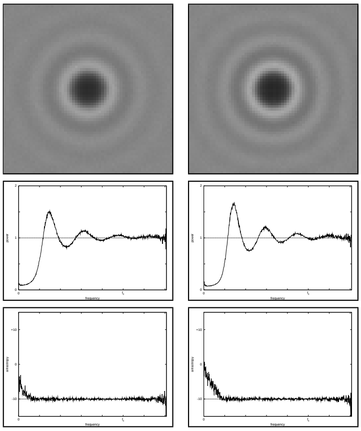


Figure 6.25: The zone plate function sampled with a Poisson disk distribution generated with corner-based Poisson disk tiles. (Please note the figure is rotated 90 degrees counterclockwise.)



(a) $\rho = 0.70$ (b) $\rho = 0.75$

Figure 6.26: Spectral analysis of Poisson disk distributions generated with the method of Jones. The Poisson disk distributions have a relative radius of (a) $\rho = 0.70$ and (b) $\rho = 0.75$.

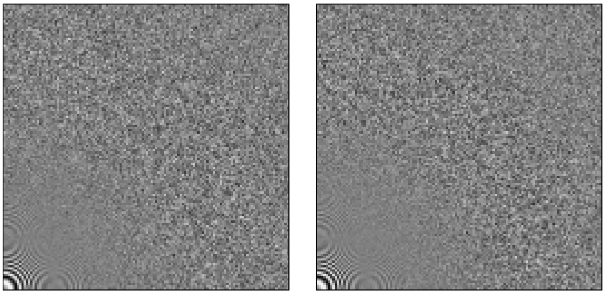
6.12 Efficient Generation of Poisson Disk Sampling Patterns

In 2006, Jones presented an efficient implementation of the dart throwing algorithm.

When generating a Poisson disk distribution, the original implementation of the dart throwing algorithm [Cook, 1986] uses trial and error to add new points to the Poisson disk distribution. The method of Jones uses a novel data structure based on the Voronoi diagram to efficiently sample the free space in the Poisson disk distribution. For more details, we refer to [Jones, 2006].

In terms of radius, spectral properties and sampling performance, the method of Jones is very similar to the original implementation of the dart throwing algorithm. Figure 6.26 shows the power spectrum of Poisson disk distributions generated with the method of Jones, and figure 6.27 shows the zone plate function sampled using a Poisson disk distribution generated with the method of Jones. As expected, the method has the same spectral properties and sampling performance as the traditional dart throwing algorithm. The Poisson disk distributions generated with the method of Jones are not toroidal. This explains the central horizontal and vertical lines in the power spectrum, and the increased anisotropy at low radial frequencies.

The implementation of Jones has a time complexity of $O(N \log N)$ and is significantly faster than the traditional implementation of the dart throwing algorithm. The time required to generate a Poisson disk distribution of $16,384$



(a) $\rho = 0.70$

(b) $\rho = 0.75$

Figure 6.27: The zone plate function sampled with a Poisson disk distribution generated with the method of Jones. The Poisson disk distributions have a relative radius of (a) $\rho = 0.70$ and (b) $\rho = 0.75$.

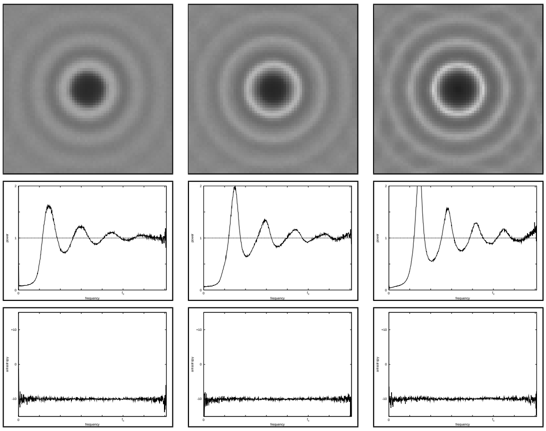
points with a relative radius of 0.70 and 0.75 is respectively 3.511 and 3.403 seconds.

6.13 A Spatial Data Structure for Fast Poisson Disk Sample Generation

Also Dunbar and Humphreys presented an efficient implementation of the dart throwing algorithm in 2006.

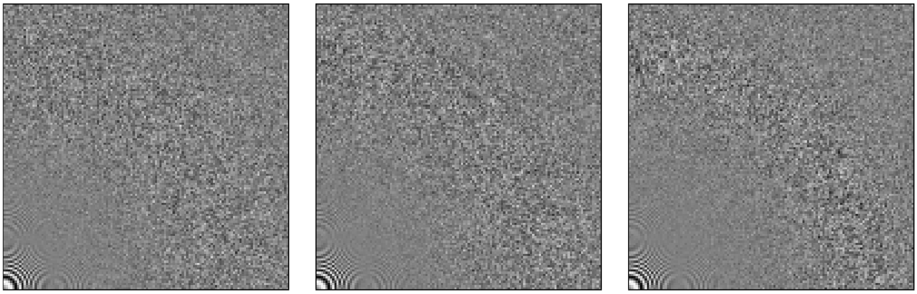
The method of Dunbar and Humphreys is similar to the one of Jones [2006]. Dunbar and Humphreys use a data structure based on scalloped sectors to efficiently sample the free space in the Poisson disk distribution. The algorithm of Dunbar and Humphreys also has a time complexity of $O(N \log N)$. Next to the logarithmic algorithm, Dunbar and Humphreys also present a linear and a boundary sampling algorithm. The linear sampling algorithm is obtained by dropping the requirement that the available neighborhoods be sampled according to an area-weighted probability density function. Although the linear algorithm has linear time complexity, it is in practice not much faster than the logarithmic one. The boundary sampling algorithm also runs in linear time and places points such that their disks touch. The boundary sampling algorithm is also in practice significantly faster than the logarithmic one. For more details, we refer to Dunbar and Humphreys [2006].

In terms of radius, spectral properties and sampling performance, the algorithms of Dunbar and Humphreys are similar to the original implementation of the dart throwing algorithm. Figure 6.28 shows the power spectrum of Poisson disk distributions generated with the algorithms of Dunbar and Humphreys, and figure 6.29 shows the zone plate function sampled using a Poisson disk distribution generated with the algorithms of Dunbar and Humphreys. The



(a) logarithmic (b) linear (c) boundary

Figure 6.28: Spectral analysis of Poisson disk distributions generated with different variants of the method of Dunbar and Humphreys. The Poisson disk distributions were generated with the (a) logarithmic, (b) linear and (c) boundary method.



(a) logarithmic (b) linear (c) boundary

Figure 6.29: The zone plate function sampled with a Poisson disk distribution generated with the method of Dunbar and Humphreys. The Poisson disk distributions were generated with the (a) logarithmic, (b) linear and (c) boundary method.

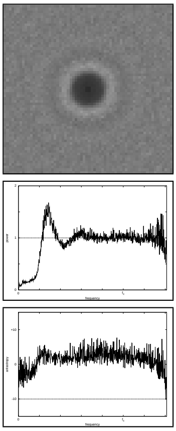


Figure 6.30: Spectral analysis of Poisson disk distributions generated with the method of Kopf et al.

logarithmic algorithm has the same spectral properties and sampling performance as the traditional dart throwing algorithm. The peaks in the power spectrum of the linear algorithm at multiples of the principal frequency are slightly larger and the power spectrum of the linear algorithm seem to be a bit skew. The power spectrum of the boundary algorithm is somewhat similar to the power spectrum of Poisson disk distributions optimized with Lloyd's relaxation method, although the peaks in the power spectrum at multiples of the principal frequency are significantly larger. The zone plate function reconstructed using a Poisson disk distribution generated with the boundary algorithm shows some structured aliasing.

The logarithmic and linear algorithms of Dunbar and Humphreys are comparable in speed with the method of Jones. The boundary sampling algorithm is significantly faster. The time required to generate a Poisson disk distribution with an absolute radius of 0.005877 (corresponding with a relative radius of 0.70 in the previous methods) using the logarithmic, linear and boundary algorithm is respectively 7.002, 7.911 and 0.137 seconds.

6.14 Recursive Wang Tiles for Real-Time Blue Noise

In 2006, Kopf et al. presented a method for efficiently generating nonuniform Poisson disk distributions. Although the method of Kopf et al. is capable of generating nonuniform Poisson disk distribution, we only analyze it here in the context of uniform Poisson disk distributions.

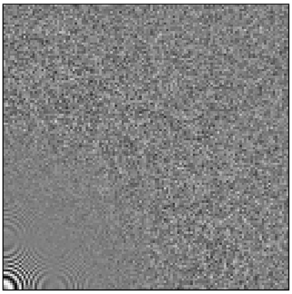


Figure 6.31: The zone plate function sampled with a Poisson disk distribution generated with the method of Kopf et al.

Kopf et al. use recursive Wang tiles that contain self-similar and progressive Poisson disk distributions. The property of self-similarity allows to increase the point density in large steps by recursively subdividing the tile. A progressive point distribution allows to smoothly adjust the density of points. Combined, these two properties enable an algorithm for generating varying-density point distributions in real time. For more details, we refer to Kopf et al. [2006].

The radius of the point distributions generated over a constant density is surprisingly low. Using the original tile set of Kopf et al., we have measured relative radii between 0.44 and less than 0.001. Most distributions had a relative radius of less than 0.001.

Figure 6.30 shows the power spectrum of Poisson disk distribution generated with the approach of Kopf et al. Compared to the previous techniques, the power spectrum rather bad, and the anisotropy is rather high.

Figure 6.31 shows the zone plate function sampled using a Poisson disk distribution generated with the method of Kopf et al. No structured aliasing artifacts are visible. This is most likely because Kopf et al. use tiles with a very large number of points.

As most tile-based methods, the method of Kopf et al. is rather fast. The time required to generate a Poisson disk distribution of approximately 16, 384 tiles is 1.005 seconds.

6.15 A Comparison of Methods for Generating Poisson Sphere Distributions

In this section, we analyze dart throwing, relaxation dart throwing and Lloyd's relaxation method for Poisson sphere distributions, and corner-based Poisson sphere tiles.

Dart throwing, relaxation dart throwing and Lloyd's relaxation method have direct three-dimensional equivalents. Corner-based Poisson sphere tiles are

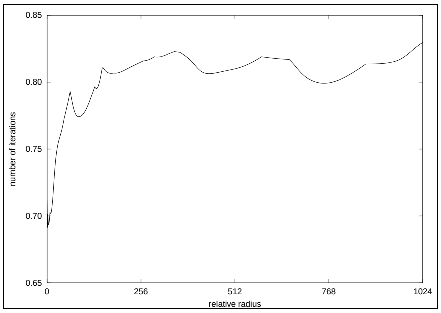


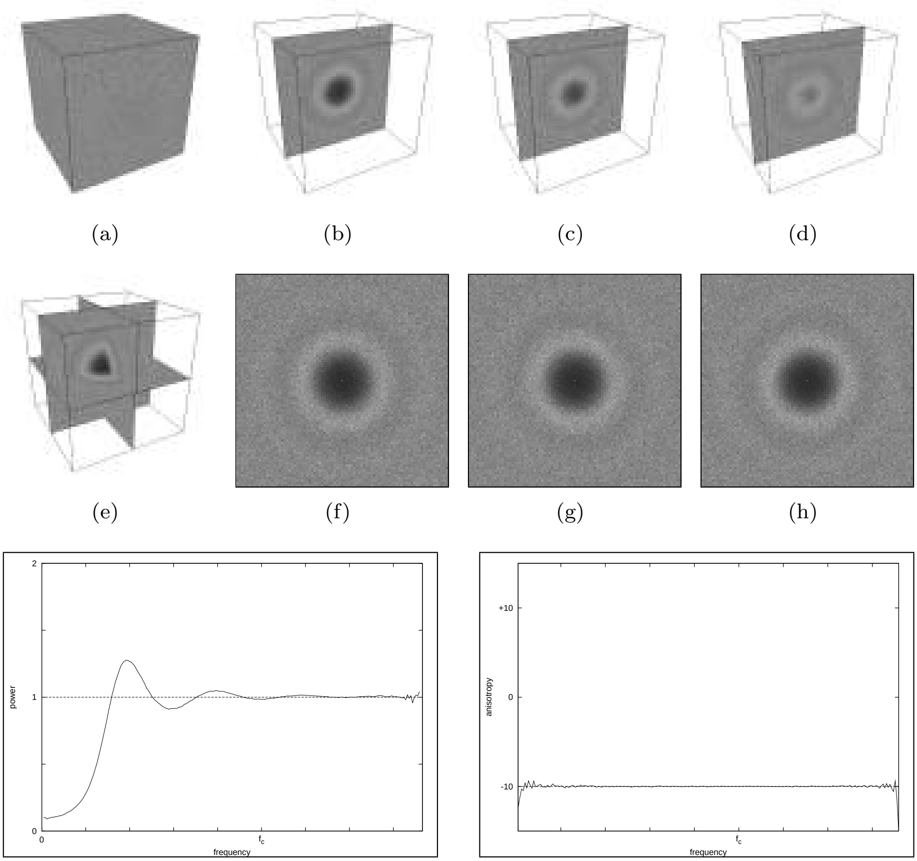
Figure 6.32: The relative radius versus the number of iterations of Lloyd's relaxation method applied to a Poisson sphere distribution.

discussed in detail in section 4.6.3.

The methods for analyzing Poisson disk distributions have direct three-dimensional equivalents. The radius of Poisson sphere distributions is analyzed using the three-dimensional relative radius specification scheme introduced in section 4.6.1. The power spectrum of a Poisson sphere distribution is three-dimensional rather than two-dimensional, and is also radially symmetric. Sampling performance is not explicitely tested, because Poisson sphere distributions are not used for sampling as much as Poisson disk distributions.

Experiments show that dart throwing can be used for generating Poisson sphere distributions with a relative radius up to 0.70, and that the relative radius of Poisson sphere distributions generated with relaxation dart throwing using an initial radius of 0.15, a radius reduction factor of 0.99, and $1,000$ failed attempts is 0.70. Lloyd's relaxation method does not converge as easily in three dimensions as in two dimensions. This is illustrated in figure 6.32. Although the global trend indicates convergence, the method seems to get stuck in local minima often. This is most likely because the maximum radius configurations are more stable in two dimensions than in three. As a consequence, much more iterations are needed in three dimensions. Corner-based Poisson disk tiles are capable generating Poisson sphere distributions with a large relative radius.

Figure 6.33 shows the power spectrum of Poisson sphere distributions generated with dart throwing. The power spectrum is radially symmetric and exhibits the typical blue noise power spectrum. Figure 6.34 shows the power spectrum of Poisson sphere distributions generated with corner-based Poisson sphere tiles. The radially averaged power spectrum is rather good. The anisotropy is a bit high.



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j)

Figure 6.33: Spectral analysis of Poisson sphere distributions generated with dart throwing. The Poisson sphere distributions consist of $65,536$ points and have a relative radius of $\rho = 0.70$. (a) The 3D power spectrum. (b, c, d) Several 2D slices of the power spectrum. (e) The coordinate plane slices of the power spectrum. (f, g, h) Close-ups of the coordinate plane slices of the power spectrum. (i) The radially averaged power spectrum. (j) The anisotropy.

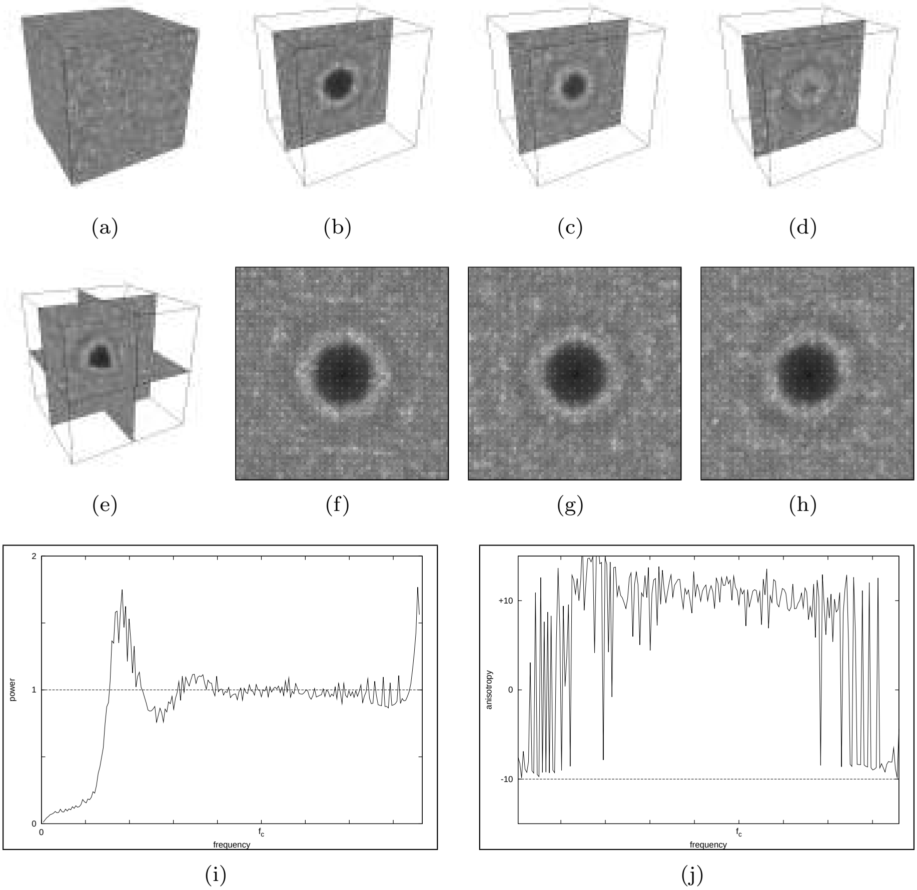


Figure 6.34: Spectral analysis of Poisson sphere distributions generated with Poisson sphere tiles. The Poisson sphere distributions were generated by tiling $8 \times 8 \times 8$ tiles of a corner-based Poisson sphere tile set over $C = 2$ colors consisting of 256 tiles with $N = 128$ points per tile, and a relative radius of $\rho = 0.75$. (a) The 3D power spectrum. (b, c, d) Several 2D slices of the power spectrum. (e) The coordinate plane slices of the power spectrum. (f, g, h) Close-ups of the coordinate plane slices of the power spectrum. (i) The radially averaged power spectrum. (j) The anisotropy.

6.16 Conclusion

In this chapter, we have introduced several methods for analyzing Poisson disk distributions, and we have compared methods for generating Poisson disk distributions and Poisson sphere distributions.

Dart throwing, relaxation dart throwing and Lloyd's relaxation method are only suited for applications that are not interactive, and for applications that do not require large Poisson disk distributions. The accelerated dart throwing algorithms of Jones and Dunbar and Humphreys are suited for interactive applications, but they are still relatively slow compared to tile based approaches, and if a large radius is required, must still be followed by Lloyd's relaxation method.

For real-time applications, and applications that require large Poisson disk distributions, tile-based approaches are the only option. Shade's Poisson disk tiles have major shortcomings and should not be used. We recommend not to use tiled blue noise samples because the radius of the generated distributions is low, and because the power spectrum is relatively bad. Edge-based Poisson disk tiles are better, both in terms of radius and spectral properties. Template Poisson disk tiles are interesting from a theoretical point of view, but the toroidal tile boundary introduces too much artifacts in the power spectrum. Corner-based Poisson disk tiles have several advantages over edge-based Poisson disk tiles. Corner-based Poisson disk tiles produce better power spectra even with less tiles, and allow to trade spectral quality for tile set size. Therefore, corner-based Poisson disk tiles seem to be the best tile-based approach.

The spectral properties of the method of Ostromoukhov et al. and Kopf et al. are below average. The radius of the Poisson disk distributions generated with the technique of Kopf et al. is surprisingly low. These techniques should therefore not be used for generating Poisson disk distributions with constant density. However, the real power of these methods is that they are capable of generating nonuniform point distributions. In this chapter, only Poisson disk distributions with constant density were investigated.

We would like to note that different applications often have different requirements. For example, a large radius is very important for object distribution, while spectral properties may not be that important. For sampling, spectral properties are very important, and a slightly smaller radius will not matter that much. Radius statistics, spectral properties and sampling performance are very important criteria for analyzing Poisson disk distributions. However, this does not mean that for a specific application certain methods should be discarded a priori.

Chapter 7

Applications of Poisson Disk Distributions

7.1 Introduction

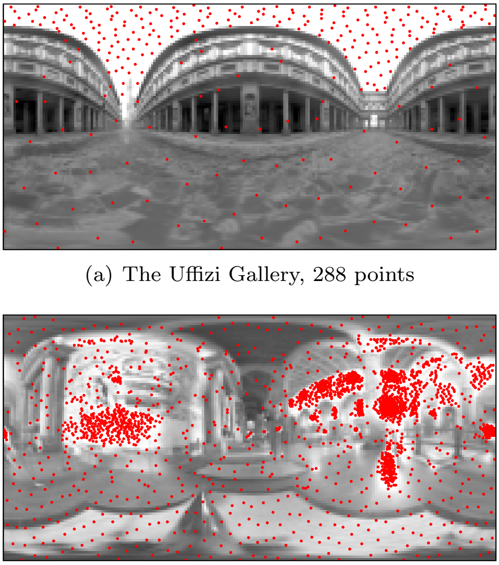
In the previous chapters we have introduced and analyzed efficient methods for generating Poisson disk distributions. Generating Poisson disk distributions is of course not a goal in itself, Poisson disk distributions have several applications in computer graphics. Efficient methods for generating Poisson disk distributions enable efficient implementation of these applications but also enable completely new applications. In this chapter we discuss several applications of Poisson disk distributions.

Overview

This chapter is organized as follows. Section 7.2 discusses sampling. In section 7.3 we discuss applications in non-photorealistic rendering. Section 7.4 introduces applications in scientific visualization. In section 7.5 we discuss procedural modeling, geometric object distribution and geometry instancing. Section 7.6 introduces a new application of Poisson disk distributions in procedural texturing. In section 7.7 we conclude.

