Aperiodic Texture Mapping

Jos Stam
VTT, Information Technology
P.O. Box 1203, FIN-02044-VTT
Tekniikantie 4 B, Espoo, Finland

Abstract

In this paper we introduce a new technique to texture surfaces. Our approach has the advantage that the texture does not have to be computed at each point of the surface. Rather, we precompute a small set of texture maps which we then map aperiodically onto the surface. The algorithm borrows from recent mathematical results in the area of aperiodic tilings of the plane. To demonstrate the feasibility of the method we apply it to the rendering of water surfaces and caustics.

Key Words & Phrases: Texture Mapping. Aperiodic Tilings of the Plane. Natural Phenomena. Texture Synthesis. Spectral Synthesis. Convolutions.

Supported by an ERCIM (European Research Consortium for Informatics and Mathematics) Fellowship. This work was done while the author was visiting VTT from July 1996 to March 1997.

1. Introduction

The objects that inhabit our visual field are usually characterized by both their shape and their texture. This separation lies at the heart of most computer graphics modeling tools. Typically, these tools allow an animator to design the shape of an object using free form surfaces. Subsequently, visual detail is added to the shape by *texture mapping* the surface. A texture map can modify any of the surface's characteristics, e.g., its reflectivity. Texture maps can also be used to coat a surface with a translucent layer. The latter technique is known as *texel* mapping [Kaj89] or *hypertexture* [Per89] and has been used to model fur and foliage. In this paper we restrict ourselves only to the class of *homogeneous* textures. One property of these textures is that their appearance is similar across the surface. Important examples of phenomena which can be modeled in this way are ocean waves, fur and foliage. Homogeneous textures arise naturally in situations where an animator wants to map a certain texture ("rusty look") onto a given shape ("an old vase"). Texture mapping techniques can be roughly classified into three groups. In the first group, the texture is defined on a two dimensional domain (typically a square) and mapped onto a surface. In the second, the texture is a function called a *solid texture* defined for each point in space [Per85,Pea85]. In the third group, the texture is "grown" on the surface itself, for example by using a reaction diffusion process [Tur91]. In this paper we focus only on the first group of mappings. We describe an efficient algorithm to texture map a homogeneous texture defined on a square domain onto a surface. For large surfaces, such as an ocean, texture maps can become prohibitively expensive to store, especially if these texture are allowed to evolve over time. A possible solution is to generate a single texture map with periodic boundaries (typically using a spectral synthesis technique, see [Mas87]). However, the periodicity is often too noticeable and makes the resulting surface look artificial and too man-made (see Figure 5). In this paper we introduce the concept of *aperiodic textures*: a texture map which is comprised of a constant number (16) of tiles which, once generated, can be mapped aperiodically onto a surface of any size. The aperiodicity guarantees that no unnatural artifacts become visible. Our technique is based on recent results from the mathematical theory of tilings [GrSh86].

In the next section we briefly mention the mathematical results on which our method is founded. In Section 3 we explain how we generate the basic tiles of our homogeneous textures. Then, in Section 4 we provide several examples of the use of our technique. We apply our technique to the problem of creating aperiodic textures of water surfaces and their corresponding caustics. Extensions and future work are the subject of Section 5.

2. Aperiodic Tiles

The subject of how to tile a given set of tiles onto the entire plane is a beautiful mathematical subject which is examined in depth in Grünbaum and Shephard's book "Tilings and Patterns" [GrSh86]. Chapter 11 of this book is devoted to the problem which concerns us most here, i.e., how to tile the plane aperiodically using a finite set of tiles. At the time of the publication of their book the smallest number of such tiles was found by R. Amman to be 16. Recently, however, this number has been brought down to 13 [Cul96]. However, no constructive algorithm is given in that paper. Therefore, we will use the tiling method described in [GrSh86], which uses the 16 tiles shown in Figure 1. This set of tiles uses six different colors for the edges. Tiling the plane using copies of these tiles can be compared to a colored domino game, with only tiles having an edge of the same color being allowed to lie side by side. The beauty of the mathematical theory is that it provides us with an effective way of putting copies of these tiles side by side in a consistent manner while covering the entire plane. The details of the derivation are omitted in this paper, and we refer the reader interested in such a derivation to reference [GrSh86]. All that is required to tile the plane is the set of substitution rules given in Figure 2. By starting from any tile and recursively replacing each tile by its rule, it is possible to generate a tiling of any desired size. Although the

substitution may appear tricky at first since not all rules yield the same number of tiles, the irregularity fortunately is consistent, i.e., if some rule yields a 2x1 substitution tiling, then all substitution on the same line will have a 1x1 substitution or a 2x1 substitution. This point is further clarified in Figure 3. The relation between these tiles and homogeneous texture mapping is discussed in the next section.