7.2 Sampling

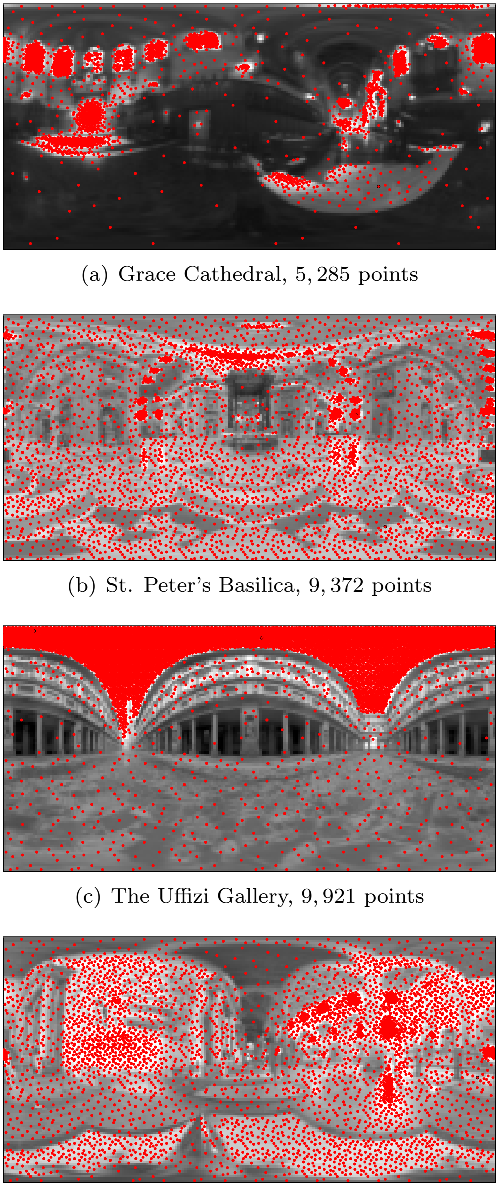
Poisson disk distributions were introduced in the field of computer graphics in the context of sampling. In 1977, Crow identified unwanted artifacts in digitally synthesized images, such as jaggies and moiré patterns, as instances of the aliasing problem from digital signal processing. In the mid-eighties, Dippé and Wold [1985], Cook [1986] and Mitchell [1987] introduced nonuniform sampling and the Poisson disk distribution to turn regular aliasing artifacts into perceptually less objectable stochastic noise. Their work was based on studies by Yellot [Yellot, 1982, 1983], who found that the photoreceptors in the retina of the eye are distributed according to a Poisson disk distribution, and presented theoretical evidence in favor of the Poisson disk distribution. It



(a) The Uffizi Gallery, 288 points

(b) Galileo's Tomb, 3, 200 points

Figure 7.1: Environment map sampling using warped Poisson disk distributed points. The (a) The Uffizi Gallery and (b) Galileo's Tomb environment maps were sampled with (a) 288 and (b) 3, 200 point light sources, by warping Poisson disk distributions generated with edge-based Poisson disk tiles. The sampling patterns were generated in approximately 150 ms. (The environment maps used in this figure are courtesy of Paul Debevec.)



(a) Grace Cathedral, 5, 285 points

(b) St. Peter's Basilica, 9, 372 points

(c) The Uffizi Gallery, 9, 921 points

(d) Galileo's Tomb, 10, 116 points

Figure 7.2: Environment map sampling using a self-similar hierarchical tile. The (a) Grace Cathedral, (b) St. Peter's Basilica, (c) The Uffizi Gallery and (d) Galileo's Tomb environment maps were sampled with (a) 5, 285, (b) 9, 372, (c) 9, 921 and (d) 10, 116 point light sources, using the self-similar hierarchical tile shown in figure 4.42. The sampling patterns were generated in approximately 30 ms. Note the periodicity in regions of constant density. (The environment maps used in this figure are courtesy of Paul Debevec.)

is now generally accepted that because of it's blue noise power spectrum, the Poisson disk distribution is one of the best stochastic sampling patterns.

The Poisson disk sampling pattern allows a better reconstruction of a sampled function than other sampling patterns. This matters a lot for applications in computer graphics, which typically cannot compute enough samples to eliminate aliasing artifacts or stochastic noise. For example, the physically based rendering system of Pharr and Humphreys [2004] uses the best-candidate sampling pattern [Mitchell, 1991], an approximate Poisson disk distribution, to sample the image plane for generating primary rays. However, the Poisson disk sampling pattern is not commonly used for sampling, mainly because it is considered too difficult and too expensive to generate. The efficient methods for generating Poisson disk distributions discussed in the previous chapters enable the use of Poisson disk distributions for sampling, even for interactive and real-time applications.

Importance sampling is one of the most frequently used variance reduction techniques in global illumination and distribution ray tracing [Dutré et al., 2002; Pharr and Humphreys, 2004]. Importance sampling requires nonuniform point distributions. Similar to Poisson disk distributions, nonuniform Poisson disk distributions have significant advantages over other point distributions. An example of importance sampling in the context of global illumination is sampling a high dynamic range environment map, representing an infinite area light source [Cohen and Debevec, 2001; Agarwal et al., 2003; Kollig and Keller, 2003]. The environment map is replaced by a number of point light sources to speed up integration of the incoming illumination. This can be done by warping Poisson disk distributed points according to a probability density function derived from the environment map. Figure 7.1 shows environment maps sampled using warped Poisson disk distributions. Another solution is to directly generate a nonuniform Poisson disk distribution using the technique introduced in section 4.7. Figure 7.2 shows environment maps sampled using this technique. However, both techniques are not optimal. Warping can introduce clumping, and using a single tile introduces periodicity. Efficiently generating sampling patterns with blue noise properties is still a very active area of research [Ostromoukhov et al., 2004; Dunbar and Humphreys, 2006; Kopf et al., 2006].

7.3 Non-Photorealistic Rendering

Non-photorealistic rendering [Gooch and Gooch, 2002] is an area of computer graphics that uses different rendering styles to communicate specific messages. Non-photorealistic rendering is used for artistic media simulation, user-assisted image creation and automatic image creation. Poisson disk distributions have several applications in non-photorealistic rendering.

A pen-and-ink illustration can be generated from a given image by placing a number of primitives, for example points or strokes, according to a density func-

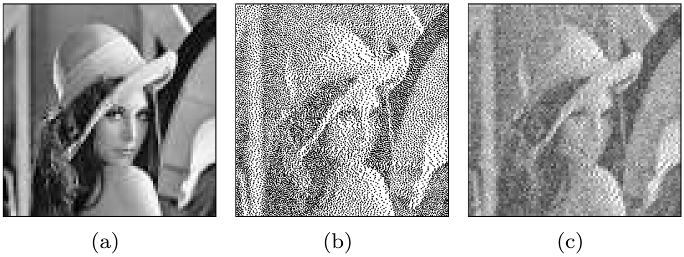


Figure 7.3: Primitive distribution for illustration using warped Poisson disk distributed points. (a) The Lena image. (b) Stippled and (c) hatched non-photorealistic renderings generated from the Lena image, by warping Poisson disk distributions generated with edge-based Poisson disk tiles. Approximately 13, 000 primitives were distributed.

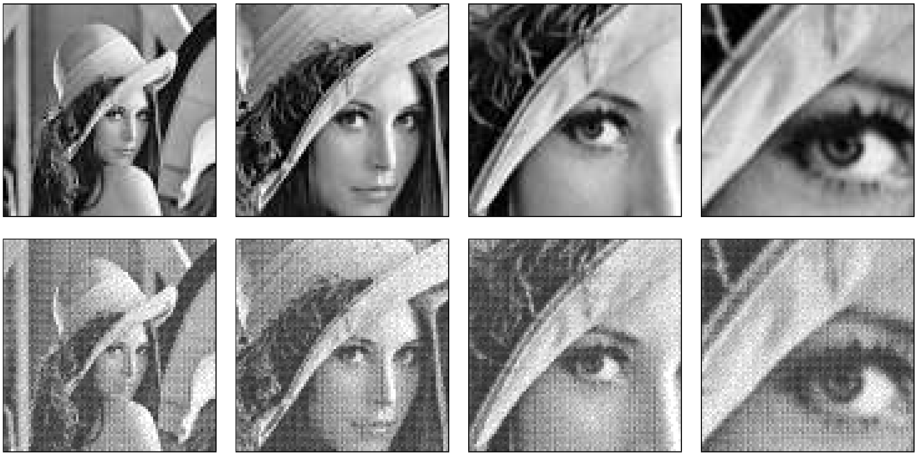


Figure 7.4: Primitive distribution for illustration using a self-similar tile. (top row) The Lena image with several zoom-ins. (bottom row) Stippled non-photorealistic renderings generated from these images using the self-similar hierarchical tile shown in figure 7.4. The point sets consist of respectively 44, 212, 44, 673, 41, 659 and 38, 301 points, and were generated in approximately 80 ms. Note the periodicity in regions of constant density.

tion derived from that image. It is widely accepted in stippling and halftoning that a Poisson disk distribution yields more visually pleasing results [Ulichney, 1987; Deussen et al., 2000; Secord et al., 2002]. However, Poisson disk distributions are not frequently used because they are considered too expensive to generate. Pen-and-ink illustrations can efficiently be generated by warping or redistributing Poisson disk distributed points using the inverse cumulative of the density function. Figure 7.3 shows illustrations generated using warped Poisson disk distributions. Another solution is to directly generate a nonuniform Poisson disk distribution using the technique introduced in section 4.7. Figure 7.4 shows illustrations generated using this technique.

Non-photorealistic rendering also employs several other techniques introduced in previous chapters to simulate artistic styles. For example, Kaplan and Salesin [2000] used the theory of tiling to create images much like the ones by the Dutch artist M. C. Escher, and Hausner [2001] used Lloyd's relaxation method to simulate decorative mosaics.

7.4 Scientific Visualization

Scientific visualization [Tufte, 1986] is a field of research that creates images, diagrams, or animations from complex scientific data. Like non-photorealistic rendering, the goal is to convey specific messages. Poisson disk distributions have several applications in scientific visualization. For example, a vector field can be visualized by sampling the vector field using icons [Tufte, 1986]. The best results are obtained when the icons are placed according to a Poisson disk distribution. Scientific visualization also employs other techniques introduced in previous chapters to create illustrations. For example, Lu and Ebert [2005] used Wang cubes with point distributions to create example-based volume illustrations.

7.5 Procedural Modeling, Geometric Object Distribution and Geometry Instancing

Modeling the real world is an important aspect of computer graphics. However, modeling complex environments such as plant ecosystems or cities by hand can be very time-consuming. Procedural modeling techniques assist the user to create complex environments, or create complex environments automatically. For example, Deussen et al. [1998] proposed a system for creating complex plant ecosystems and Parish and Müller [2001] presented a method for modeling cities.

Geometric object distribution is an important aspect of procedural modeling. Many man-made and natural distributions follow a pattern with a minimum distance criterion. For example the trees in a forest and the individual hairs in

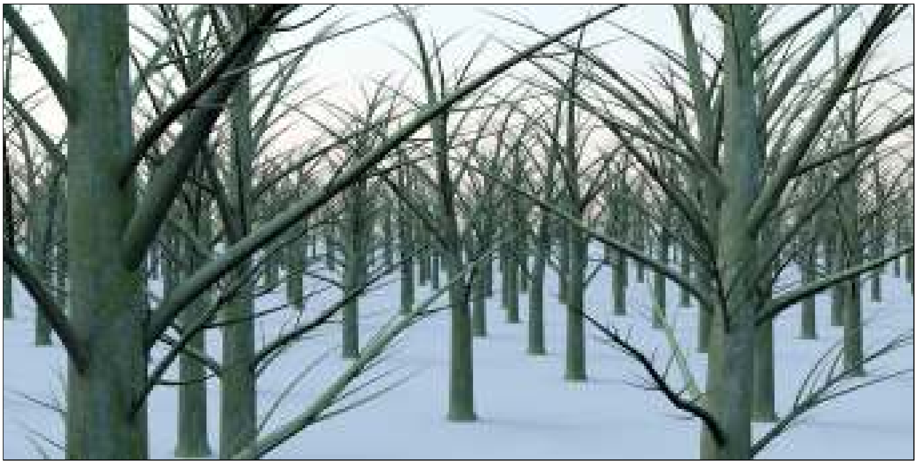


Figure 7.5: A beech forest in the winter. Over $2,000$ instances of 5 beeches were distributed using Poisson disk tiles to create this beech forest. Each beech consists of about $16,000$ triangles. (The beeches were generated with NatFX from Bionatics by Karl vom Berge. The environment map was created using the Utah sky model.)

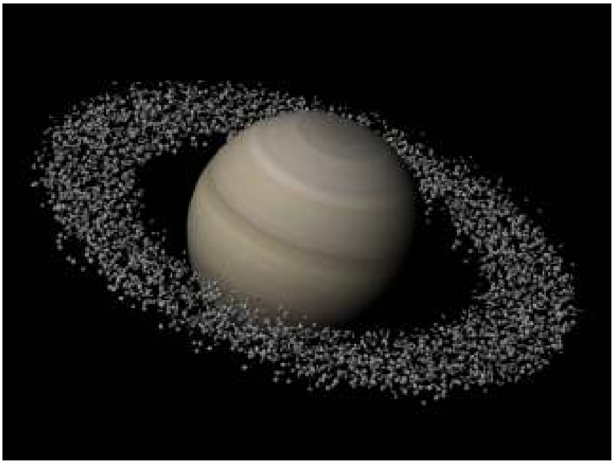


Figure 7.6: A planet with an asteroid belt. The asteroid belt was modeled by instancing several thousand asteroids using a Poisson sphere distribution. (The map of Saturn is courtesy of Björn Jónsson. The asteroid models are courtesy of Scott Hudson.)

fur. These distributions can easily be modeled using Poisson disk distributions. Figure 7.5 shows a beech forest in the winter. The trees were distributed according to a Poisson disk distribution. Figure 7.6 shows a planet with an asteroid belt. The asteroid belt was modeled by cutting out a ring of points from a Poisson sphere distribution.

Geometry instancing is frequently used to efficiently implement geometric object distribution. Instead of using a unique geometric model for each distributed object, only a limited set of geometric models is used, and each distributed object is an instance of one of these models. The instances may have differentiating parameters, such as orientation, size and color. This technique was also used in figures 7.5 and 7.6.

However, for very large or complex environments, placing and storing all instances is still expensive. This problem can be relieved by using the tile-based methods for generating Poisson disk distributions introduced in chapter 4. Because the direct stochastic tiling algorithm allows to efficiently evaluate a Poisson disk distribution locally, it enables on the fly instancing. This eliminates the cost of storing instancing information. This principle could also be used in real-time applications, such as flight simulators or games.

7.6 Procedural Texturing

Texture mapping [Catmull, 1974] is commonly used to increasing the visual complexity of computer-generated images without adding geometric detail. A texture is mapped onto the surface of a shape to add color or detail to the shape. Traditional textures are raster graphics images. Raster graphics images have several disadvantages. Raster graphics images have a fixed resolution and size, and have large storage requirements.

Procedural textures are textures defined by a procedure or an algorithm rather than by a raster graphics image. Compared to traditional textures, procedural textures are compact, have no fixed resolution and size, and can be easily parameterized. Procedural texturing has become an invaluable tool for high-quality image synthesis. Procedural techniques are capable of generating a large variety of convincing textures, such as marble, wood and stone.

At the heart of procedural texturing are texture basis functions. They bootstrap the visual complexity which is present in the generated textures. The most famous texture basis function is Perlin's noise function [Perlin, 1985], or as Peachy states, "the function that launched a thousand textures" [Ebert et al., 2002]. However, the use of texture basis functions is not limited to procedural texturing. Texture basis functions are also used in procedural modeling, shading and animation. This large variety of applications is a motivation to find new texture basis functions and expand the range of textures that can be generated procedurally.

In this section, we present a procedural object distribution function. This new texture basis function distributes procedurally generated objects over a procedurally generated texture, which serves as background. Objects are placed uniformly over the texture, and are guaranteed not to overlap. The texture basis function allows intuitive control over the scale, size and orientation of the objects being distributed, and can be evaluated efficiently. We discuss the history and background of procedural texturing, and present a two-dimensional as well as a tree-dimensional procedural object distribution function.

7.6.1 History and Background

The introduction of solid texturing by Perlin [1985] and Peachy [1985] in the mid-eighties was a milestone in the field of procedural modeling.

The most popular three-dimensional texture basis function is Perlin's noise function [Perlin, 1985; Perlin and Hoffert, 1989; Perlin, 2002]. The noise value at each point is determined by computing a pseudo-random gradient at each of the eight nearest vertices on the integer cubic lattice, followed by splined interpolation. Perlin's noise function has become the standard way to model natural materials such as marble, wood and stone, and natural phenomena such as smoke, water and fire. Although presented in 1985, the Perlin's texture basis function is still heavily used nowadays.

Another useful 3D texture basis function is the cellular texture basis function of Worley [1996]. Random feature points are scattered throughout space, and the function returns the distance to the closest feature points. This process is accelerated using space subdivision: feature points are generated on the fly, in the cubes defined by the integer lattice. Worley's texture basis function is suited for generating rocks, tiled areas, and a variety of organic patterns. Worley introduced his cellular texture basis function in 1996, although a simpler version of this texture basis function was already proposed in 1988 by Burchill.

To address a number of shortcomings of Perlin's noise function, Cook and DeRose [2005] presented wavelet noise, a band-limited version of Perlin's noise function. Their work was inspired by earlier work by Lewis [1989].

There are several other techniques to generate textures procedurally. For example, Turk [1991] presented a biologically inspired method, called reaction-diffusion, that generates interesting mammalian patterns. These methods, however, do not qualify as texture basis functions, because they do not have the semantics of a point evaluation, but require global operations to work.

For an excellent overview of the field of procedural texturing and modeling, we refer to [Ebert et al., 2002].

7.6.2 A 2D Procedural Object Distribution Function

The two-dimensional procedural object distribution function is a new texture basis function that distributes procedurally generated objects over a procedurally generated background.

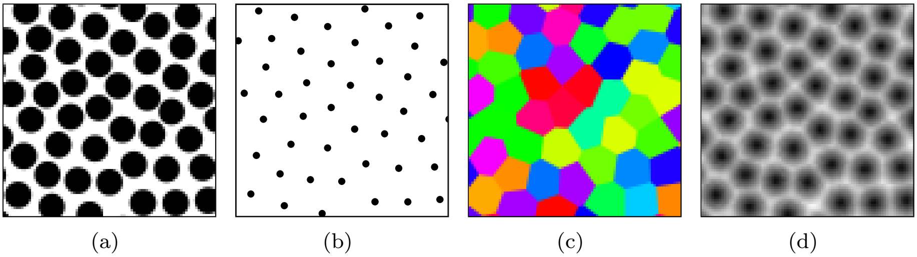


Figure 7.7: Evaluation of the 2D object distribution texture basis function. The texture basis function returns (a) a boolean value indicating whether the point of evaluation is within the Poisson disk of the closest feature point, (b) the coordinates of the closest feature point, (c) a unique ID identifying the closest feature point, and (d) the distance to the closest feature point.

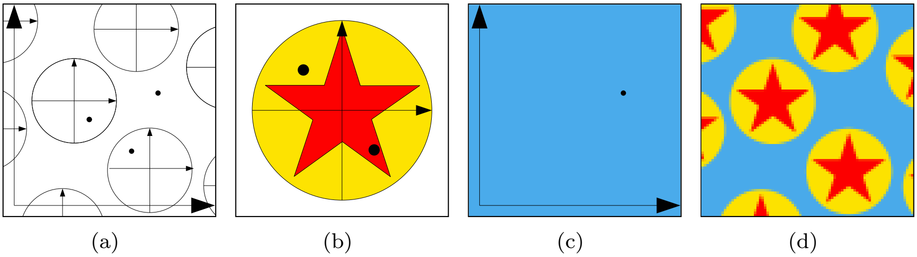


Figure 7.8: Procedural object distribution with the 2D object distribution texture basis function. (a) The texture basis function is evaluated. (b) If the point of evaluation lies within a Poisson disk, it is transformed to the local coordinate system of that disk, and a procedural object is evaluated. (c) If the point of evaluation is not located inside a Poisson disk, a procedural texture which serves as background is evaluated. (d) The resulting procedural texture.

The texture basis function is defined over the infinite plane. When evaluated, it returns the point in a tiled Poisson disk distribution closest to the point of evaluation, and a unique identifier for this point. The function also returns the distance to the closest point, and a boolean value indicating whether the point of evaluation is within the Poisson disk of the closest point. This is illustrated in figure 7.7.

To distribute procedural objects over a procedural background, the texture basis function is evaluated. If the point of evaluation lies within a Poisson disk, it is transformed to the local coordinate system of that disk, and a procedural object is evaluated. If the point of evaluation is not located inside a disk, a procedural texture which serves as background is evaluated. This process is illustrated in figure 7.8.

In the remainder of this subsection, we discuss how to evaluate the texture basis function efficiently, and how to control the placement of the distributed objects. We also present several results and discuss some more advanced topics.

7.6.2.1 Evaluation

Evaluation of the texture basis function is straightforward. The Poisson disk tile that contains the point of evaluation (x, y) is located at tile coordinates $(\lfloor x \rfloor, \lfloor y \rfloor)$, and is provided by the direct stochastic tiling algorithm. The tile and its neighbors are then searched for the closest point. The unique identifier of the closest point is a combination of the hash value of the tile coordinates of the tile where the closest point was found, and the index of the closest point in that tile.

Only a single Poisson disk tile set is needed. Randomness is introduced by the direct stochastic tiling algorithm, randomizing the texture basis function is done by randomizing the permutation table used by the hash function of the tiling algorithm.

The texture basis function can be implemented using edge-based Poisson disk tiles, template Poisson disk tiles or corner-based Poisson disk tiles. Corner-based Poisson disk tiles are recommended because the tile sets are smaller and the tiling algorithms are more efficient.

Several optimizations are employed to evaluate the texture basis function efficiently. After constructing a Poisson disk tile set, the points in the tiles are sorted lexicographically. This speeds up the location of the closest point. Also note that if the distance to a candidate closest point is less than the Poisson disk radius, it must be the closest point. The largest empty circle optimization limits the number of neighboring tiles that has to be searched while locating the closest point. During construction of the Poisson disk tile set, the radius of the largest empty circle r_e is computed. Alternatively, r_e can be bounded analytically. This radius determines different regions in the tile, much like the ones in figure 4.3(a). If the point of evaluation (x, y) is closer to a corner than r_e, three neighboring tiles have to be considered. Else, if (x, y) is closer to an

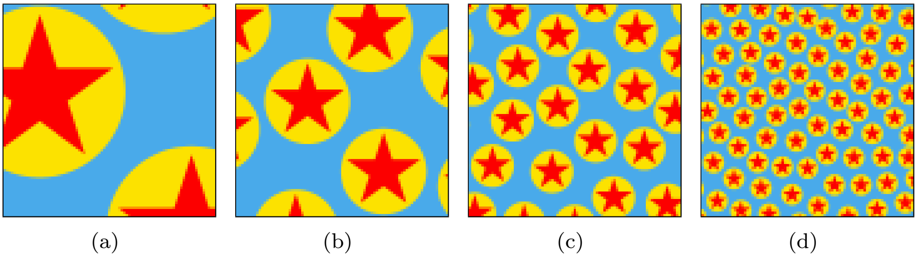


Figure 7.9: Manipulation of the scale s of the 2D object distribution texture basis function. (a) $s = 1$. (b) $s = 4$. (c) $s = 16$. (d) $s = 64$. Note that each image is a closeup of the next one.

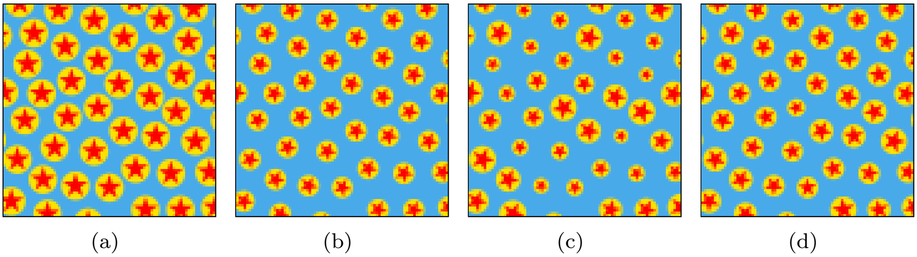


Figure 7.10: Manipulation of the size r and orientation θ of the 2D object distribution texture basis function. (a) $r = 1$, $\theta = 0$. (b) $r = 0.75$, $\theta = \pi/4$. (c) $r \sim U(0.5, 1)$, $\theta \sim N(\pi/4, \pi/32)$. (d) $r \sim N(0.8, 0.05)$, $\theta \sim U(0, 2\pi)$. The scale s of all procedural textures is 36.

edge than r_e, one neighboring tile needs to be considered. In all other cases, the closest point must lie within the same tile as (x, y). This optimization is very effective. For a Poisson disk tile set with $N = 32$ points per tile, and $\alpha = 0.75$, r_e was approximately 0.16. For about 10% of the evaluations, four tiles had to be considered. Roughly 40% of the evaluations required two tiles, and for almost 50% of the evaluations, only a single tile was visited.

Due to these optimizations, the texture basis function can be evaluated very efficiently. In our implementation, one evaluation of the new texture basis function is as expensive as 5 evaluations of Perlin's two-dimensional noise function. This makes our texture basis function also suited for interactive and real-time applications.

7.6.2.2 Parameters

The placement of the distributed objects can be controlled by four parameters: the scale s, the size r, the orientation θ and the aspect ratio a.

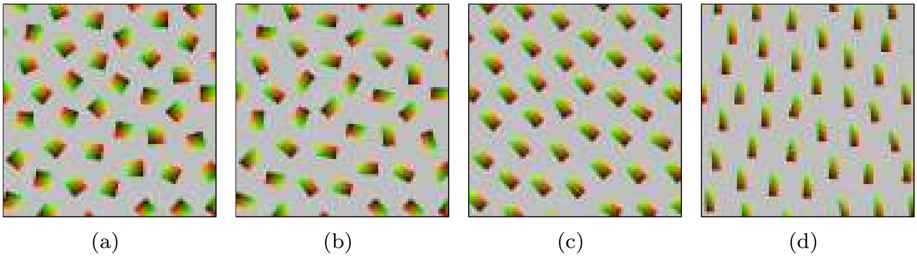


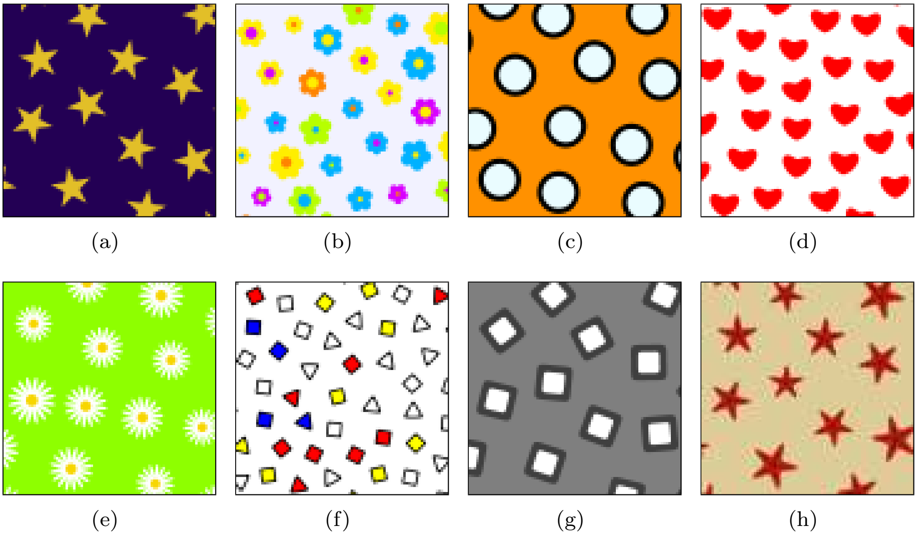
Figure 7.11: Manipulation of the aspect ratio a of the 2D object distribution texture basis function. These procedural textures show a color encoding of the local coordinate systems. (a) $\theta \sim U(0, 2\pi)$, $a = 1$. (b) $\theta \sim U(0, 2\pi)$, $a = \phi$ (the golden ratio, $\phi \approx 1.6180$). (c) $\theta \sim N(\pi/4, \pi/32)$, $a = \phi$. (d) $\theta \sim N(0, \pi/32)$, $a \sim N(2.5, 0.1)$. The scale s and size r of all procedural textures is 36 and 0.80 respectively.

To decouple the texture basis function as much as possible from the underlying tiled Poisson disk distribution, a scale parameter s is introduced that controls the density of objects. A scale of s corresponds to an object density of s objects per unit square. Controlling the scale of the texture basis function is done by scaling the domain over which it is evaluated. To obtain an object density of s, the tiled Poisson disk distribution is scaled by a factor of $\sqrt{S/N}$, where N is the number of points per tile. Figure 7.9 shows a procedural texture for different values of the scale parameter. Note that this scale parameter is different from the original scale parameter introduced in [Lagae and Dutré, 2005a]. The new scale parameter is more intuitive and easier to use.

When the texture basis function is evaluated, and the point of evaluation lies within a disk, it is transformed to the local coordinate system of that disk. These coordinates are then used to evaluate the procedural object. Manipulating the size r and orientation θ of the distributed objects is done by scaling the local coordinate system by a factor $r \in [0, 1]$, and rotating it by an angle $\theta \in [0, 2\pi]$, before evaluating the procedural object. Figure 7.10 shows a procedural texture for different values of the size and orientation parameters.

By introducing the aspect ratio a, a very general and flexible object distribution function is obtained. As figure 7.11 shows, distributions of local coordinate systems can be generated procedurally using only four intuitive parameters. Arbitrary procedural content can be placed in these coordinate systems.

Object attributes, such as size, orientation and aspect ratio, can be chosen at random on a per-object basis. However, some care must be taken. Although each object may have different attributes, all evaluations of the texture basis function involving the same object must produce the same random values for the attributes. This is why the texture basis function provides a unique identifier associated with each disk. When used to seed a random number generator,



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 7.12: Textures generated with the 2D object distribution texture basis function. (a) Stars. (b) Flowers. (c) Polka dots. (d) Hearts. (e) Daisies. (f) Mondriaan shapes. (g) Abstract squares. (h) Starfish.

for example a fast linear congruential generator, random attributes can be generated correctly on a per-object basis. The unique identifier can also be used to generate additional object attributes.

7.6.2.3 Examples

The procedural object distribution function extends the range of textures that can be generated procedurally. Figures 7.12, 7.13 and 7.14 show several procedural textures generated with the new texture basis function. They demonstrate the procedural object distribution function for several settings of the scale, size and orientation parameters. Like all procedural textures, these textures have no fixed resolution and size, and can be easily parameterized.

A lot of interesting procedural objects can be generated with the so called superformula [Gielis, 2003; Gielis et al., 2003]. The heart shape of figure 7.12(d) is based on the polar equation $r(\theta) = \cos 5\theta - 5\cos\theta$. A single petal of a daisy of figure 7.12(e) was created using an exponentiated cosine lobe. The texture of figure 7.12(f) is inspired by Mondriaan's painting *Composition with red, yellow and blue*. The parameters for the texture basis function are $r = 0.8$ and $\theta \sim U(0, 2\pi)$. The rounded triangle is a supershape with parameters $m = 3$, $n_1 = 6.7$, $n_2 = n_3 = 12$ and $a = b = 1$, and the rounded rectangle is a supershape with parameters $m = 4$, $n_1 = n_2 = n_3 = 12$ and $a = b = 1$. The

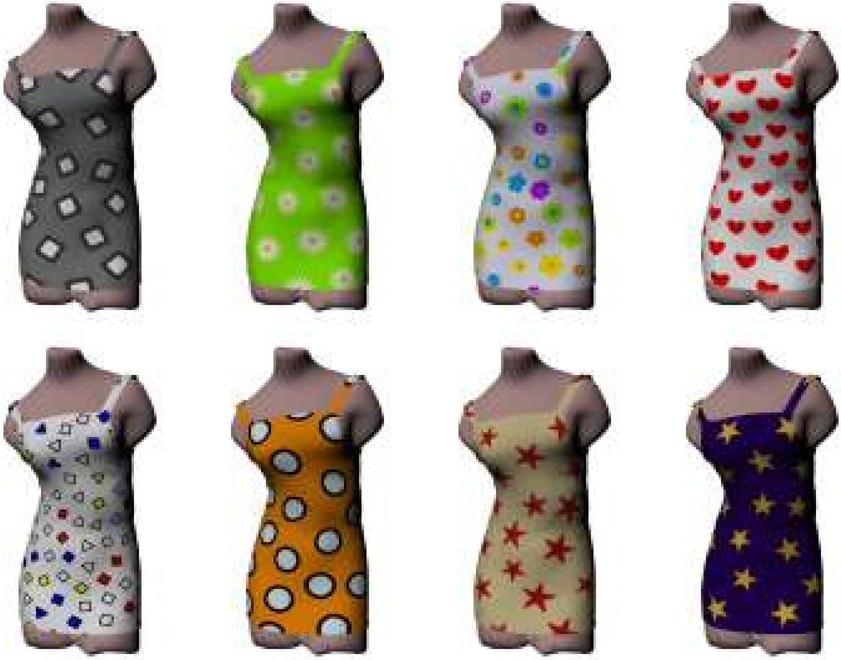


Figure 7.13: Dresses worn by the Venus model. The dresses are textured with the procedural textures shown in figure 7.12.

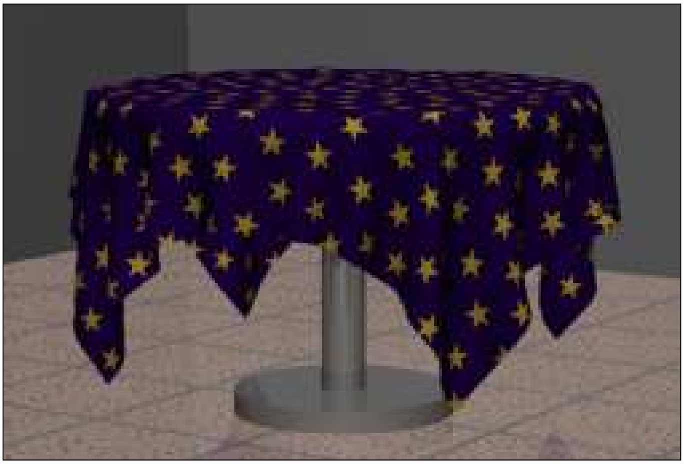


Figure 7.14: A table with a table cloth on a granite floor. The textures used in this scene were generated with the 2D procedural object distribution function.

rounded rectangle of figure 7.12(g) is a supershape with parameters $m = 4$, $n_1 = n_2 = n_3 = 8$ and $a = b = 1$. The starfish of figure 7.12(h) consists of two supershapes. The parameters for the outer one are $m = 5$, $n_1 = 2$, $n_2 = n_3 = 7$ and $a = b = 1$, and the parameters for the inner one are $m = 5$, $n_1 = 2$, $n_2 = n_3 = 13$ and $a = b = 1$. The particles in the granite of figure 7.14 are random convex hexagons. The color of these particles, the color of the mortar and the base color were modulated with Perlin noise.

7.6.2.4 Discussion

By modifying the hash function used in the direct stochastic tiling algorithm, seamless textures can be created. For example, evaluating the hash function with tile coordinates modulo M results in a periodic tiling with period M, and can be used to produce a toroidally wrapping texture to cover a cylinder or texture.

The procedural object distribution function is somewhat similar to the cellular texture basis function of Worley [1996]. However, the cellular texture basis function of Worley uses feature points randomly scattered in space, and therefore cannot be used to distribute objects without overlap.

In general, most texture basis functions generate some kind of pseudo-random scalar value over their domain. From that perspective, the procedural object distribution function is not a typical texture basis function. However, the ultimate goal of all texture basis functions is the same: providing a solid basis for generating a large variety of textures. The procedural object distribution function does just that.

7.6.3 A 3D Procedural Object Distribution Function

Solid textures [Perlin, 1985; Peachy, 1985] are three-dimensional textures that simulate solid materials. When a solid texture is applied to the surface of an object, the object appears to be carved out of that material. Most texture basis functions are available in two as well as three dimensions.

The two-dimensional procedural object distribution function easily extends to three dimensions using three-dimensional corner tiles and corner-based Poisson sphere tiles (see section 4.6). Figure 7.15 shows the outputs of the three-dimensional texture basis function.

The three-dimensional procedural object distribution function is good at modeling natural materials with particle distributions, such as granite, and abstract man-made patterns. Figure 7.16 shows several procedural solid textures generated with the texture basis function. The texture basis function has a small memory footprint and is quite efficient: one evaluation is about as expensive as 20 evaluations of Perlin's Noise function. Figure 7.17 shows how we integrated the procedural object distribution functions into a commercial rendering system.

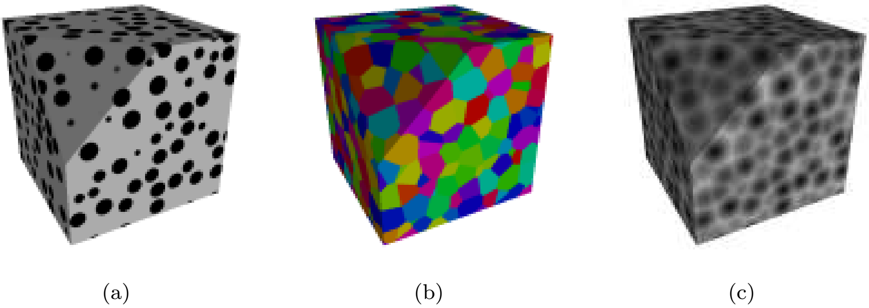


Figure 7.15: Evaluation of the 3D object distribution texture basis function. The texture basis function returns (a) a boolean value indicating whether the point of evaluation is within the Poisson disk of the closest feature point, (b) a unique ID identifying the closest feature point, and (c) the distance to the closest feature point. The coordinates of the closest feature point (not shown) are also returned.

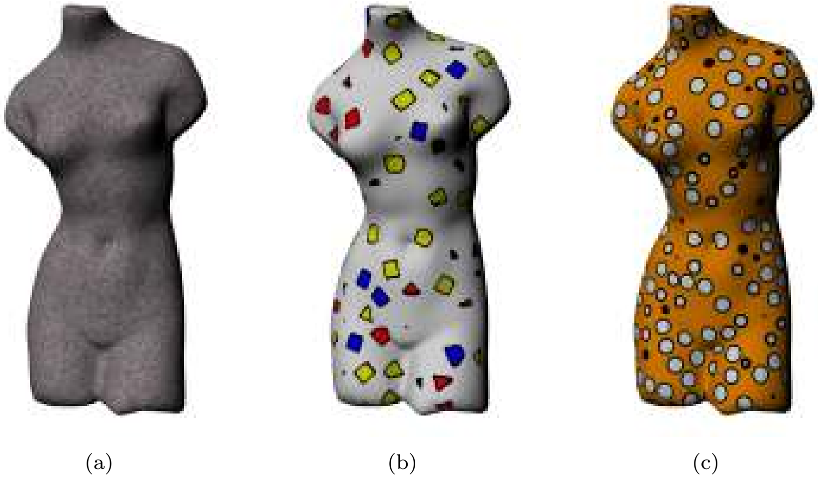


Figure 7.16: The Venus model carved from solid textures generated with the 3D object distribution texture basis function. (a) Granite. (b) Mondriaan shapes. (c) Polka dots.

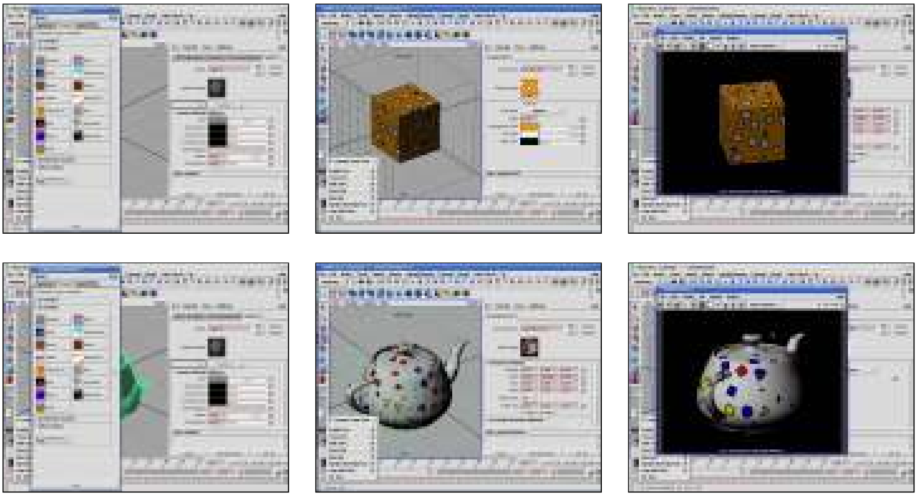


Figure 7.17: Integration of the object distribution texture basis function in a commercial rendering system.

7.7 Conclusion

In this chapter we have discussed applications of Poisson disk distributions in sampling, non-photorealistic rendering, scientific visualization and procedural modeling. We have also introduced a procedural object distribution function, a new texture basis function that extends the range of textures than can be generated procedurally. We have shown that Poisson disk distributions are a general tool in computer graphics, and that the tile-based methods for generating Poisson disk distributions introduced in the previous chapters can be used to improve existing applications but also enable new applications.

Chapter 8

Small Aperiodic Sets of Corner Tiles

8.1 Introduction

Wang tiles originated in the field of discrete mathematics and were first used to study the tiling problem and aperiodic tile sets. Later, Wang tiles were introduced in the field of computer graphics to facilitate the generation of complex non-periodic signals. Corner tiles originated in the field of computer graphics as an alternative for Wang tiles. In this chapter, we study corner tiles in the original context of the tiling problem and aperiodic tile sets. We construct small aperiodic sets of corner tiles to show that corner tiles are not just an ad hoc extension of Wang tiles, and that corner tiles are sound also from a discrete mathematical point of view.

Overview

This chapter is organized as follows. Section 8.2 sketches the history and background of aperiodic tile sets. In section 8.3 we show how to construct small aperiodic corner tile sets from small aperiodic Wang tile sets, and section 8.4 shows how to do the inverse. In section 8.5 we conclude.

8.2 History and Background

In 1961, Wang studied the tiling problem with Wang tiles and conjectured that if a set of tiles can tile the plane, then they can always be arranged to do so periodically. This conjecture was refuted in 1966 by Berger, who showed that the tiling problem was undecidable and constructed the first aperiodic tile set, consisting of $20,426$ Wang tiles. This was one of the most remarkable discoveries in the theory of tilings.

This number was greatly reduced to 104 by Berger in 1966 [Berger, 1966] and subsequently to 92 by Knuth in 1968 [Knuth, 1968], to 40 by Läuchli in 1966 [Wang, 1975], to 52 by Robinson in 1967 [Robinson, 1967], to 32 by Penrose, to 24 by Robinson in 1977, to 14 by Kari in 1996 [Kari, 1996] and finally to 13

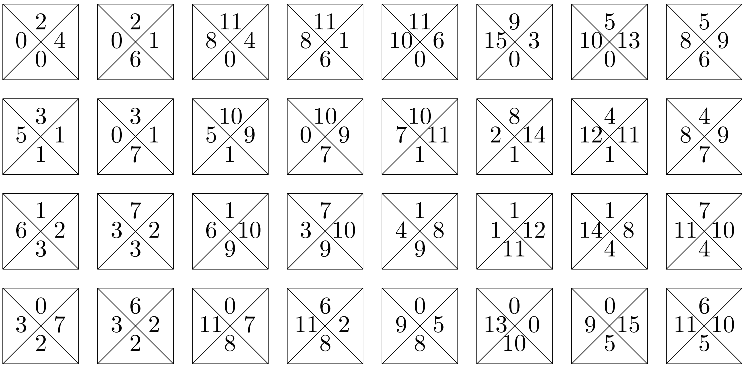


Figure 8.1: An aperiodic Wang tile set of 32 tiles over 16 colors.