3. Homogeneous Texture Tiles

The relation between the colors of the tiles described in the previous section and the continuity of the boundaries of the texture tiles is straightforward. The colors of the tiles in the mathematical theory correspond to the boundaries of the texture tiles. Note that the transition of the texture across the boundary corresponding to two tile edges should be smooth. We address how to achieve this continuity in this section. Let N denote the size of the side of the texture tile, i.e. the tile has NxN elements. We assume that the texture can be computed as the transformation of an uncorrelated noise (array of random values). The value of the texture at a point (i,j) is obtained through an arbitrary function which depends only on the (2K+1)x(2K+1) points in the noise centered at (i,j), with K<N/2. This assumption is not very restrictive since it includes most solid textures and convolutions. The function should of course be the same at each point in accordance with our assumption of homogeneity. Hence, in order to generate a texture tile we need an input noise of size (N+2K)x(N+2K). To guarantee smooth transitions between the boundaries of two different texture tiles corresponding to two tiles sharing an edge of the same color, we have to ensure that the same input noise is used to compute the texture at the edges. Consequently, the input noise of each tile is subdivided as shown in Figure 4. We can identify three different types of regions: (I) four square “corners” of size 2Kx2K, (II) four rectangular “edges” of size 2Kx(N-2K) or (N-2K)x2K and (III) one square “center” of size (N-2K)x(N-2K). Regions of the first type are shared by four arbitrary adjacent tiles. Consequently the same input noise of size 2Kx2K is used for all four corners of each texture tile. The regions of the second type correspond to the colors of the equivalent tiles. We generate a 2Kx(N-2K) input noise for each color. If the color appears horizontally, then the noise coordinates are transposed. Finally, the third region is unique to each texture tile and assures that each of the 16 tiles has a different appearance. The choice of the sizes N and K depends on the nature of the transformation and on the hardware available (in our examples N=128 and K=16). In the next section we demonstrate how our new method can be used to simulate water waves and their associated caustics.

4. An Application: Waves and Caustics

We have chosen to apply our algorithm to the depiction of water waves and the caustics they produce by reflecting and refracting incoming light onto nearby surfaces. Ocean scenes typically require that a large portion of the ocean be visible at any time. Instantiating the entire geometry defining the sea surface is clearly too costly. Fortunately by using our method, only 16 texture-tiles have to be precomputed. The tiling algorithm described in Section 2 can then be used to instantiate the different texture-tiles. We used the Radiance ray tracer to render pictures of a “wine-dark sea” surface (see Figure 9) [Homer]. The Radiance program has a built in feature which allows geometry to be stored in an octree and then be instantiated many times with a minimal overhead in storage [War94]. To compute an animation of an evolving surface, we created an ensemble of time series by convolving a white noise in the frequency domain and then taking an inverse transform [NRC88]. The input noise used to compute our tiles thus consists of temporally correlated time series. Note also that they remain uncorrelated in the spatial domain. Since the input noise is periodic in time due to the spectral synthesis technique, the resulting tiles are also periodic in time. Therefore, a finite number of tiles provides us with endless animations. Note that it is not possible to generate aperiodic tilings in one dimension [Moz89]. Choosing enough

samples in the time series makes it hard for an observer to spot the time-periodicity.

To model caustics, we computed the caustic map produced by each water surface texture-tile for a given depth. This was achieved using the hardware compositing feature available on an Iris SGI workstation (this takes less than a second for a resolution of 256x256). For each triangle of the surface-tile we accumulate the intensity of the refracted triangle into an alpha buffer. The intensity of the triangle is directly proportional to its area. The caustic map is usually larger than the initial texture as shown in Figure 6. By compositing the caustic textures from adjacent tiles we obtain caustics maps of any desired size (Figure 6). Figure 7 shows a frame of an animation of a pool scene. We computed the caustics for both the bottom and the sides of the pool. The bottom of the pool consists of 40 copies of our 16 basic tiles. In Figure 8 we depict a closer view of the beautiful caustics on the bottom of the pool.

5. Future Work

We have shown that aperiodic tiling algorithms can have useful applications in the area of mapping homogenous textures onto large surfaces. We have reported only results relating to water waves and caustics, however the technique is in no way restricted to these phenomena. In particular, we intend to apply this technique to time evolving texels. An obvious extension is to extend the tiling methods to three-dimensions. However, according to [GrSh86] research in this area is still in its infancy and no mathematically rigorous proofs that a given three-dimensional tiling is aperiodic has been given. Three-dimensional tiling could be used to efficiently compute solid textures for example. Another possible line of research would be to decrease the number of tiles required (16 in our case) by either making the method in [Cul96] constructive or by considering tiles with non-square shapes. The smallest number of polygons known to aperiodically tile the plane is two. For example, one could use Penrose's famous "kite" and "darts" [GrSh86]. However, generating textures on non-square polygons while satisfying the required boundary conditions is clearly a non-trivial problem. We are also investigating automatic procedures to generate our texture tiles from photographs of real textures as in [HeBe95].