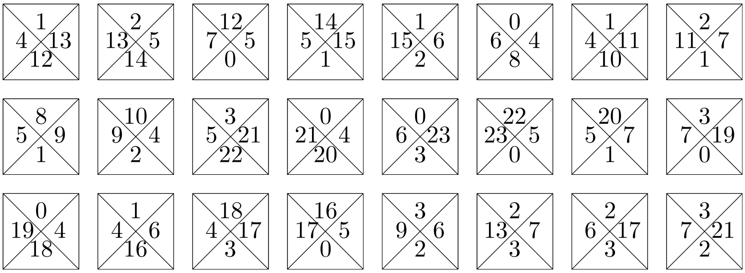


Figure 8.2: An aperiodic Wang tile set of 24 tiles over 24 colors.

by Culik in 1996 [Culik, 1996]. For an excellent discussion of these results, we refer to Grünbaum and Shepard [1986, chapters 10 and 11].

Not only Wang tiles allow the construction of aperiodic tile sets. Since the late sixties, several aperiodic tile sets have been discovered. In 1974, Penrose discovered his famous kite and dart, an aperiodic set of only two tiles. Whether a single aperiodic tile exists is still an open question.

8.3 Construction of Aperiodic Corner Tile Sets from Aperiodic Wang Tile Sets

Because Wang tiles and corner tiles are so closely related, we construct aperiodic sets of corner tiles using isomorphisms between Wang tilings and corner tilings. In this section, we present five such construction methods: diagonal translation, horizontal translation, vertical translation, rotation and subdivision.

The aperiodic corner tile sets we construct are based on small aperiodic Wang tile sets. We use the following five aperiodic Wang tile sets.

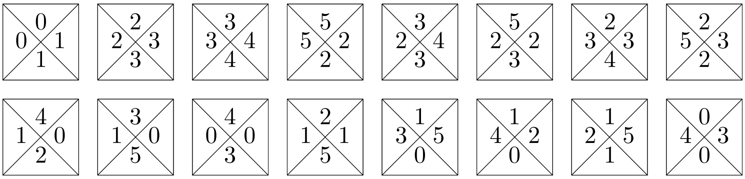


Figure 8.3: An aperiodic Wang tile set of 16 tiles over 6 colors.

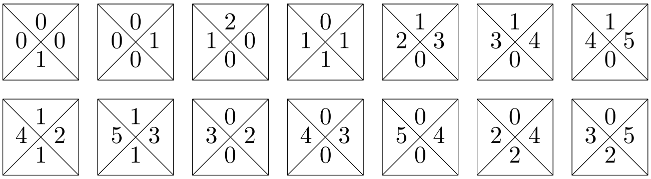


Figure 8.4: An aperiodic Wang tile set of 14 tiles over 6 colors.

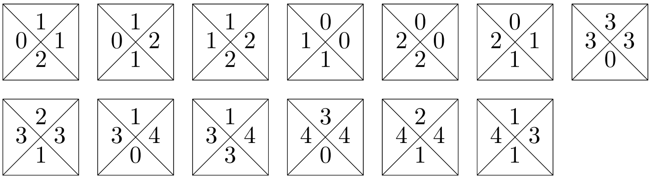


Figure 8.5: An aperiodic Wang tile set of 13 tiles over 5 colors. This is the smallest aperiodic Wang tile set currently known.

Wang tile set	diagonal translation	horizontal translation	vertical translation	rotation	subdivision
13/5	125/13	failed	failed	60(13+47)/9	52/19
14/6	214/14	failed	86/6	94(14+80)/9	56/21
16/6	87/16	44/6	44/6	49(16+33)/12	64/23
24/24	203/24	62/12	72/12	67(24+43)/24	96/49
32/16	114/32	failed	failed	90(32+58)/28	128/49

Table 8.1: The size of aperiodic corner tile sets constructed with the proposed construction methods. For the rotation method, the number of white and black tiles is also indicated.

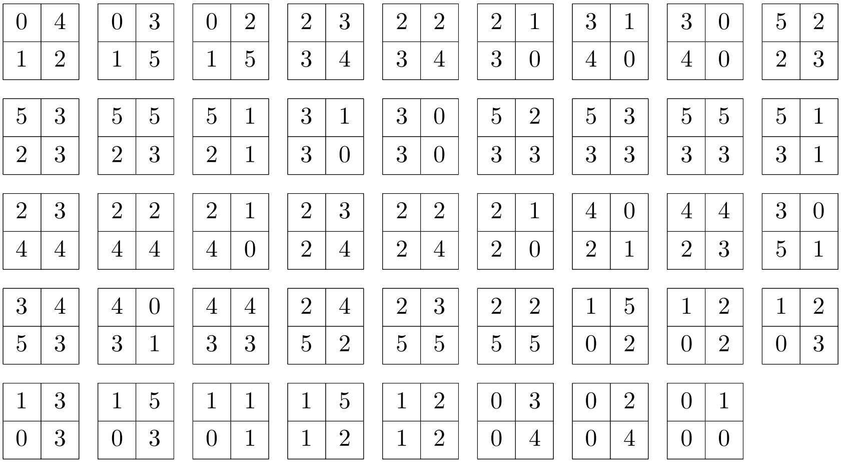


Figure 8.6: An aperiodic corner tile set of 44 tiles over 6 colors. This tile set was constructed from the aperiodic set of 16 Wang tiles over 6 colors shown in figure 8.3, using the horizontal translation construction method. This is the smallest aperiodic corner tile set currently known.

- The aperiodic set of 32 Wang tiles over 16 colors, obtained by cutting up a tiling by Penrose kites and darts [Grünbaum and Shepard, 1986, page 593]. This tile set is shown in figure 8.1.

- The aperiodic set of 24 Wang tiles over 24 colors, constructed from the Ammann prototiles (set A2) [Grünbaum and Shepard, 1986, page 593]. This tile set is shown in figure 8.2.

- The aperiodic set of 16 Wang tiles over 6 colors, generated using the Ammann set A2 and Ammann bars [Grünbaum and Shepard, 1986, page 595]. This tile set is shown in figure 8.3.

- The aperiodic set of 14 Wang tiles over 6 colors, constructed in 1996 by Kari with a method based on Mealy machines that multiply Beatty sequences of real numbers by rational constants. This tile set is shown in figure 8.4.

- The aperiodic set of 13 Wang tiles over 5 colors, created also in 1996 by Culik using a construction method based on the method developed by Kari. This tile set is shown in figure 8.5.

The colors of each aperiodic Wang tile set were relabeled to form an integer sequence starting with 0 and ending with the number of colors minus one.

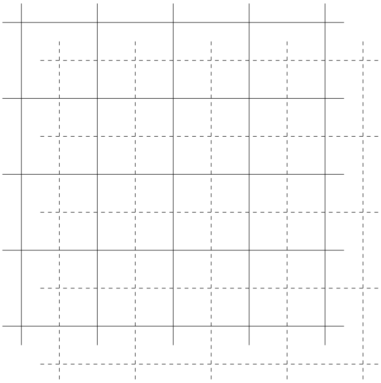


Figure 8.7: The diagonal translation method for constructing an aperiodic corner tile set. The lattice of the corner tiles (dashed lines) is translated diagonally with respect to the lattice of the Wang tiles (solid lines).

8.3.1 Diagonal Translation

The diagonal translation construction method works for an arbitrary aperiodic set of Wang tiles.

The corner tiles are placed on a lattice translated diagonally with respect to the lattice of the Wang tiles, as shown in figure 8.7. Each Wang tile is given a distinct color, and the edge colors of the Wang tiles are ignored. Each corner of each corner tile now receives the color of the Wang tile it lies on. The corner tile set constructed with this method consists of a single tile for each valid two by two square configuration of Wang tiles, and the number of colors used by the corner tile set equals the number of tiles in the Wang tile set. If the Wang tile set is aperiodic, then the corner tile set will clearly also be aperiodic.

The second column of table 8.1 summarizes the results obtained with this construction method.

8.3.2 Horizontal and Vertical Translation

The horizontal and vertical translation construction methods are very similar to the diagonal translation construction method, but now the corner tiles are placed on a lattice translated horizontally or vertically with respect to the lattice of the Wang tiles, as shown in figure 8.8. A corner of a corner tile now receives the edge color of the edge of the Wang tile it lies on. In general, this is not an isomorphism, because all vertical or horizontal edges of the Wang tiles are ignored. However, for certain aperiodic Wang tile sets, this morphism is bijective.

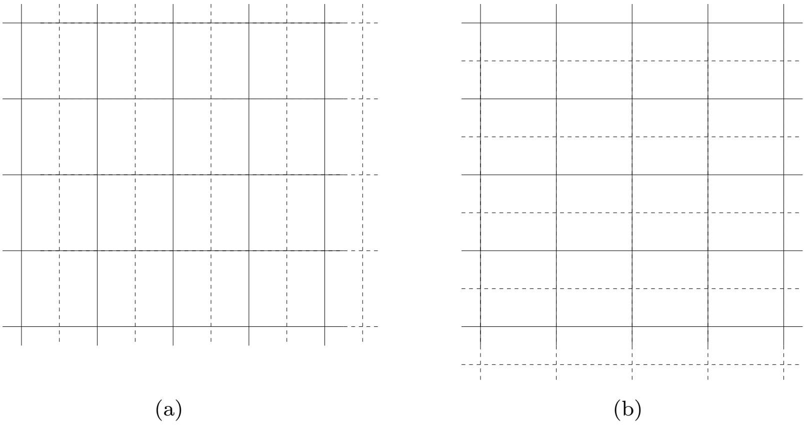


Figure 8.8: The (a) horizontal and (b) vertical translation methods for constructing an aperiodic corner tile set. The lattice of the corner tiles (dashed lines) is translated (a) horizontally and (b) vertically with respect to the lattice of the Wang tiles (solid lines).

The third and fourth column of table 8.1 summarize the results obtained with this construction method. The aperiodic set of 44 corner tiles over 6 colors obtained using the horizontal translation method is shown in figure 8.6.

8.3.3 Rotation

The rotation construction method is somewhat more complicated than previous methods. The corner tiles are placed on a lattice rotated 45 degrees counterclockwise with respect to the lattice of the Wang tiles, as shown in figure 8.9. A corner of a corner tile receives the edge color of the edge of the Wang tile it lies on. However, the rotation results in two kinds of corner tiles: white tiles, corresponding to a single Wang tile, and black tiles, corresponding to a 2 by 2 square configuration of Wang tiles. The white and black tiles follow a checkerboard pattern. A corner tiling constructed this way cannot have a period that maps white tiles onto white tiles (or black tiles onto black tiles), because the Wang tiling is aperiodic, and a period that maps black tiles onto white tiles, or vice versa, is also impossible, because twice that period maps white tiles onto white tiles. We only need to enforce that all valid tilings with the corner tile set follow the checkerboard pattern. This can be done by superimposing on each corner tile one of the tiles in figure 8.10. As shown in figure 8.11, this checkers corner tile set enforces a checkerboard pattern. This operation effectively doubles (at most) the number of colors of the corner tile set.

The fifth column of table 8.1 summarizes the results obtained with this con-

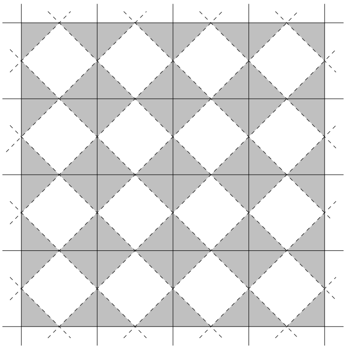
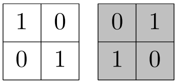


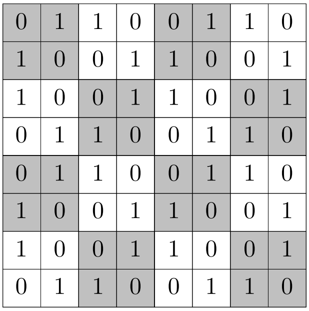
Figure 8.9: The rotation method for constructing an aperiodic corner tile set. The lattice of the corner tiles (dashed lines) is rotated with respect to the lattice of the Wang tiles (solid lines). This results in two kinds of tiles, black tiles and white tiles.



1	0
0	1

0	1
1	0

Figure 8.10: A checkers corner tile set. This tile set enforces a checkerboard pattern.



0	1	1	0	0	1	1	0
1	0	0	1	1	0	0	1
1	0	0	1	1	0	0	1
0	1	1	0	0	1	1	0
0	1	1	0	0	1	1	0
1	0	0	1	1	0	0	1
1	0	0	1	1	0	0	1
0	1	1	0	0	1	1	0

Figure 8.11: A tiling with the checkers corner tile set.

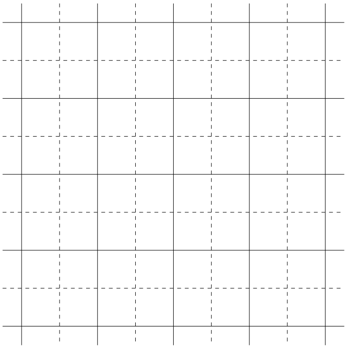


Figure 8.12: The subdivision method for constructing an aperiodic corner tile set. The lattice of the corner tiles (dashed lines) is obtained by subdividing the lattice of the Wang tiles (solid lines), such that each Wang tile corresponds with four corner tiles.

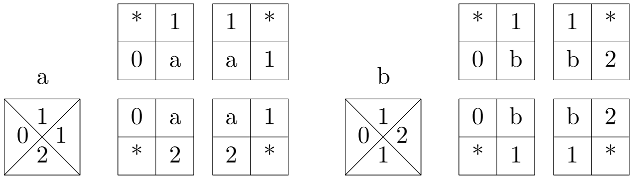


Figure 8.13: Examples of the subdivision method for constructing an aperiodic corner tile set. Two tiles from the aperiodic set of 13 Wang tiles over 5 colors, numbered a and b, and the corner tiles they produce. The star is a new color.

struction method.

8.3.4 Subdivision

The final construction method we discuss is subdivision.

The corner tiles are placed on a lattice obtained by subdividing the lattice of the Wang tiles, as shown in figure 8.12. Each Wang tile corresponds to four corner tiles. The corners that lie on the middle of an edge of a Wang tile receive the color of that edge. The corners that lie in the center of a Wang tile are colored with a color that uniquely determines that Wang tile. One additional color is assigned to the rest of the colors of all corner tiles. This procedure is illustrated in figure 8.13. The corner tile set obtained this way counts four times as much tiles as the Wang tile set. The number of colors is equal to the sum of the number of Wang tiles and the number of colors used in the Wang

Wang tile set	diagonal translation	horizontal translation	vertical translation	rotation	subdivision
13/5	125/50	failed	failed	60/21	52/36
14/6	214/86	failed	86/36	94/23	56/40
16/6	87/44	44/24	44/24	49/32	64/44
24/24	203/72	62/42	72/38	67/38	96/72
32/16	114/84	failed	failed	90/52	128/96

Table 8.2: The size of aperiodic Wang tile sets constructed with the proposed construction methods. Each entry in this table is the number of tiles and the number of colors of the aperiodic Wang tile set constructed from the aperiodic corner tile set at the corresponding position in table 8.1.

tile set plus one. If the Wang tile set is aperiodic, then the corner tile set will also be aperiodic.

The sixth column of table 8.1 summarizes the results obtained with this construction method.

Note that the additional color can be one of the edge colors used in the Wang tile set (but not one of the colors that uniquely determine the Wang tiles). Also note that some tiles can be eliminated by grouping certain colors that uniquely determine the Wang tiles. For example, one of the corner tiles in figure 8.13 is eliminated in by merging the colors a and b. We were able to reduce the number of tiles mentioned in table 8.1, but we have not succeeded in constructing a tile set smaller than the one shown in figure 8.6.

This construction method was recently used by Lukkarila [2006] to show that the square tiling problem is NP-complete for deterministic tile sets.

8.4 Construction of Aperiodic Wang Tile Sets from Aperiodic Corner Tile Sets

Constructing an aperiodic Wang tile set from an aperiodic corner tile set is done by encoding each unique combination of two corner colors along a horizontal or vertical edge into a single new edge color. The number of tiles in the aperiodic Wang tile set constructed with this method is the same as the number of tiles in the corner tile set, and the number of colors is squared at most. Table 8.2 shows the size of the resulting aperiodic Wang tile sets for each of the aperiodic corner tile sets of table 8.1.

From an aperiodic Wang tile set created this way we can again construct an aperiodic corner tile set. Table 8.3, 8.4, 8.7, 8.5 and 8.6 show sequences of aperiodic tile sets constructed by repeated application of the diagonal translation, horizontal and vertical translation, rotation and subdivision construction methods. Note that the number of tiles in the tile sets is not always an increasing

13/5	14/6	16/6	24/24	32/16
125/13	214/14	87/16	203/24	114/32
125/50	214/86	87/44	203/72	114/84
1240/125	3052/214	286/87	540/203	280/114
1240/418	3052/1208	286/160	540/326	280/204
14810/1240	58910/3052	777/286	2068/540	662/280
14810/4523	58910/19274	777/464	2068/990	662/512
160168/14810	1213029/58910	1642/777	7092/2068	1124/662
...	...	...	...	...

Table 8.3: The size of aperiodic Wang and corner tile sets constructed by repeated application of the diagonal translation method.

13/5	14/6	16/6	24/24	32/16
failed	failed	44/6	62/12	failed
		44/24	62/42	
		138/24	156/42	
		138/82	156/104	
		438/82	436/104	
		438/271	436/272	
		1406/271	1154/272	
		...	...	

Table 8.4: The size of aperiodic Wang and corner tile sets constructed by repeated application of the horizontal translation method.

13/5	14/6	16/6	24/24	32/16
60/9	94/9	49/12	67/24	90/28
60/21	94/23	49/32	67/38	90/52
126/42	210/46	113/52	171/76	212/104
126/84	210/136	113/80	171/118	212/164
336/149	596/204	257/160	367/224	464/308
336/228	596/372	257/226	367/292	464/386
702/456	1306/744	545/416	801/584	972/772
...	...	...	...	...

Table 8.5: The size of aperiodic Wang and corner tile sets constructed by repeated application of the rotation method.

13/5	14/6	16/6	24/24	32/16
52/19	56/21	64/23	96/49	128/49
52/36	56/40	64/44	96/72	128/96
208/89	224/97	256/109	384/169	512/225
208/176	224/192	256/216	384/336	512/448
832/385	896/417	1024/473	1536/721	2048/961
832/768	896/832	1024/944	1536/1440	2048/1920
3328/1601	3584/1729	4096/1969	6144/2977	8192/3969
...	...	...	...	...

Table 8.6: The size of aperiodic Wang and corner tile sets constructed by repeated application of the subdivision method.

13/5	14/6	16/6	24/24	32/16
failed	86/6	44/6	72/12	failed
	86/36	44/24	72/38	
	504/36	138/24	192/38	
	504/216	138/82	192/122	
	2968/216	438/82	564/122	
	2968/1272	438/271	564/378	
	17464/1272	1406/271	1584/378	
	...	...	...	

Table 8.7: The size of aperiodic Wang and corner tile sets constructed by repeated application of the vertical translation method.

sequence.

8.5 Conclusion

In this chapter we have studied corner tiles in the context of the tiling problem and aperiodic tile sets. We have shown how to construct an aperiodic set of square tiles with colored corners from an aperiodic set of Wang tiles, and vice versa. We have constructed several new aperiodic sets of Wang tiles and corner tiles. The smallest set of corner tiles we have constructed consists of 44 tiles over 6 colors. We have shown that, if W and C are the cardinalities of the smallest aperiodic Wang tile set and the smallest aperiodic corner tile set, then $W \leq C \leq 4W$. Grünbaum and Shepard [1986] state that "every new aperiodic tile set is clearly of interest". We hope that the wide range of aperiodic tile sets we construct in this chapter sheds some new light on aperiodic tilings.

Chapter 9

Conclusion

9.1 Summary

A common problem in the field of computer graphics is the synthesis and storage of complex signals, such as point distributions or textures. In this dissertation we have presented tile-based methods to solve this problem. Instead of synthesizing a complex signal directly, the signal is synthesized over a small set of tiles. Arbitrary large amounts of that signal can then be generated very efficiently simply by generating a stochastic tiling.

We have introduced corner tiles as an alternative for Wang tiles. In contrast with Wang tiles, corner tiles also constrain the four diagonally neighboring tiles, and are therefore not subject to the corner problem. We have revisited the most important applications of Wang tiles, and we have shown that corner tiles have substantial advantages for each of these applications.

We have presented direct stochastic tiling algorithms for Wang tiles and corner tiles. In contrast with scanline stochastic tiling algorithms, direct stochastic tiling algorithms are capable of evaluating a stochastic tiling locally, without explicitely constructing and storing the tiling up to that point. We have also presented long-period hash functions for direct stochastic tiling algorithms.

We have demonstrated tile-based methods for generating Poisson disk distributions and for synthesizing textures. Tile-based methods not only allow to efficiently generate Poisson disk distributions or synthesize textures, but also enable new applications such as tile-based texture synthesis and a procedural object distribution function. This new texture basis function allows to distribute procedural objects over a procedural background, using intuitive parameters such as the scale, size and orientation of the objects.

We have presented an overview of applications of tiled Poisson disk distributions, and a detailed comparison of methods for generating Poisson disk distributions. We have also studied corner tiles in the context of the tiling problem and aperiodic tile sets, and we have introduced several new aperiodic sets of Wang tiles and corner tiles.

The tile-based methods we have presented in this dissertation are a valuable tool for computer graphics, and help to keep up with the continuously increasing demand for more complexity and realism in digitally synthesized images.

9.2 Contributions

This is a detailed overview of the contributions of this dissertation. A complete list of publications is provided on page 173.

Chapter 2

- The concept of corner tiles is an original contribution. Corner tiles were presented in [Lagae and Dutré, 2006a]. In concurrent work, Ng et al. [2005] presented ω-tiles, a patch combination strategy for texture tiles (see section 5.3), which basically are corner tiles.
- The enumeration schemes for Wang tile sets and corner tile sets (see section 2.8) are original contributions. The enumeration scheme for corner tile sets was presented in [Lagae and Dutré, 2006a], although we already used the enumeration scheme for Wang tile sets since [Lagae and Dutré, 2005a].

Chapter 3

- The scanline stochastic tiling algorithm for corner tiles (see subsection 3.2.2) is an original contribution. This algorithm was presented in [Lagae and Dutré, 2006a]. However, the algorithm is only a simple extension of the scanline stochastic tiling algorithm for Wang tiles of Cohen et al. [2003] (see subsection 3.2.1).
- The concept of direct stochastic tiling algorithms (see section 3.3) is an original contribution. The direct stochastic tiling algorithm for corner tiles (see subsection 3.3.1) was presented in [Lagae and Dutré, 2006a]. The direct stochastic tiling algorithm for Wang tiles using two hash functions and the direct stochastic tiling algorithm for compact sets of Wang tiles (see subsection 3.3.2) were presented in [Lagae and Dutré, 2005a]. The direct stochastic tiling algorithm for Wang tiles using a hash function defined at the tile edges (see subsection 3.3.2) was not published before. In concurrent work, Wei [2004] presented a tiling algorithm for Wang tiles similar in spirit to the direct stochastic tiling algorithms presented in this dissertation.
- Long-period hash functions based on permutation tables (see section 3.4) are an original contribution. This family of hash functions was presented in [Lagae and Dutré, 2006d].

Chapter 4

- The relative radius specification scheme for Poisson disk distributions (see subsection 4.2.3) is an original contribution. This radius specification scheme was presented in [Lagae and Dutré, 2005a], and reformulated in terms of packing density in [Lagae and Dutré, 2006e].

- Edge-based Poisson disk tiles (see section 4.3) are an original contribution. Edge-based Poisson disk tiles were presented in [Lagae and Dutré, 2005a]. At that time, edge-based Poisson disk tiles were the only method capable of efficiently generating high-quality Poisson disk distributions.

- Template Poisson disk tiles (see section 4.4) are an original contribution. Template Poisson disk tiles were presented in [Lagae and Dutré, 2005b].

- Corner-based Poisson disk tiles (see section 4.5) are an original contribution. Corner-based Poisson disk tiles were presented in [Lagae and Dutré, 2006a].

- The relative radius specification scheme for Poisson sphere distributions, three-dimensional corner tiles, and corner-based Poisson disk tiles (see section 4.6) are original contributions. These were presented in [Lagae and Dutré, 2006e].

- The tile-based method for generating nonuniform Poisson disk distributions (see section 4.7) is an original contribution. This method was presented in [Lagae and Dutré, 2006c]. In concurrent work, Kopf et al. [2006] presented a somewhat similar technique.

Chapter 5

- The methods for tile-based texture synthesis (see section 5.3) were developed by Cohen et al. [2003], Burke, and Ng et al. [2005]. The texture tile sets based on corner tiles were generated with the help of Wang Yue, Tuen-Young Ng and Tiow-Seng Tan from the National University of Singapore, using the method of Ng et al. [2005].

- The tile-based texture mapping algorithm based on corner tiles (see section 5.4) is an original contribution. This algorithm was presented in [Lagae and Dutré, 2006a]. The tile-based texture mapping algorithm based on corner tiles is a revision of the tile-based texture mapping algorithm based on Wang tiles of Wei [2004].

- The solutions to the corner tile packing problem (see section 5.5.3) and the puzzles derived from the tile packing problem (see section 5.5.4) are original contributions. These were presented in [Lagae and Dutré, 2006a,f]. The Wang tile packing problem (see section 5.5.2) was solved by Wei [2004].

Chapter 6

- The comparison of methods for generating Poisson disk distributions is an original contribution. Most of this comparison is presented in [Lagae and Dutré, 2006b]. The sampling performance results were not published before.

Chapter 7

- The applications discussed in this chapter, with the exception of procedural texturing (see section 7.6), were already known. However, tile-based Poisson disk distributions open up new possibilities for each of these applications.

- The procedural object distributions functions (see section 7.6) are original contributions. The two-dimensional procedural object distribution function was presented in [Lagae and Dutré, 2005a], and the three-dimensional procedural object distribution function was presented in [Lagae and Dutré, 2006e].

Chapter 8

- The methods for constructing aperiodic corner tile sets from aperiodic Wang tile sets (see section 8.3), the method for constructing aperiodic Wang tile sets from aperiodic corner tile sets (see section 8.4), and the aperiodic corner tile sets and Wang tile sets presented in this chapter are original contributions. These are presented in [Lagae et al., 2006]. This work was done in collaboration with Jarkko Kari from the Department of Mathematics of the University of Turku.

9.3 Future Work

This dissertation provides a complete set of workable algorithms for tile-based methods using corner tiles. However, some problems remain unsolved. We have not presented a direct stochastic tiling algorithm for compact sets of corner tiles (see section 3.3), and we have not found a constructive method for solving the corner tile packing problem (see section 5.5.3). We hope that these open problems will eventually be solved.

There are several opportunities for future work.

- Efficiently generating nonuniform Poisson disk distributions remains a challenging problem. This problem was already explored by Ostromoukhov et al. [2004], Kopf et al. [2006], and in section 4.7, but none of the presented solutions is satisfactory.

- Exploring more advanced texture synthesis methods, such as the one of Liu et al. [2004] and Kwatra et al. [2005], for synthesizing a texture over a set of Wang tiles or corner tiles could be useful to further improve the quality of tile-based texture synthesis.
- In this dissertation we have applied tile-based methods to Poisson disk distributions and textures. It would be interesting to apply tile-based methods to other kinds of complex signals.
- A promising venue for further research is tiling complex signals over a mesh, in the spirit of Neyret and Cani [1999] and Fu and Leung [2005]. Tiling complex signals over other domains, such as a spherical domain, would also be interesting.

We hope that this dissertation will inspire future work, and that future tile-based methods will consider corner tiles as a viable alternative for Wang tiles.

Bibliography

Agarwal, S., Ramamoorthi, R., Belongie, S., and Jensen, H. W. Structured importance sampling of environment maps. *ACM Transactions on Graphics*, 22(3):605–612, 2003.

Ball, W. W. R. *Mathematical recreations and essays.* MacMillan and Co., 1926.

Bartlett, M. S. *An introduction to stochastic processes with special reference to methods and applications.* Cambridge University Press, 1955.

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

Bonet, J. S. D. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings of ACM SIGGRAPH 1997*, pages 361–368. 1997.

Burchill, L. Graphics goodies #2 - a simple, versatile procedural texture. *Computer Graphics*, 22(1):29–30, 1988.

Catmull, E. E. *A Subdivision Algorithm for Computer Display of Curved Surfaces.* Ph.D. thesis, Department of Computer Science, University of Utah, 1974.

Cipra, B. Packing challenge mastered at last. *Science*, 281, 1998.

Cohen, J. and Debevec, P. LightGen, HDRShop plugin. `http://gl.ict.usc.edu/HDRShop/lightgen/lightgen.html`, 2001.

Cohen, M. F., Shade, J., Hiller, S., and Deussen, O. Wang tiles for image and texture generation. *ACM Transactions on Graphics*, pages 287–294, 2003.

Cook, R. L. Stochastic sampling in computer graphics. *Computer Graphics (Proceedings of ACM SIGGRAPH 86)*, 5(1):51–72, 1986.

Cook, R. L. and DeRose, T. Wavelet noise. *ACM Transactions on Graphics*, 24(3):803–811, 2005.

Crow, F. C. The aliasing problem in computer-generated shaded images. *Communications of the ACM*, 20(11):799–805, 1977.

Culik, II, K. An aperiodic set of 13 Wang tiles. *Discrete Mathematics*, 160(1-3):245–251, 1996.

Culik, II, K. and Kari, J. An aperiodic set of Wang cubes. *Journal of Universal Computer Science*, 1(10), 1995.

Deussen, O., Hanrahan, P., Lintermann, B., Měch, R., Pharr, M., and Prusinkiewicz, P. Realistic modeling and rendering of plant ecosystems. In *Proceedings of ACM SIGGRAPH 1998*, pages 275–286. 1998.

Deussen, O., Hiller, S., van Overveld, C., and Strothotte, T. Floating points: A method for computing stipple drawings. *Computer Graphics Forum*, 19(3):40–51, 2000.

Dippé, M. A. Z. and Wold, E. H. Antialiasing through stochastic sampling. *Computer Graphics (Proceedings of ACM SIGGRAPH 85)*, 19(3):69–78, 1985.

Du, Q., Faber, V., and Gunzburger, M. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4):637–676, 1999.

Dunbar, D. and Humphreys, G. A spatial data structure for fast Poisson-disk sample generation. *ACM Transactions on Graphics*, 25(3):503–508, 2006.

Dutré, P., Bala, K., and Bekaert, P. *Advanced Global Illumination*. A. K. Peters, Ltd., Natick, MA, USA, 2002.

Ebert, D. S., Musgrave, F. K., Peachey, D., Perlin, K., and Worley, S. *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann Publishers, Inc., 2002.

Efros, A. A. and Freeman, W. T. Image quilting for texture synthesis and transfer. In *Proceedings of ACM SIGGRAPH 2001*, pages 341–346. 2001.

Efros, A. A. and Leung, T. K. Texture synthesis by non-parametric sampling. In *International Conference on Computer Vision*, pages 1033–1038. 1999.

Escher, M. C. and Locher, J. C. *The World of M. C. Escher*. Abrams, New York, NY, USA, 1971.

Fu, C.-W. and Leung, M.-K. Texture tiling on arbitrary topological surfaces using Wang tiles. In *Rendering Techniques 2005*, pages 99–104. 2005.

Gielis, J. A generic geometric transformation that unifies a wide range of natural and abstract shapes. *American Journal of Botany*, 90(3):333–338, 2003.

Gielis, J., Beirinckx, B., and Bastiaens, E. Superquadrics with rational and irrational symmetry. In *Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 262–265. 2003.

Glassner, A. *Andrew Glassner's notebook: recreational computer graphics.* Morgan Kaufmann Publishers, Inc., San Fransisco, CA, USA, 1999.

Glassner, A. S. *Principles of digital image synthesis.* Morgan Kaufmann Publishers, Inc., San Fransisco, CA, USA, 1995.

Gooch, B. and Gooch, A. *Non-Photorealistic Rendering.* A. K. Peters, Ltd., Natick, MA, USA, 2002.

Grünbaum, B. and Shepard, G. C. *Tilings and patterns.* W. H. Freeman and Company, 1986.

Hausner, A. Simulating decorative mosaics. In *Proceedings of ACM SIGGRAPH 2001*, pages 573–580. 2001.

Heeger, D. J. and Bergen, J. R. Pyramid-based texture analysis/synthesis. In *Proceedings of ACM SIGGRAPH 1995*, pages 229–238. 1995.

Hiller, S., Deussen, O., and Keller, A. Tiled blue noise samples. In *Vision, Modeling, and Visualization 2001*, pages 265–272. 2001.

Jones, T. R. Efficient generation of Poisson-disk sampling patterns. *Journal of Graphics Tools*, 11(2):27–36, 2006.

Kaplan, C. S. and Salesin, D. H. Escherization. In *Proceedings of ACM SIGGRAPH 2000*, pages 499–510. 2000.

Kari, J. A small aperiodic set of Wang tiles. *Discrete Mathematics*, 160(1-3):259–264, 1996.

Klassen, R. V. Filtered jitter. *Computer Graphics Forum*, 19(4):223–230, 2000.

Knuth, D. E. *The art of computer programming*, volume 1. Addison-Wesley, Reading, MA, USA, 1968.

Kollig, T. and Keller, A. Efficient illumination by high dynamic range images. In *Proceedings of the 14th Eurographics workshop on Rendering*, pages 45–50. 2003.

Kopf, J., Cohen-Or, D., Deussen, O., and Lischinski, D. Recursive Wang tiles for real-time blue noise. *ACM Transactions on Graphics*, 25(3):509–518, 2006.

Kwatra, V., Essa, I., Bobick, A., and Kwatra, N. Texture optimization for example-based synthesis. *ACM Transactions on Graphics*, 24(3):795–802, 2005.

Kwatra, V., Schödl, A., Essa, I., Turk, G., and Bobick, A. Graphcut textures: image and video synthesis using graph cuts. *ACM Transactions on Graphics*, 22(3):277–286, 2003.

Lagae, A. and Dutré, P. A procedural object distribution function. *ACM Transactions on Graphics*, 24(4):1442–1461, 2005a.

Lagae, A. and Dutré, P. Template Poisson disk tiles. Report CW 413, Department of Computer Science, K.U.Leuven, Leuven, Belgium, 2005b.

Lagae, A. and Dutré, P. An alternative for Wang tiles: Colored edges versus colored corners. *ACM Transactions on Graphics*, 25(4):1442–1459, 2006a.

Lagae, A. and Dutré, P. A comparison of methods for generating Poisson disk distributions. Report CW 459, Department of Computer Science, K.U.Leuven, Leuven, Belgium, 2006b.

Lagae, A. and Dutré, P. Generating well-distributed point sets with a self-similar hierarchical tile. Report CW 462, Department of Computer Science, K.U.Leuven, Leuven, Belgium, 2006c.

Lagae, A. and Dutré, P. Long period hash functions for procedural texturing. In *Vision, Modeling, and Visualization 2006*, pages 225–228. Akademische Verlagsgesellschaft Aka GmbH, Berlin, 2006d.

Lagae, A. and Dutré, P. Poisson sphere distributions. In *Vision, Modeling, and Visualization 2006*, pages 373–379. Akademische Verlagsgesellschaft Aka GmbH, Berlin, 2006e.

Lagae, A. and Dutré, P. The tile packing problem. Report CW 461, Department of Computer Science, K.U.Leuven, Leuven, Belgium, 2006f.

Lagae, A., Kari, J., and Dutré, P. Aperiodic sets of square tiles with colored corners. Report CW 460, Department of Computer Science, K.U.Leuven, Leuven, Belgium, 2006.

L'Ecuyer, P. Efficient and portable combined random number generators. *Communications of the ACM*, 31(6):742–749, 1988.

Lefebvre, S. and Neyret, F. Pattern based procedural textures. In *Proceedings of the 2003 Symposium on Interactive 3D Graphics*, pages 203–212. 2003.

Lewis, J. P. Algorithms for solid noise synthesis. *Computer Graphics (Proceedings of ACM SIGGRAPH 89)*, 23(3):263–270, 1989.

Liang, L., Liu, C., Xu, Y.-Q., Guo, B., and Shum, H.-Y. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics*, 20(3):127–150, 2001.

Liu, Y., Lin, W.-C., and Hays, J. Near-regular texture analysis and manipulation. *ACM Transactions on Graphics*, 23(3):368–376, 2004.

Lloyd, S. P. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

Lu, A. and Ebert, D. S. Example-based volume illustrations. In *Proceedings of IEEE Visualization*, pages 655–662. 2005.

Lukkarila, V. The square tiling problem is NP-complete for deterministic tile sets. Technical Report TUCS Technical Report No 754, Turku Centre for Computer Science, 2006.

MacMahon, M. P. A. *New mathematical pastimes.* Cambridge University Press, 1921.

Matsumoto, M. and Nishimura, T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.

McCool, M. and Fiume, E. Hierarchical Poisson disk sampling distributions. In *Proceedings of Graphics Interface '92*, pages 94–105. 1992.

Mitchell, D. P. Generating antialiased images at low sampling densities. *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, 21(4):65–72, 1987.

Mitchell, D. P. Spectrally optimal sampling for distribution ray tracing. *Computer Graphics (Proceedings of ACM SIGGRAPH 91)*, 25(4):157–164, 1991.

Mitchell, D. P. and Netravali, A. N. Reconstruction filters in computer graphics. *Computer Graphics (Proceedings of ACM SIGGRAPH 88)*, 22(4):221–228, 1988.

Mitsa, T. and Parker, K. J. Digital halftoning using a blue-noise mask. *Image Processing Algorithms and Techniques II*, 1452(1):47–56, 1991.

Neyret, F. and Cani, M.-P. Pattern-based texturing revisited. In *Proceedings of ACM SIGGRAPH 1999*, pages 235–242. 1999.

Ng, T.-Y., Wen, C., Tan, T.-S., Zhang, X., and Kim, Y. J. Generating an ω-tile set for texture synthesis. In *Proceedings of Computer Graphics International 2005*, pages 177–184. 2005.

Niederreiter, H. *Random number generation and quasi-Monte Carlo methods.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

Ostromoukhov, V., Donohue, C., and Jodoin, P.-M. Fast hierarchical importance sampling with blue noise properties. *ACM Transactions on Graphics*, 23(3):488–495, 2004.

Parish, Y. I. H. and Müller, P. Procedural modeling of cities. In *Proceedings of ACM SIGGRAPH 2001*, pages 301–308. 2001.

Peachy, D. R. Solid texturing of complex surfaces. *Computer Graphics (Proceedings of ACM SIGGRAPH 85)*, 19(3):279–286, 1985.

Penrose, R. The rôle of aesthetics in pure and applied mathematical research. *Bulletin of the Institute of Mathematics and its Applications*, 10:266–271, 1974.

Perlin, K. An image synthesizer. *Computer Graphics (Proceedings of ACM SIGGRAPH 85)*, 19(3):287–296, 1985.

Perlin, K. Improving noise. *ACM Transactions on Graphics*, pages 681–682, 2002.

Perlin, K. and Hoffert, E. M. Hypertexture. *Computer Graphics (Proceedings of ACM SIGGRAPH 89)*, 23(3):253–262, 1989.

Pharr, M. and Humphreys, G. *Physically Based Rendering.* Morgan Kaufmann Publishers, Inc., San Fransisco, CA, USA, 2004.

Robinson, R. M. Seven polygons which admit only nonperiodic tilings of the plane (abstract). *Notices of the American Mathematical Society*, 14:835, 1967.

Saladin, H. *L'Alhambra de Grenade.* Morance, Paris, France, 1926.

Secord, A., Heidrich, W., and Streit, L. Fast primitive distribution for illustration. In *Proceedings of the 13th Eurographics workshop on Rendering*, pages 215–226. 2002.

Shade, J., Cohen, M. F., and Mitchell, D. P. Tiling layered depth images. Technical report, University of Washington, Department of Computer Science and Engineering, 2000.

Sloan, I. H. and Joe, S. *Lattice methods for multiple integration.* Oxford University Press, New York, NY, USA, 1994.

Stam, J. Aperiodic texture mapping. Technical Report ERCIM-01/97-R046, European Research Consortium for Informatics and Mathematics (ECRIM), 1997.

Steinhaus, H. *Mathematical Snapshots.* Dover Publications, Inc., Mineaola, NY, USA, 1999.

Tufte, E. R. *The Visual Display of Quantitative Information.* Graphics Press, Cheshire, CT, USA, 1986.

Turk, G. Generating textures on arbitrary surfaces using reaction-diffusion. *Computer Graphics (Proceedings of ACM SIGGRAPH 91)*, 25(4):289–298, 1991.

Ulichney, R. *Digital Halftoning.* The MIT Press, Cambridge, MA, USA, 1987.

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

Wang, H. Games, logic and computers. *Scientific American*, 213(5):98–106, 1965.

Wang, H. Notes on a class of tiling problems. *Fundamenta Mathematicae*, 82:295–305, 1975.

Wei, L.-Y. Tile-based texture mapping on graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 55–63. 2004.

Wei, L.-Y. and Levoy, M. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of ACM SIGGRAPH 2000*, pages 479–488. 2000.

Wichmann, B. A. and Hill, I. D. An efficient and portable pseudo-random number generator. *Applied Statistics*, 31:188–190, 1982.

Worley, S. A cellular texture basis function. In *Proceedings of ACM SIGGRAPH 1996*, pages 291–294. 1996.

Yellot, Jr., J. I. Spectral analysis of spatial sampling by photoreceptors: Topological disorder prevents aliasing. *Vision Research*, 22:1205–1210, 1982.

Yellot, Jr., J. I. Spectral consequences of photoreceptor sampling in the rhesus retina. *Science*, 221:382–385, 1983.

List of Publications

This is a list of all scientific publications of Ares Lagae (2002 – 2006). This list also includes publications not discussed in this dissertation.

Articles in International Reviewed Journals

- Frank Suykens, Karl vom Berge, **Ares Lagae**, and Philip Dutré. Interactive rendering with bidirectional texture functions. *Computer Graphics Forum*, 22(3):463–472, September 2003.

- **Ares Lagae** and Philip Dutré. An efficient ray-quadrilateral intersection test. *Journal of Graphics Tools*, 10(4):23–32, 2005.

- **Ares Lagae** and Philip Dutré. A procedural object distribution function. *ACM Transactions on Graphics*, 24(4):1442–1461, 2005.

- **Ares Lagae** and Philip Dutré. An alternative for Wang tiles: Colored edges versus colored corners. *ACM Transactions on Graphics*, 25(4):1442–1459, 2006.

Contributions at International Conferences, Published in Proceedings

- **Ares Lagae**, Oliver Dumont, and Philip Dutré. Geometry synthesis by example. In *International Conference on Shape Modeling and Applications*, pages 176–185. IEEE Computer Society, 2005.

- **Ares Lagae** and Philip Dutré. Poisson sphere distributions. In *Vision, Modeling, and Visualization 2006*, pages 373–379, Berlin, 2006. Akademische Verlagsgesellschaft Aka GmbH.

- **Ares Lagae** and Philip Dutré. Long period hash functions for procedural texturing. In *Vision, Modeling, and Visualization 2006*, pages 225–228, Berlin, 2006. Akademische Verlagsgesellschaft Aka GmbH.

Contributions at International Conferences, not Published or Only as Abstract

- **Ares Lagae**, Oliver Dumont, and Philip Dutré. Geometry synthesis. SIGGRAPH 2004 Sketch, SIGGRAPH 2004, Los Angeles, USA, August 2004.

Technical Reports

- **Ares Lagae**, Frank Suykens, and Philip Dutré. A hardware accelerated alternative for the accumulation buffer. Report CW 369, Department of Computer Science, K.U.Leuven, Leuven, Belgium, Januari 2003.
- **Ares Lagae**, Oliver Dumont, and Philip Dutré. Geometry synthesis. Report CW 381, Department of Computer Science, K.U.Leuven, Leuven, Belgium, March 2004.
- **Ares Lagae** and Philip Dutré. An efficient ray-quadrilateral intersection test. Report CW 394, Department of Computer Science, K.U.Leuven, Leuven, Belgium, October 2004.
- **Ares Lagae** and Philip Dutré. Template Poisson disk tiles. Report CW 413, Department of Computer Science, K.U.Leuven, Leuven, Belgium, May 2005.
- **Ares Lagae** and Philip Dutré. A comparison of methods for generating Poisson disk distributions. Report CW 459, Department of Computer Science, K.U.Leuven, Leuven, Belgium, August 2006.
- **Ares Lagae**, Jarkko Kari, and Philip Dutré. Aperiodic sets of square tiles with colored corners. Report CW 460, Department of Computer Science, K.U.Leuven, Leuven, Belgium, August 2006.
- **Ares Lagae** and Philip Dutré. The tile packing problem. Report CW 461, Department of Computer Science, K.U.Leuven, Leuven, Belgium, August 2006.
- **Ares Lagae** and Philip Dutré. Generating well-distributed point sets with a self-similar hierarchical tile. Report CW 462, Department of Computer Science, K.U.Leuven, Leuven, Belgium, August 2006.

Miscellaneous

- **Ares Lagae**. Interactive realistic rendering using a hardware programmable graphics pipeline. Master's thesis, Katholieke Universiteit Leuven, Departement Computerwetenschappen, Celestijnenlaan 200A, B-3001 Heverlee, Belgium, May 2002.

- **Ares Lagae**, Vincent Masselus, and Philip Dutré. Eurographics symposium on rendering 2003 event report. Computer Graphics Forum 23(1):123, Januari 2004.

Biography

Ares Lagae was born on October 2, 1980 in Roeselare, Belgium. He received a Bachelor's degree in Informatics (Kandidaat in de Informatica) and a Master's degree in Informatics (Licenciaat in de Informatica) from the Katholieke Universiteit Leuven in Belgium. He graduated Summa cum Laude in July 2002. His master's thesis, supervised by prof. dr. ir. Philip Dutré, was awarded with the Jos Schepens Memorial Fund prize. He started working as a doctoral student in the Computer Graphics Research Group of the Department of Computer Science at the Katholieke Universiteit Leuven in October 2002, under the supervision of prof. dr. ir. Philip Dutré, funded as Aspirant van het Fonds Wetenschappelijk Onderzoek — Vlaanderen. He completed his Ph.D. thesis in April 2007.

KATHOLIEKE UNIVERSITEIT LEUVEN
FACULTEIT INGENIEURSWETENSCHAPPEN
DEPARTEMENT COMPUTERWETENSCHAPPEN
AFDELING INFORMATICA
Celestijnenlaan 200 A — B-3001 Leuven

TEGELGEBASEERDE METHODES IN COMPUTER GRAPHICS

Examencommissie:
Prof. dr. ir. Jean Berlamont, voorzitter
Prof. dr. ir. Philip Dutré, promotor
Prof. dr. ir. Ronald Cools
Prof. dr. ir. Marc Van Barel
Prof. dr. Philippe Bekaert (Universiteit Hasselt)
Prof. dr. Marc Stamminger
(Friedrich-Alexander-Universität Erlangen-Nürnberg)

Proefschrift voorgedragen tot
het behalen van het doctoraat
in de ingenieurswetenschappen

door

Ares LAGAE

UDC 681.3∗I3

April 2007

Samenvatting

Veel complexe signalen, zoals puntenverdelingen en texturen, kunnen niet efficiënt gesynthetiseerd en opgeslaan worden. In deze thesis stellen we tegelgebaseerde methodes voor om dit probleem op te lossen. In plaats van een complex signaal te synthetiseren wanneer nodig, wordt het signaal vooraf gesynthetiseerd over een kleine verzameling tegels. Willekeurig grote hoeveelheden van dit signaal kunnen dan efficiënt gesynthetiseerd worden wanneer nodig, door het genereren van een stochastische tegeling.

Tegelgebaseerde methodes zijn traditioneel gebaseerd op Wangtegels. De gekleurde randen van Wangtegels beperken enkel de vier direct aanliggende tegels, maar niet de vier schuin aanliggende tegels. Dit probleem veroorzaakt ongewenste artefacten in de getegelde signalen en maakt methodes om signalen te synthetiseren over een verzameling Wangtegels ingewikkelder. Om dit probleem op te lossen stellen we hoektegels voor. Hoektegels zijn vierkante tegels met gekleurde hoeken in plaats van randen. De gekleurde hoeken van Wangtegels beperken alle aanliggende tegels. We herzien de belangrijkste toepassingen van Wangtegels en we tonen aan dat hoektegels belangrijke voordelen hebben voor elk van deze toepassingen.

Stochastische tegelingen worden traditioneel gegenereerd met lijngebaseerde stochastische tegelalgoritmes. Deze algoritmes slaan echter de hele tegeling op en zijn daardoor niet efficiënt. Om dit probleem om te lossen introduceren we directe stochastische tegelalgoritmes voor Wangtegels en hoektegels. Deze algoritmes zijn in staat om een stochastische tegeling lokaal te evalueren, zonder expliciet de ganse tegeling tot op dat punt te construeren en op te slaan. We introduceren ook hashfuncties met een lange periode om zeer grote tegelingen te genereren.

Poissonschijfverdelingen en texturen zijn twee voorbeelden van complexe signalen. We stellen tegelgebaseerde methodes voor om Poissonschijfverdeling te genereren en om texturen te synthetiseren. Tegelgebaseerde methodes laten niet enkel toe om efficiënt Poissonschijfverdelingen te genereren en texturen te synthetiseren, maar maken ook nieuwe toepassingen mogelijk, zoals een tegelgebaseerd textuurafbeeldingsalgoritme en een procedurale objectverdelingsfunctie. Deze nieuwe textuurbasisfunctie laat toe om procedurale objecten te verdelen over een procedurale achtergrond, gebruik makende van intuïtieve parameters zoals de schaal, grootte en oriëntatie van de objecten. We geven tevens een overzicht van toepassingen van getegelde Poissonschijfverdelingen en een gedetailleerde vergelijking van verschillende methodes om Poissonschijfverdelingen te genereren. We bestuderen ook hoektegels in de context van het tegelprobleem en aperiodische tegelverzamelingen en we construeren verschillende nieuwe aperiodische verzamelingen Wangtegels en hoektegels.

De tegelgebaseerde methodes die we in deze thesis voorstellen zijn een waardevol middel voor computer graphics en helpen om bij te blijven met de steeds toenemende vraag naar meer complexiteit en realisme in computergegenereerde beelden.

1 Inleiding

Het genereren en opslaan van complexe signalen, zoals puntenverdelingen en texturen, is een algemeen probleem in computer graphics. Voor veel van deze signalen zijn geen efficiënte synthesealgoritmes beschikbaar en het opslaan van grote hoeveelheden van deze signalen is duur. In deze thesis stellen we tegelgebaseerde algoritmes voor om dit probleem op te lossen.

In plaats van een complex signaal te genereren wanneer nodig, wordt het vooraf gesynthetiseerd over een kleine verzameling Wangtegels of hoektegels. Willekeurig grote hoeveelheden van dit signaal kunnen dan efficiënt gegenereerd worden wanneer nodig, door het genereren van een stochastische tegeling.

In deze thesis stellen we tegelgebaseerde methodes voor om Poissonschijfverdelingen te genereren en om texturen te synthetiseren. Deze methodes laten toe om bestaande toepassingen te verbeteren, maar maken ook nieuwe toepassingen mogelijk, zoals een tegelgebaseerd textuurafbeeldingsalgoritme en een procedurale objectverdelingsfuntie.

Overzicht

In sectie 2 introduceren we tegelingen, Wangtegels en hoektegels. Sectie 3 introduceert efficiënte algoritmes voor het genereren van stochastische tegelingen met Wangtegels en hoektegels. In sectie 4 stellen we tegelgebaseerde methodes voor om Poissonschijfverdelingen te genereren. Sectie 5 bespreekt tegelgebaseerde methodes voor textuursynthese en textuurafbeelding. In sectie 6 vergelijken we verschillende methodes om Poissonschijfverdelingen te genereren in detail. Sectie 7 geeft een overzicht van verschillende toepassingen van Poissonschijfverdelingen. In sectie 8 introduceren we methodes om kleine aperiodische verzamelingen hoektegels te construeren. We besluiten in sectie 9.

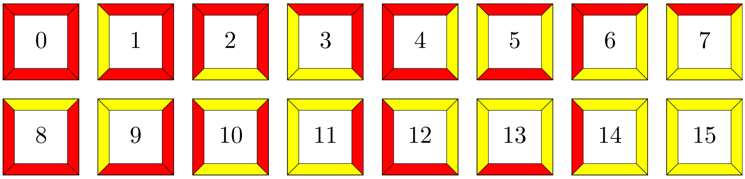
2 Wangtegels en Hoektegels

De tegelgebaseerde methodes die we in deze thesis voorstellen zijn gebaseerd op Wangtegels en hoektegels. In deze sectie introduceren we tegelingen, Wangtegels en hoektegels.

2.1 Tegelingen

Tegelingen vinden we overal terug. Denken maar aan het Alhambra in Granada en het werk van de Nederlandse kunstenaar M. C. Escher.

Een tegeling (Engels: *tiling*) is een configuratie van vlakke figuren die het vlak bedekt zonder gaten en overlappingen. Elke vlakke figuur is een tegel (Engels: *tile*). De verzameling van vlakke figuren gebruikt in de tegeling is de tegelverzameling (Engels: *tile set*).



Figuur 1: De volledige verzameling Wangtegels over twee kleuren.

Een tegeling is periodisch (Engels: *periodic*) als er een verschuiving bestaat die de tegeling bewaart. Als dit niet het geval is, dan is de tegeling niet-periodisch (Engels: *non-periodic*). Een aperiodische (Engels: *aperiodic*) tegelverzameling is een tegelverzameling waarmee geen periodische tegelingen kunnen geconstrueerd worden. Een tegeling gegenereerd met een aperiodische tegelverzameling is een aperiodische tegeling.

Het klassieke werk over tegelingen is *Tilings and Patterns* [Grünbaum and Shepard, 1986].

2.2 Wangtegels

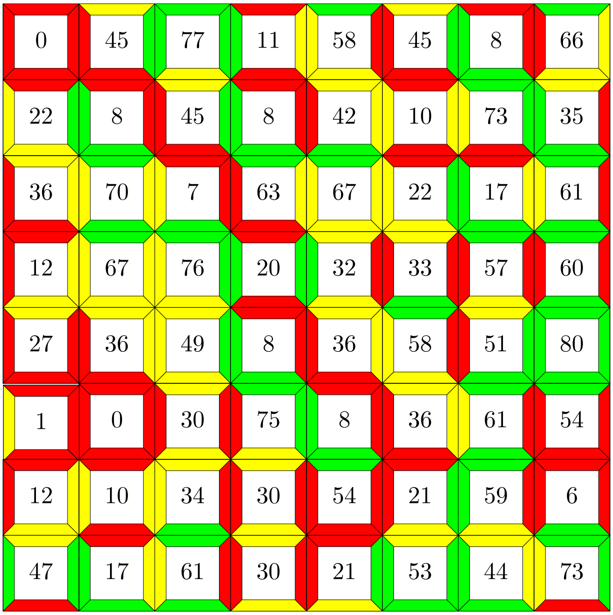
Wangtegels (Engels: *Wang tiles*) zijn vierkante tegels. De tegels hebben dezelfde grootte en elke rand heeft een bepaalde kleur. Deze kleuren worden gecombineerd op verschillende manieren. Om het vlak te tegelen met een verzameling Wangtegels mag elke tegel een willekeurig aantal keer gebruikt worden. Aaneensluitende randen moeten echter steeds dezelfde kleur hebben. Figuur 1 toont een verzameling Wangtegels en figuur 2 toont een tegeling gegenereerd met Wangtegels.

Wangtegels werden voorgesteld door Wang in 1961 en werden populair door een artikel in *Scientific American* [Wang, 1965]. In deze artikels bestudeerde Wang het tegelprobleem met Wangtegels. Wang stelde een algoritme voor om te bepalen of een gegeven verzameling Wangtegels het vlak kon tegelen.

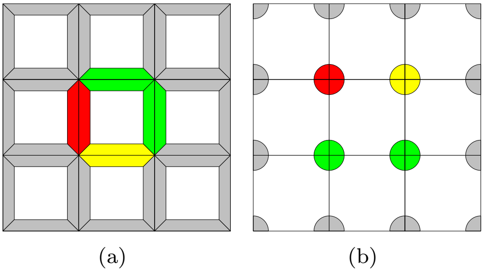
Wangtegels werden geïntroduceerd in computer graphics door Stam [1997] en werden populair door een artikel van Cohen et al. [2003]. In computer graphics worden Wangtegels gebruikt voor de synthese en opslag van complexe signalen. In plaats van een complex signaal te genereren wanneer nodig, wordt het vooraf gesynthetiseerd over een kleine verzameling Wangtegels. Willekeurig grote hoeveelheden van dit signaal kunnen dan efficiënt gegenereerd worden wanneer nodig, door het genereren van een stochastische tegeling.

2.3 Hoektegels

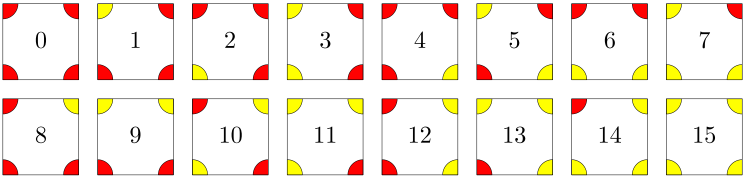
Een nadeel van Wangtegels is dat ze hun diagonale buren niet beperken. Hierdoor kunnen Wangtegels geen continuïteit over hoeken verzekeren in het gegenereerde signaal. Dit is geïllustreerd in figuur 3(a). Om het hoekprobleem



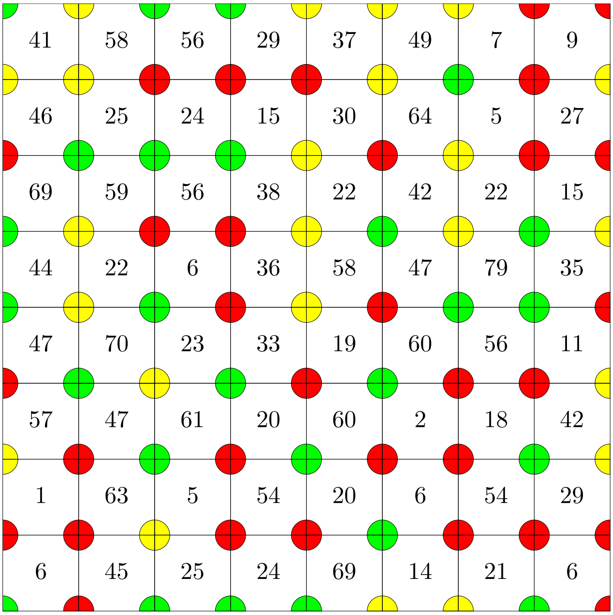
Figuur 2: Een tegeling met de volledige verzameling Wangtegels over drie kleuren.



Figuur 3: Het hoekprobleem. (a) Wangtegels beperken enkel hun direct aanliggende buren en niet hun schuin aanliggende buren. (b) Hoektegels beperken zowel hun direct aanliggende als hun schuin aanliggende buren.



Figuur 4: De volledige verzameling hoektegels over twee kleuren.



41	58	56	29	37	49	7	9
46	25	24	15	30	64	5	27
69	59	56	38	22	42	22	15
44	22	6	36	58	47	79	35
47	70	23	33	19	60	56	11
57	47	61	20	60	2	18	42
1	63	5	54	20	6	54	29
6	45	25	24	69	14	21	6

Figuur 5: Een tegeling met de volledige verzameling hoektegels over drie kleuren.

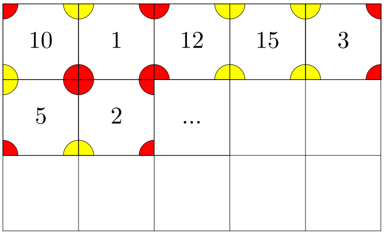
(Engels: *corner problem*) op te lossen stellen we hoektegels (Engels: *corner tiles*) voor. Hoektegels zijn gelijkaardig aan Wangtegels, maar hun gekleurde hoeken verzekeren continuïteit over zowel randen als hoeken in het gegenereerde signaal. Dit is geïllustreerd in figuur 3(b). Figuur 4 toont een verzameling hoektegels en figuur 5 toont een tegeling gegenereerd met hoektegels.

3 Tegelalgoritmes voor Wangtegels en Hoektegels

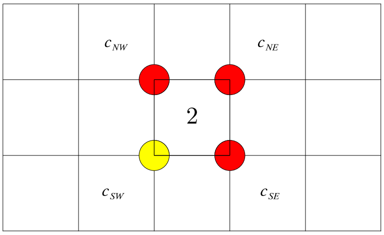
Na het synthetiseren van een complex signaal over een verzameling Wangtegels of hoektegels kan een willekeurig grote hoeveelheid van het signaal efficiënt gegenereerd worden door middel van een tegeling. Omdat herhaling in het signaal visueel storend is, gebruiken toepassingen in computer graphics willekeurige of stochastische tegelingen. In deze sectie bespreken we twee soorten stochastische tegelalgoritmes. We bespreken tevens hashfuncties, een essentieel ingrediënt van stochastische tegelalgoritmes.

3.1 Lijngebaseerde Stochastische Tegelalgoritmes

Stochastische tegelingen worden traditioneel gegenereerd met lijngebaseerde stochastisch tegelalgoritmes (Engels: *scanline stochastic tiling algorithms*). We



Figuur 6: Een lijngebaseerd stochastisch tegelalgoritme voor hoektegels.



Figuur 7: Een direct stochastisch tegelalgoritme voor hoektegels.

introduceren een lijngebaseerd stochastisch tegelalgoritme voor hoektegels, gebaseerd op een gelijkaardig algoritme voor Wangtegels [Cohen et al., 2003].

De hoektegels worden tegel per tegel geplaatst, van links naar rechts en van boven naar onder. Een willekeurige tegel wordt gekozen voor de linkerbovenhoek. De rij wordt aangevuld met tegels waarvan de hoekkleuren overeenkomen met de reeds geplaatste tegels. Dit is geïllustreerd in figuur 6. Om een niet-periodische tegeling te garanderen, is de verzameling hoektegels zo opgesteld dat er voor elke mogelijke combinatie van reeds aanwezige hoekkleuren ten minste twee tegels zijn. Zo kan er bij het toevoegen van een tegel steeds een willekeurige keuze gemaakt worden.

3.2 Directe Stochastische Tegelalgoritmes

Een nadeel van lijngebaseerde stochastische tegelalgoritmes is dat ze niet toelaten om efficiënt een tegeling lokaal te evalueren. Om dit probleem op te lossen introduceren we directe stochastische tegelalgoritmes (Engels: *direct stochastic tiling algorithms*) voor Wangtegels en hoektegels.

Directe stochastische tegelalgoritmes zijn gebaseerd op hashfuncties gedefinieerd over het integer rooster. Deze functies associëren een willekeurige kleur met elk roosterpunt. Dit gekleurd rooster wordt getransformeerd naar een tegeling. Dit is geïllustreerd in figuur 7. De hashfuncties die we gebruiken kunnen lokaal geëvalueerd worden en zijn efficiënt in ruime en tijd. Het directe stochastisch tegelalgoritme voor hoektegels is bijzonder eenvoudig, omdat het gekleurde rooster equivalent is met de tegeling. Directe stochastische tegelalgoritmes voor Wangtegels zijn iets ingewikkelder.

3.3 Hashfuncties

Hashfuncties (Engels: *hash functions*) gedefinieerd over het integer rooster associëren een willekeurig getal met elk roosterpunt. Hashfuncties zijn een essentieel ingrediënt van stochastische tegelalgoritmes, maar hebben ook verschillende andere toepassingen, zoals procedurale modellering.

Een ééndimensionale hashfunctie h is traditioneel gedefinieerd als [Perlin, 2002]

$$h(x) = P[x\%N], \tag{1}$$

waarbij x een geheel getal is, % de modulodeling voorstelt en P een permutatietabel met grootte N is. Een permutatietabel met grootte N bevat een willekeurige permutatie van de getallen $\{0, 1, \ldots, N-1\}$. Deze hashfunctie is een periodische functie met periode N.

Een nadeel van traditionele hashfuncties is dat de periode van de functie relatief kort is in verhouding tot de grootte van de permutatietabel. Daarom stellen we hashfuncties voor met een lange periode (Engels: *long-period hash functions*). Een ééndimensionale hashfunctie h met een lange periode is gedefinieerd als

$$h(x) = \left(\sum_{i=0}^{M-1} P_i[x\%N_i] \right) \%N_j, \tag{2}$$

waarbij $P_0, P_1, \ldots, P_{M-1}$ M permutatietabellen met groottes $N_0, N_1, \ldots, N_{M-1}$ zijn en N_j één van deze groottes is. Deze hashfunctie is een periodische functie met als periode het kleinste gemeen veelvoud van $N_0, N_1, \ldots, N_{M-1}$, terwijl de gecombineerde grootte van de permutatietabellen slechts $\sum_{i=0}^{M-1} N_i$ is.

Een hashfunctie met lange periode gebaseerd op 7 permutatietabellen met als grootte 17, 19, 23, 24, 29, 31, 27 heeft een periode van 5.930.659.848, terwijl een traditionele hashfunctie gebaseerd op één permutatietabel met als grootte 180, de gecombineerde grootte van deze permutatietabellen, een periode heeft van slechts 180.

4 Tegelgebaseerde Methodes voor het Genereren van Poissonschijfverdelingen

In computer graphics worden Wangtegels en hoektegels gebruikt voor de synthese van complexe signalen. Poissonschijfverdelingen zijn puntenverdelingen die moeilijk efficiënt te genereren zijn. In deze sectie stellen we methodes voor om Poissonschijfverdelingen te construeren over een verzameling Wangtegels of hoektegels.

4.1 Poissonschijfverdelingen

Een Poissonverdeling (Engels: *Poisson distribution*) is een willekeurige puntenverdeling waarin de punten en de coördinaten van de punten geen verband met elkaar hebben. Een Poissonschijfverdeling (Engels: *Poisson disk distribution*) is een tweedimensionale Poissonverdeling waarin de punten gescheiden zijn door een minimale afstand. De helft van deze afstand wordt de straal r van de puntenverdeling genoemd. Als een schijf met straal r op elk punt geplaatst wordt, dan overlappen de schijven niet.

Poissonschijfverdelingen werden geïntroduceerd in computer graphics in de context van niet-uniforme bemonstering. Voor computer graphics is de Poissonschijfverdelingen één van de beste stochastische bemonsteringspatronen [Dippé and Wold, 1985; Cook, 1986; Mitchell, 1987].

Traditioneel worden Poissonschijfverdelingen gegenereerd met dartgooien (Engels: *dart throwing*) [Cook, 1986], relaxatiedartgooien (Engels: *relaxation dart throwing*) [McCool and Fiume, 1992] en Lloyd's relaxatiemethode (Engels: *Lloyd's relaxation method*) [McCool and Fiume, 1992].

De straal r van een Poissonschijfverdelingen wordt typisch uitgedrukt als een absoluut getal. Omdat dit niet praktisch is stellen we het relatieve straalspecificatieschema (Engels: *relative radius specification scheme*) voor. De straal wordt uitgedrukt als een fractie $\rho \in [0,1]$ van de maximale straal r_{max}

$$r = \rho\, r_{max}. \tag{3}$$

Voor een puntenverdeling van N punten wordt de maximale straal r_{max} gegeven door

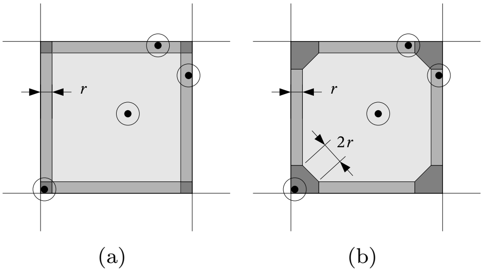
$$r_{max} = \sqrt{\frac{1}{2\sqrt{3}N}}. \tag{4}$$

De relatieve straal ρ is een maat voor de kwaliteit van de Poissonschijfverdelingen. Een goede Poissonschijfverdeling moet een relatieve straal hebben die hoog is ($\rho \geq 0.65$), maar ook niet te hoog ($\rho \leq 0.85$).

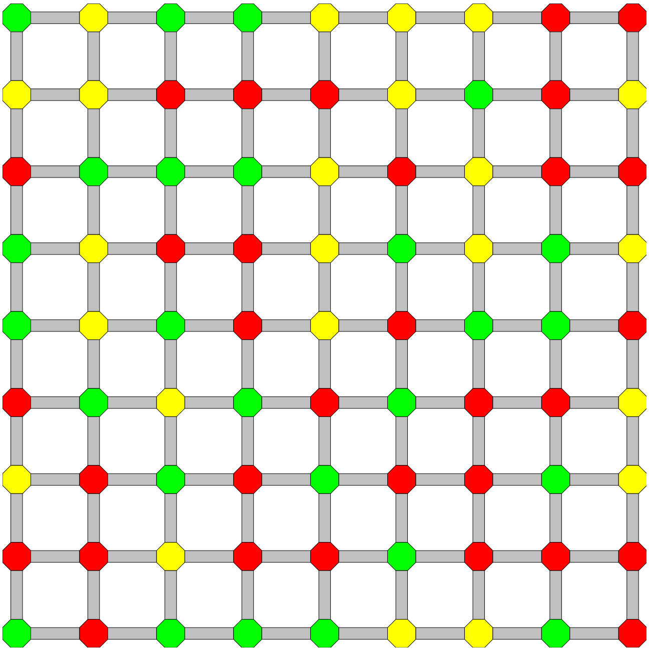
4.2 Poissonschijftegels Gebaseerd op Hoektegels

Poissonschijftegels gebaseerd op hoektegels (Engels: *corner-based Poisson disk tiles*) is een methode om een Poissonschijfverdeling te genereren in elke tegel van een verzameling hoektegels, zodat elke mogelijke tegeling resulteert in een geldige Poissonschijfverdeling.

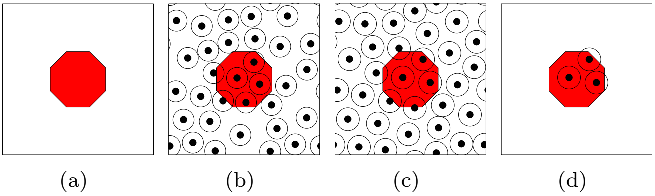
Een punt in een tegel kan naargelang zijn positie punten in drie, één of geen enkele aanliggende tegel beïnvloeden. Op deze manier wordt een tegel onderverdeeld in hoekregio's (Engels: *corner regions*), randregio's (Engels: *edge regions*) en een inwendige regio (Engels: *interior region*). Samen worden deze regio's de Poissonschijftegelregio's (Engels: *Poisson disk regions*) genoemd. Dit



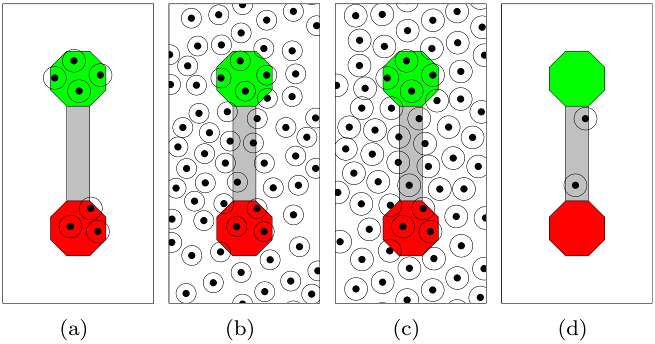
Figuur 8: De Poissonschijftegelregio's en de aangepaste Poissonschijftegelregio's. (a) De Poissonschijfstraal r bepaalt hoekregio's, randregio's en een inwendige regio. (b) De hoekregio's worden aangepast zodat de afstand tussen regio's van hetzelfde type tenminste $2r$ bedraagt.



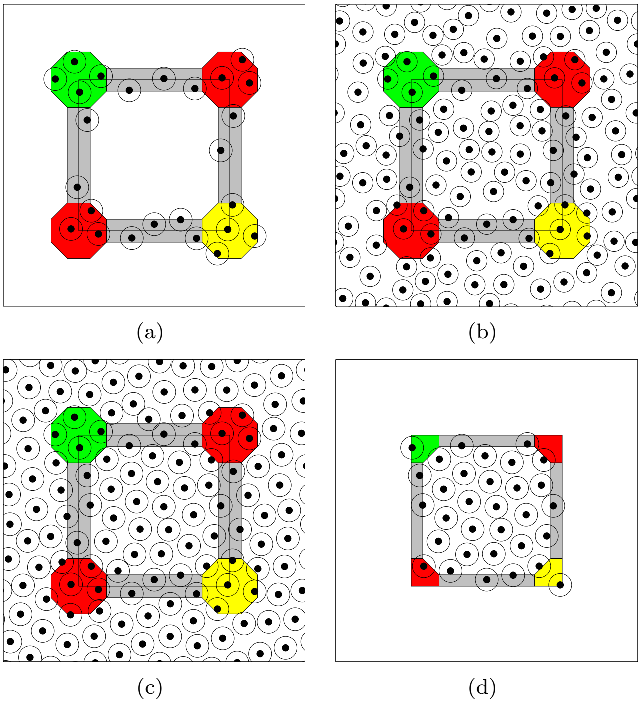
Figuur 9: De tegeling bekomen door de aangepaste Poissonschijftegelregio's te combineren met de volledige verzameling hoektegels over drie kleuren. Deze tegeling werd gegenereerd van de tegeling getoond in figuur 5.



Figuur 10: De generatie van een Poissonschijfverdeling in een hoektegel van een verzameling Poissonschijftegels gebaseerd op hoektegels. (a) De hoektegel. (b) Een Poissonschijfverdeling wordt gegenereerd. (c) De Poissonschijfverdeling wordt geoptimaliseerd met Lloyd's relaxatiemethode. (d) De hoektegel wordt uit de Poissonschijfverdeling gesneden.



Figuur 11: De generatie van een Poissonschijfverdeling in een verticale randtegel van een verzameling Poissonschijftegels gebaseerd op hoektegels. (a) De randtegel en de overeenkomende hoektegels worden geassembleerd. (b) Een Poissonschijfverdeling wordt gegenereerd. (c) De Poissonschijfverdeling wordt geoptimaliseerd met Lloyd's relaxatiemethode. (d) De randtegel wordt uit de Poissonschijfverdeling gesneden.

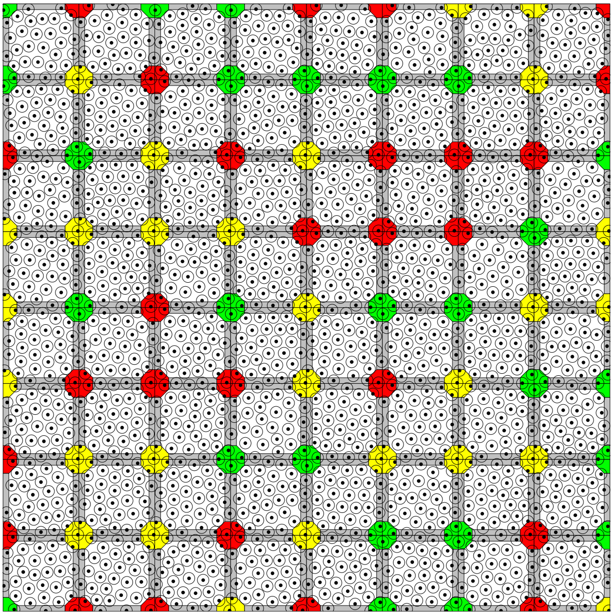

(a)

(b)

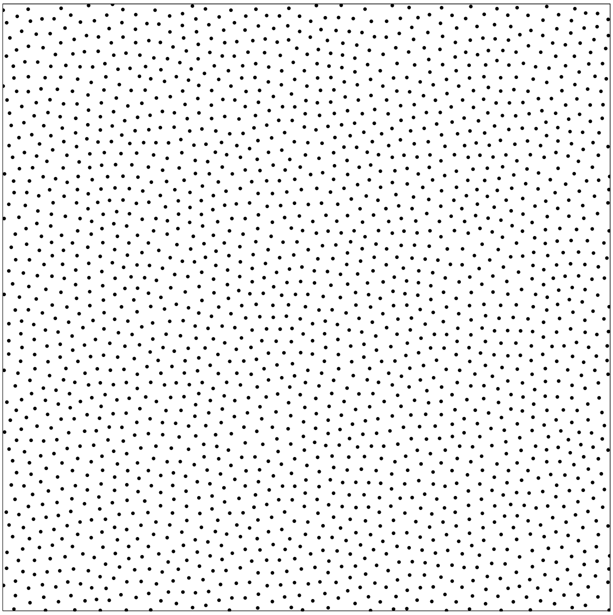
(c)

(d)

Figuur 12: De generatie van een Poissonschijfverdeling in een tegel van een verzameling Poissonschijftegels gebaseerd op hoektegels. (a) De inwendige tegel en de overeenkomende hoektegels en randtegels worden geassembleerd. (b) Een Poissonschijfverdeling wordt gegenereerd. (c) De Poissonschijfverdeling wordt geoptimaliseerd met Lloyd's relaxatiemethode. (d) De tegel wordt uit de Poissonschijfverdeling gesneden.



Figuur 13: Een tegeling met een verzameling Poissonschijftegels gebaseerd op hoektegels.



Figuur 14: Een Poissonschijfverdeling gegenereerd met een verzameling Poissonschijftegels gebaseerd op hoektegels. Deze Poissonschijfverdeling werd gegenereerd van de tegeling getoond in figuur 13.

is geïllustreerd in figuur 8(a). De grootte van deze regio's wordt bepaald door de straal r van de Poissonschijfverdeling.

Om de invloed tussen de verschillende regio's te beperken worden de hoekregio's vergroot zodat de afstand tussen de randregio's $2r$ wordt. Deze nieuwe regio's worden de aangepaste Poissonschijftegelregio's (Engels: *modified Poisson disk regions*) genoemd. Dit is geïllustreerd in figuur 8(b).

Door de aangepaste Poissonschijftegelregio's te combineren met een verzameling hoektegels wordt een nieuwe tegeling verkregen. Dit is geïllustreerd in figuur 9. In deze tegeling worden drie soorten tegels gebruikt. Hoektegels (Engels: *corner tiles*) komen overeen met de unie van vier aangepaste hoekregio's, randtegels (Engels: *edge tiles*) komen overeen met de unie van twee aangepaste randregio's en inwendige tegels (Engels: *interior tiles*) komen overeen met de aangepaste inwendige regio's.

Eerst wordt een Poissonschijfverdeling gegenereerd in de hoektegels. Dan wordt een Poissonschijfverdeling gegenereerd in de randtegels. De punten in de hoektegels worden hierbij niet meer aangepast. Tenslotte wordt een Poissonschijfverdeling gegenereerd in de inwendige tegels. Dan worden de Poissonschijftegels uitgesneden. Dit is geïllustreerd in figuur 10, figuur 11 en figuur 12.

Elke tegeling met de Poissonschijftegels resulteert in een geldige Poissonschijfverdeling. Figuur 13 toont een tegeling met Poissonschijftegels en figuur 14 toont de corresponderende Poissonschijfverdeling.

4.3 Andere Methodes

Naast Poissonschijftegels gebaseerd op hoektegels stellen we verschillende andere tegelgebaseerde methodes voor om Poissonschijfverdelingen te genereren.

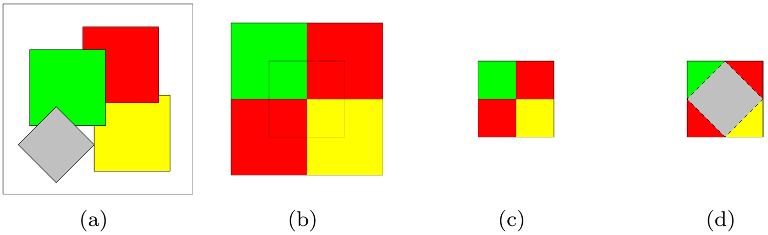
Poissonschijftegels gebaseerd op Wangtegels (Engels: *edge-based Poisson disk tiles*) is een voorloper van Poissonschijftegels gebaseerd op hoektegels. Het gebruik van Wangtegels resulteert door het hoekprobleem echter in een te grote verzameling Poissonschijftegels.

Poissonschijftegels gebaseerd op een sjabloon (Engels: *template Poisson disk tiles*) is een methode opgesteld om de invloed van de grootte van de verzameling Poissonschijftegels op de kwaliteit van de Poissonschijfverdeling te onderzoeken. Deze methode is in de praktijk echter niet bruikbaar.

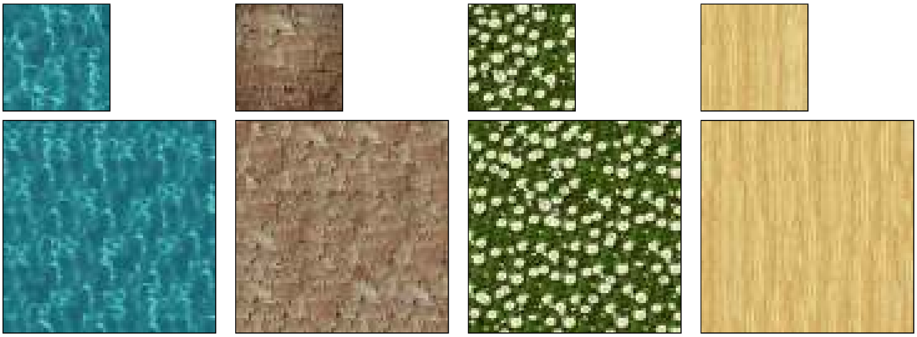
Naast tegelgebaseerde methodes voor het genereren van Poissonschijfverdelingen stellen we ook tegelgebaseerde methodes voor om Poissonbolverdelingen (Engels: *Poisson sphere distributions*) en niet-uniforme Poissonschijfverdelingen (Engels: *nonuniform Poisson disk distributions*) te genereren.

5 Tegelgebaseerde Methodes voor Textuursynthese

In computer graphics worden Wangtegels en hoektegels gebruikt voor de synthese van complexe signalen. Texturen zijn complexe signalen die moeilijk te



Figuur 15: De synthese van een voorbeeldtextuur over een verzameling textuurtegels gebaseerd op hoektegels. (a) Voor elke kleur wordt een vierkante lap gekozen in de voorbeeldtextuur (de rode, groene en gele lap). (b) De lappen worden geassembleerd volgens de hoekkleuren van de tegel. (c) De tegel wordt uitgesneden. (d) De naad wordt bedekt met een nieuwe onregelmatige lap van de voorbeeldtextuur (de grijze lap).



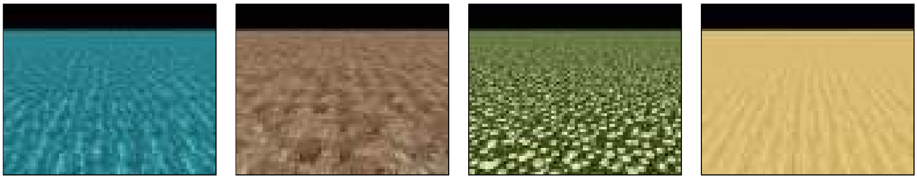
Figuur 16: Texturen gesynthetiseerd met textuurtegels gebaseerd op hoektegels. Deze texturen werden gesynthetiseerd door het genereren van een stochastische 4 × 4 tegeling met een verzameling textuurtegels gebaseerd op hoektegels.

synthetiseren zijn. In deze sectie bespreken we een methode om een textuur te synthetiseren over een verzameling Wangtegels of hoektegels en een textuurafbeeldingsalgoritme om grote niet-herhalende texturen te genereren.

5.1 Tegelgebaseerde Textuursynthese

Textuurafbeelding (Engels: *texture mapping*) is een methode om het realisme van beelden gegenereerd met de computer te verhogen zonder geometrie toe te voegen. In plaats daarvan wordt een textuur afgebeeld op het oppervlak van een object.

Een textuur kan op verschillende manieren verkregen worden. Eén van deze manieren is textuursynthese (Engels: *texture synthesis*). Hierbij wordt een



Figuur 17: Tegelgebaseerde textuurafbeelding gebaseerd op hoektegels. De textuur op het vlak wordt interactief gegenereerd op de grafische kaart gebruik makende van het tegelgebaseerde textuurafbeeldingsalgoritme gebaseerd op hoektegels.

nieuwe textuur gegenereerd van een voorbeeldtextuur. Tegelgebaseerde textuursynthese (Engels: *tile-based texture synthesis*) maakt gebruik van tegels om texturen te synthetiseren.

Om een textuur te synthetiseren over een verzameling hoektegels gebruiken we de methode van Ng et al. [2005]. Voor elke kleur gebruikt in de verzameling hoektegels wordt een vierkante lap (Engels: *patch*) gekozen in de voorbeeldtextuur. Voor elke hoektegel worden de vier overeenstemmende lappen samengelegd. Hieruit wordt de tegel gesneden. Tenslotte wordt de kruisvormige naad bedekt met een nieuwe lap uit de voorbeeldtextuur. Dit is geïllustreerd in figuur 15. Figuur 16 toont enkele texturen gesynthetiseerd met deze methode.

De methode van Ng et al. [2005] voor het genereren van textuurtegels gebaseerd op hoektegels is een uitbreiding van de methode van Cohen et al. [2003] voor het genereren van textuurtegels gebaseerd op Wangtegels. Hoektegels zijn beter geschikt voor textuursynthese dan Wangtegels omdat elke hoektegel een potentieel nieuw deel van de voorbeeldtextuur bevat.

5.2 Tegelgebaseerde Textuurafbeelding

Een voordeel van tegelgebaseerde textuursynthese is dat het proces van textuursynthese opgesplitst wordt in twee stappen. In een eerste stap wordt de voorbeeldtextuur gesynthetiseerd over een verzameling Wangtegels of hoektegels. In een tweede stap wordt een nieuwe textuur gesynthetiseerd door het genereren van een stochastische tegeling. Tegelgebaseerde textuurafbeelding (Engels: *tile-based texture mapping*) voert deze tweede stap efficiënt uit op de grafische kaart (Engels: *GPU*).

Tegelgebaseerde textuurafbeelding neemt als invoer een voorbeeldtextuur gesynthetiseerd over een verzameling Wangtegels of hoektegels en synthetiseert een willekeurig grote niet-herhalende textuur op de grafische kaart. Dit is geïllustreerd in figuur 17. Het doel is om textuurgeheugen, een schaars middel op de grafische kaart, te besparen. Wei [2004] stelde een algoritme voor gebaseerd op Wangtegels. We verbeterden het algoritme van Wei door gebruik te maken van hoektegels. Het algoritme gebaseerd op hoektegels is sneller en reduceert het nodige textuurgeheugen met een factor twee.

De verzameling Wangtegels of hoektegels met de voorbeeldtextuur moet in een bepaalde configuratie worden aangeboden aan de grafische kaart. Deze configuratie wordt een tegelpakking (Engels: *tile packing*) genoemd. Voor hoektegels is het berekenen van een tegelpakking een interessant combinatorisch probleem.

6 Een Vergelijking van Methodes voor het Genereren van Poissonschijfverdelingen

Poissonschijfverdelingen hebben verschillende toepassingen in computer graphics. Over de jaren heen werden verschillende methodes voorgesteld om Poissonschijfverdelingen te genereren. Deze methodes en de Poissonschijfverdelingen gegenereerd met deze methodes zijn dikwijls heel verschillend. Dit maakt het moeilijk om de juiste methode te kiezen voor een bepaalde toepassing. Daarom vergelijken we gedetailleerd de verschillende methodes voor het genereren van Poissonschijfverdelingen.

6.1 Methodologie

Om Poissonschijfverdelingen te vergelijken gebruiken we drie criteria. Een eerste criterium is de relatieve straal van de Poissonschijfverdelingen. Dit is een belangrijke maat voor de kwaliteit van de verdeling. Een tweede criterium is het spectrum (Engels: *power spectrum*) van de Poissonschijfverdelingen. Dit mag geen grote pieken bevatten en er mag geen energie op lage frequenties aanwezig zijn [Yellot, 1982]. Een derde criterium is de bemonsteringsperformantie (Engels: *sampling performance*) van de Poissonschijfverdelingen.

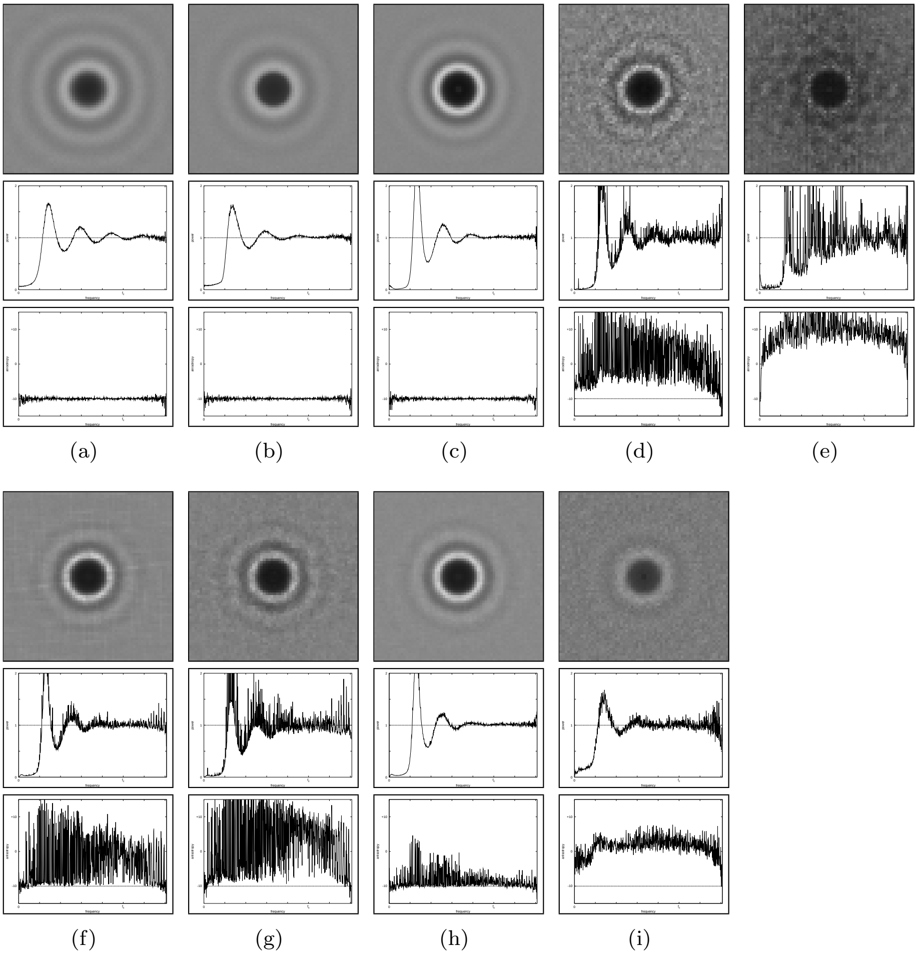
6.2 Vergelijking

Dartgooien (Engels: *dart throwing*) [Cook, 1986] is de meest natuurlijke manier om een Poissonschijfverdeling te genereren. Het spectrum van deze methode wordt daarom gebruikt als referentiespectrum. Dartgooien is echter traag en moeilijk controleerbaar.

Relaxatiedartgooien (Engels: *relaxation dart throwing*) [McCool and Fiume, 1992] is een verbetering van dartgooien. Deze methode genereert daarenboven puntenverdelingen die op verschillende schalen aan de Poissonschijfvoorwaarde voldoen. Relaxatiedartgooien is echter nog steeds traag.

Lloyd's relaxatiemethode (Engels: *Lloyd's relaxation method*) [Lloyd, 1982; McCool and Fiume, 1992] is de enige methode om Poissonschijfverdelingen met een hoge relatieve straal te genereren. Deze methode is echter zeer traag.

De methode van Shade et al. [2000] is een uitbreiding van dartgooien naar Wangtegels. Deze methode is echter niet correct [Cohen et al., 2003].



(a) (b) (c) (d) (e)

(f) (g) (h) (i)

Figuur 18: Spectrale analyse van Poissonschijfverdelingen gegenereerd met (a) dartgooien, (b) relaxatiedartgooien, (c) Lloyd's relaxatiemethode, (d) de methode van Hiller et al., (e) de methode van Ostromoukhov et al., (f) Poissonschijftegels gebaseerd op Wangtegels, (g) Poissonschijftegels gebaseerd op een sjabloon, (h) Poissonschijftegels gebaseerd op hoektegels, (i) de methode van Kopf et al.

De methode van Hiller et al. [2001] is een uitbreiding van Lloyd's relaxatiemethode naar Wangtegels. Deze methode convergeert echter slecht en gebruikt te weinig tegels om goede Poissonschijfverdelingen te genereren.

De methode van Ostromoukhov et al. [2004] genereert niet-uniforme Poissonschijfverdelingen. Deze methode kan ook gebruikt worden om Poissonschijfverdelingen te genereren, maar het spectrum van de verdelingen is relatief slecht.

Poissonschijftegels gebaseerd op Wangtegels is de eerste tegelgebaseerde methode die relatief goede Poissonschijfverdelingen kan genereren. Het aantal Poissonschijftegels hiervoor nodig is echter veel te groot.

Poissonschijftegels gebaseerd op een sjabloon werd ontwikkeld om de invloed van de grootte van de verzameling Poissonschijftegels op de kwaliteit van de Poissonschijfverdeling te onderzoeken. Deze methode genereert echter Poissonschijfverdelingen met een relatief slecht spectrum.

Poissonschijftegels gebaseerd op hoektegels is de eerste tegelgebaseerde methode die in staat is om Poissonschijfverdelingen met een relatief goede kwaliteit te genereren met een aanvaardbaar aantal Poissonschijftegels.

De methode van Jones [2006] en de methode van Dunbar and Humphreys [2006] zijn efficiënte implementaties van dartgooien. Deze methodes produceren Poissonschijfverdelingen met een goede kwaliteit, maar zijn niet geschikt voor interactieve toepassingen en voor toepassingen die Poissonschijfverdelingen met een grote relatieve straal vereisen.

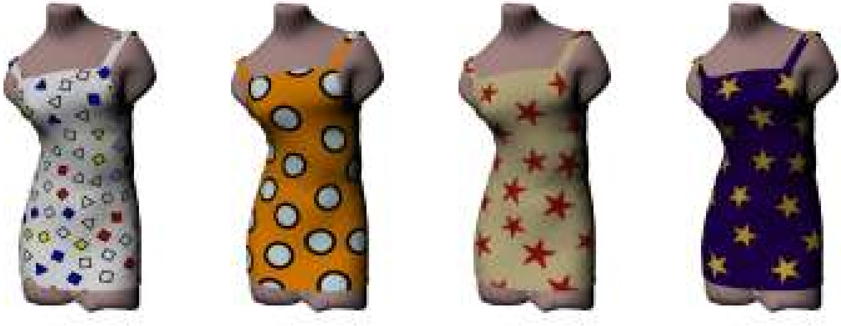
De methode van Kopf et al. [2006] genereert niet-uniforme Poissonschijfverdelingen. Deze methode kan ook gebruikt worden om Poissonschijfverdelingen te genereren, maar de relatieve straal van de Poissonschijfverdelingen is laag en het spectrum van de verdelingen is relatief slecht.

Het spectrum van Poissonschijfverdelingen gegenereerd met de bovenstaande methodes wordt getoond in figuur 18.

Voor interactieve toepassingen, toepassingen die Poissonschijfverdelingen met een grote relatieve straal vereisen en voor grote Poissonschijfverdelingen zijn tegelgebaseerde methodes de enige optie. Van de tegelgebaseerde methodes is Poissonschijftegels gebaseerd op hoektegels de beste methode.

7 Toepassingen van Poissonschijfverdelingen

In de vorige secties hebben we verschillende methodes om Poissonschijfverdelingen te genereren besproken en geanalyseerd. Het genereren van Poissonschijfverdelingen is natuurlijk geen doel op zich. Poissonschijfverdelingen hebben verschillende toepassingen in computer graphics. In deze sectie bespreken we een aantal van deze toepassingen.



Figuur 19: Procedurale texturen gegenereerd met de procedurale objectverdelingsfunctie.

7.1 Procedurale Textuurgeneratie

Procedurale texturen (Engels: *procedural textures*) zijn texturen gedefinieerd door een algoritme. Ten opzichte van traditionele texturen hebben procedurale texturen heel wat voordelen. Procedurale texturen zijn compact, hebben geen vaste resolutie en grootte en zijn gemakkelijk te parametriseren.

Procedurale texturen worden gegenereerd met behulp van textuurbasisfuncties (Engels: *texture basis functions*). De meest gekende textuurbasisfunctie is die van Perlin [Perlin, 1985, 2002]. Naast procedurale texturen worden textuurbasisfuncties ook gebruikt voor procedurale modellering (Engels: *procedural modeling*).

We introduceren een procedurale objectverdelingsfunctie (Engels: *procedural object distribution function*), een nieuwe textuurbasisfunctie. Deze textuurbasisfunctie evalueert efficiënt een getegelde Poissonschijfverdeling. Aan de hand hiervan kunnen procedurale objecten verspreid worden over een procedurale achtergrond, zonder dat de objecten overlappen. De textuurbasisfunctie laat intuïtieve manipulatie toe van de grootte en oriëntatie van de objecten. Dit is geïllustreerd in figuur 19. Dergelijke texturen konden voorheen niet procedurale gegenereerd worden.

7.2 Andere Toepassingen

Poissonschijfverdelingen werden geïntroduceerd in computer graphics in de context van bemonstering. De Poissonschijfverdeling is één van de beste stochastische bemonsteringspatronen [Dippé and Wold, 1985; Cook, 1986; Mitchell, 1987]. De Poissonschijfverdeling wordt echter niet veel gebruikt, vooral omdat er geen efficiënte algoritmes beschikbaar waren om Poissonschijfverdelingen te genereren. De tegelgebaseerde methodes die we voorgesteld hebben bieden hiervoor een oplossing.

Figuur 20: Procedurale modellering van geometrie met Poissonschijfverdelingen. Om dit beukenbos te modelleren werden meer dan 2000 instantiaties van vijf beuken verdeeld met behulp van een Poissonschijfverdeling.

Poissonschijfverdelingen worden dikwijls gebruikt voor procedurale modellering, geometrische objectverdeling (Engels: *geometric object distribution*) en geometrie-instantiëring (Engels: *geometry instancing*) [Deussen et al., 1998]. Dit is geïllustreerd in figuur 20. Veel verdelingen van objecten volgen namelijk een patroon dat gelijkaardig is aan een Poissonschijfverdeling.

Andere toepassingen van Poissonschijfverdelingen zijn niet-fotorealistische visualisering (Engels: *non-photorealistic rendering*) [Secord et al., 2002] en wetenschappelijke visualisering (Engels: *scientific visualization*).

8 Kleine Aperiodische Verzamelingen Hoektegels

Wangtegels werden geïntroduceerd in de context van aperiodische tegelverzamelingen en werden pas later gebruikt in computer graphics. Hoektegels werden geïntroduceerd in de context van computer graphics. In deze sectie bestuderen we hoektegels in de originele context van aperiodische tegelverzamelingen.

8.1 Aperiodische Tegelverzamelingen

In 1961 bestudeerde Wang het tegelprobleem met Wangtegels. Wang stelde een algoritme voor om te bepalen of een gegeven verzameling Wangtegels het vlak kon tegelen. Hierbij veronderstelde Wang dat aperiodische tegelverzamelingen niet bestonden. In 1966 toonde Berger aan dat dit niet zo was. Berger construeerde de eerste aperiodische tegelverzameling, die 20.426 Wangtegels telde.

0 4 1 2	0 3 1 5	0 2 1 5	2 3 3 4	2 2 3 4	2 1 3 0	3 1 4 0	3 0 4 0	5 2 2 3
5 3 2 3	5 5 2 3	5 1 2 1	3 1 3 0	3 0 3 0	5 2 3 3	5 3 3 3	5 5 3 3	5 1 3 1
2 3 4 4	2 2 4 4	2 1 4 0	2 3 2 4	2 2 2 4	2 1 2 0	4 0 2 1	4 4 2 3	3 0 5 1
3 4 5 3	4 0 3 1	4 4 3 3	2 4 5 2	2 3 5 5	2 2 5 5	1 5 0 2	1 2 0 2	1 2 0 3
1 3 0 3	1 5 0 3	1 1 0 1	1 5 1 2	1 2 1 2	0 3 0 4	0 2 0 4	0 1 0 0	

Figuur 21: Een aperiodische verzameling van 44 hoektegels over zes kleuren. Dit is momenteel de kleinste aperiodische verzameling hoektegels.

Dit was één van de meest opvallende ontdekkingen in het onderzoeksgebied van tegelingen.

Over de jaren heen werd dit aantal herhaaldelijk teruggebracht. De kleinste aperiodische verzameling Wangtegels bestaat momenteel uit 13 tegels [Culik, 1996]. Er werden ook verschillende aperiodische tegelingen ontdekt met andere tegels, bijvoorbeeld de Penrose tegeling [Penrose, 1974]. De vraag of er één enkele aperiodische tegel bestaat is nog steeds niet opgelost.

Volgens Grünbaum and Shepard [1986] is elke aperiodische tegelverzameling interessant. Daarom construeerden we kleine aperiodische verzamelingen hoektegels.

8.2 Constructie van Aperiodische Verzamelingen Hoektegels van Aperiodische Verzamelingen Wangtegels

Omdat Wangtegels en hoektegels gelijkaardig zijn, gebruiken we isomorfismes tussen aperiodische tegelingen met Wangtegels en aperiodische tegelingen met hoektegels om aperiodische verzamelingen hoektegels te construeren.

We stellen vijf methodes voor om een aperiodische verzamelingen hoektegels te construeren van een aperiodische verzameling Wangtegels: diagonale, horizontale en verticale verschuiving, rotatie en onderverdeling. Deze methodes worden toegepast op reeds gekende kleine aperiodische verzamelingen Wangtegels. Zo worden nieuwe aperiodische verzamelingen hoektegels bekomen.

De meest eenvoudige methode is diagonale verschuiving. Het rooster van de hoektegels is diagonaal verschoven ten opzichte van het rooster van de Wangtegels. Aan elke tegel in de verzameling Wangtegels wordt een unieke kleur toegekend. Aan iedere hoek van elke hoektegel wordt vervolgens de kleur van de overeenkomende Wang tegel toegekend. De aperiodische verzameling hoektegels die zo bekomen wordt, bestaat uit één tegel voor elke geldige combinatie van vier Wangtegels.

De kleinste aperiodische verzameling hoektegels geconstrueerd met één van bovenstaande methodes bestaat uit 44 tegels over zes kleuren en is getoond in figuur 21.

8.3 Constructie van Aperiodische Verzamelingen Wangtegels van Aperiodische Verzamelingen Hoektegels

Een aperiodische verzameling Wangtegels kan geconstrueerd worden van een aperiodische verzameling hoektegels door elke combinatie van twee hoekkleuren te vervangen door een nieuwe randkleur. Op deze manier bekomen we nieuwe aperiodische verzamelingen Wangtegels. We tonen aan dat, als W en C de groottes zijn van de kleinste aperiodische verzameling Wangtegels en hoektegels, dan $W \leq C \leq 4W$.

9 Besluit

In deze thesis hebben we tegelgebaseerde methodes voorgesteld om efficiënt complexe signalen te genereren en op te slaan. We hebben aangetoond dat tegelgebaseerde methodes in staat zijn om complexe signalen te reproduceren met hoge kwaliteit en dat hoektegels hiervoor beter geschikt zijn dan Wangtegels.

De belangrijkste bijdragen van deze thesis zijn de introductie van hoektegels, directe stochastische tegelalgoritmes voor Wangtegels en hoektegels, hashfuncties met een lange periode, het relatieve straalspecificatieschema voor Poissonschijfverdelingen, tegelgebaseerde methodes voor het genereren van Poissonschijfverdelingen, tegelgebaseerde textuurafbeelding gebaseerd op hoektegels, de vergelijking van de verschillende methodes voor het genereren van Poissonschijfverdelingen, de procedurale objectverdelingsfunctie en de kleine aperiodische verzamelingen hoektegels en Wangtegels. Deze bijdragen werden gerapporteerd in verschillende publicaties (zie bibliografie).

Er zijn verschillende mogelijkheden voor verder onderzoek. Het efficiënt genereren van niet-uniforme Poissonschijfverdelingen is nog steeds een uitdagend probleem. De technieken voorgesteld in deze thesis kunnen natuurlijk ook op andere complexe signalen toegepast worden. Een veelbelovende richting is het tegelen van complexe signalen over oppervlakken.

We hopen dat deze thesis verder onderzoek inspireert en dat toekomstige tegelgebaseerde methodes hoektegels zullen overwegen als een waardig alternatief voor Wangtegels.

Bibliografie

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

Cohen, M. F., Shade, J., Hiller, S., and Deussen, O. Wang tiles for image and texture generation. *ACM Transactions on Graphics*, pages 287–294, 2003.

Cook, R. L. Stochastic sampling in computer graphics. *Computer Graphics (Proceedings of ACM SIGGRAPH 86)*, 5(1):51–72, 1986.

Culik, II, K. An aperiodic set of 13 Wang tiles. *Discrete Mathematics*, 160(1-3):245–251, 1996.

Deussen, O., Hanrahan, P., Lintermann, B., Měch, R., Pharr, M., and Prusinkiewicz, P. Realistic modeling and rendering of plant ecosystems. In *Proceedings of ACM SIGGRAPH 1998*, pages 275–286. 1998.

Dippé, M. A. Z. and Wold, E. H. Antialiasing through stochastic sampling. *Computer Graphics (Proceedings of ACM SIGGRAPH 85)*, 19(3):69–78, 1985.

Dunbar, D. and Humphreys, G. A spatial data structure for fast Poisson-disk sample generation. *ACM Transactions on Graphics*, 25(3):503–508, 2006.

Grünbaum, B. and Shepard, G. C. *Tilings and patterns*. W. H. Freeman and Company, 1986.

Hiller, S., Deussen, O., and Keller, A. Tiled blue noise samples. In *Vision, Modeling, and Visualization 2001*, pages 265–272. 2001.

Jones, T. R. Efficient generation of Poisson-disk sampling patterns. *Journal of Graphics Tools*, 11(2):27–36, 2006.

Kopf, J., Cohen-Or, D., Deussen, O., and Lischinski, D. Recursive Wang tiles for real-time blue noise. *ACM Transactions on Graphics*, 25(3):509–518, 2006.

Lagae, A. and Dutré, P. A procedural object distribution function. *ACM Transactions on Graphics*, 24(4):1442–1461, 2005a.

Lagae, A. and Dutré, P. Template Poisson disk tiles. Report CW 413, Department of Computer Science, K.U.Leuven, Leuven, Belgium, 2005b.

Lagae, A. and Dutré, P. An alternative for Wang tiles: Colored edges versus colored corners. *ACM Transactions on Graphics*, 25(4):1442–1459, 2006a.

Lagae, A. and Dutré, P. A comparison of methods for generating Poisson disk distributions. Report CW 459, Department of Computer Science, K.U.Leuven, Leuven, Belgium, 2006b.

Lagae, A. and Dutré, P. Generating well-distributed point sets with a self-similar hierarchical tile. Report CW 462, Department of Computer Science, K.U.Leuven, Leuven, Belgium, 2006c.

Lagae, A. and Dutré, P. Long period hash functions for procedural texturing. In *Vision, Modeling, and Visualization 2006*, pages 225–228. Akademische Verlagsgesellschaft Aka GmbH, Berlin, 2006d.

Lagae, A. and Dutré, P. Poisson sphere distributions. In *Vision, Modeling, and Visualization 2006*, pages 373–379. Akademische Verlagsgesellschaft Aka GmbH, Berlin, 2006e.

Lagae, A. and Dutré, P. The tile packing problem. Report CW 461, Department of Computer Science, K.U.Leuven, Leuven, Belgium, 2006f.

Lagae, A., Kari, J., and Dutré, P. Aperiodic sets of square tiles with colored corners. Report CW 460, Department of Computer Science, K.U.Leuven, Leuven, Belgium, 2006.

Lloyd, S. P. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

McCool, M. and Fiume, E. Hierarchical Poisson disk sampling distributions. In *Proceedings of Graphics Interface '92*, pages 94–105. 1992.

Mitchell, D. P. Generating antialiased images at low sampling densities. *Computer Graphics (Proceedings of ACM SIGGRAPH 87)*, 21(4):65–72, 1987.

Ng, T.-Y., Wen, C., Tan, T.-S., Zhang, X., and Kim, Y. J. Generating an ω-tile set for texture synthesis. In *Proceedings of Computer Graphics International 2005*, pages 177–184. 2005.

Ostromoukhov, V., Donohue, C., and Jodoin, P.-M. Fast hierarchical importance sampling with blue noise properties. *ACM Transactions on Graphics*, 23(3):488–495, 2004.

Penrose, R. The rôle of aesthetics in pure and applied mathematical research. *Bulletin of the Institute of Mathematics and its Applications*, 10:266–271, 1974.

Perlin, K. An image synthesizer. *Computer Graphics (Proceedings of ACM SIGGRAPH 85)*, 19(3):287–296, 1985.

Perlin, K. Improving noise. *ACM Transactions on Graphics*, pages 681–682, 2002.

Secord, A., Heidrich, W., and Streit, L. Fast primitive distribution for illustration. In *Proceedings of the 13th Eurographics workshop on Rendering*, pages 215–226. 2002.

Shade, J., Cohen, M. F., and Mitchell, D. P. Tiling layered depth images. Technical report, University of Washington, Department of Computer Science and Engineering, 2000.

Stam, J. Aperiodic texture mapping. Technical Report ERCIM-01/97-R046, European Research Consortium for Informatics and Mathematics (ECRIM), 1997.

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

Wang, H. Games, logic and computers. *Scientific American*, 213(5):98–106, 1965.

Wei, L.-Y. Tile-based texture mapping on graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 55–63. 2004.

Yellot, Jr., J. I. Spectral analysis of spatial sampling by photoreceptors: Topological disorder prevents aliasing. *Vision Research*, 22:1205–1210, 1982.