6. References

[Cul96] K. Culik II, "An Aperiodic Set of 13 Wang Tiles", *Discrete Mathematics* 160 (1996), pp. 245-251.

[GrSh86] B. Grünbaum and G. C. Shephard, *Tilings and Patterns*, Freeman, New York, 1986.

[HeBe95] D. J. Heeger and J. R. Bergen, "Pyramid-Based Texture Analysis/Synthesis", *Computer Graphics Proceedings (SIGGRAPH'95), Annual Conference Series*, 1995, pp.229-238.

[Homer] The ancient Greeks never described the sea as being blue. For example, Homer in *The Odyssey* talks of "the wine-dark sea" (transl. T.E. Lawrence, Wordsworth Ed. Ltd., Great Britain, 1992).

[Kaj89] J. T. Kajiya and T. L. Kay, "Rendering Fur Wih Three Dimensional Textures", Proceedings of SIGGRAPH'89. In *Computer Graphics* (1989), vol. 23, ACM SIGGRAPH, pp. 271-280.

[Mas87] G. A. Mastin, P. A. Watterberg, and J. F. Mareda, "Fourier Synthesis of Ocean Scenes", IEEE Computer Graphics & Applications, March 1987, pp. 16-23.

[Moz89] S. Mozes, "Tilings, Substitution Systems and Dynamical Systems Generated by Them", *Journal d'Analyse Mathématique*, vol. 53, 1989, pp. 139-186.

[Pea85] D. R. Peachy, "Solid Texturing of Complex Surfaces", Proceedings of SIGGRAPH'85. In *Computer Graphics* (1985), vol. 19, ACM SIGGRAPH, pp. 279-286.

[Per85] K. Perlin, "An Image Synthesizer", Proceedings of SIGGRAPH'85. In *Computer*

Graphics (1985), vol 19, ACM SIGGRAPH, pp. 287-296.

[Per89] K. Perlin and E. M. Hoffert, "Hypertexture", Proceedings of SIGGRAPH'89. In *Computer Graphics* (1989), vol. 23, ACM SIGGRAPH, pp. 253-261.

[NRC88] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes in C. The Art of Scientific Computing*. Cambridge University Press, Cambridge, 1988.

[Tur91] G. Turk, "Generating Textures on Arbitrary Surfaces Using Reaction-Diffusion". Proceedings of SIGGRAPH'91. In *Computer Graphics* (1991), vol. 25, ACM SIGGRAPH, pp. 289-298.

[War94] G. Ward, "The Radiance Lighting Simulation and Rendering System", *Computer Graphics Proceedings (SIGGRAPH'97), Annual Conference Series*, 1994, pp. 459-472.

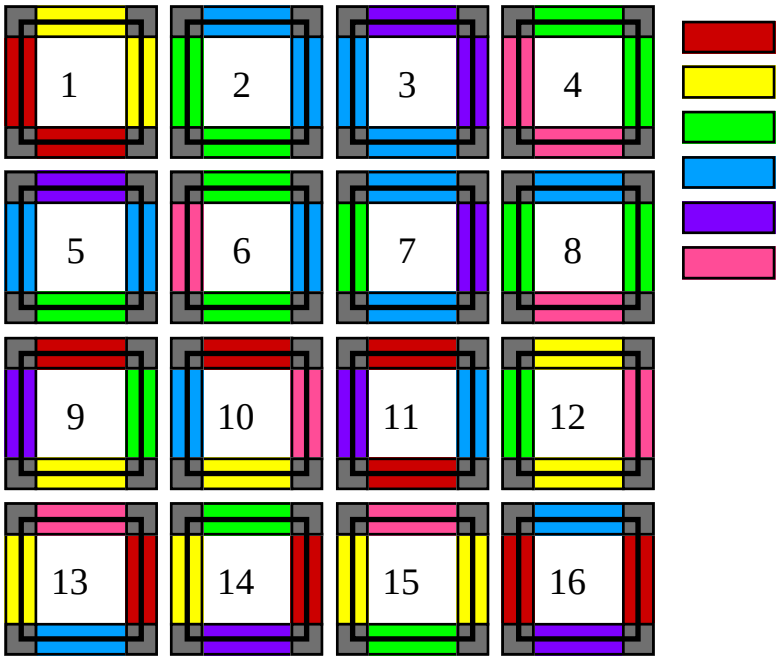

Figure 1: The 16 tiles along with their coloured edges. There are 6 different colours.

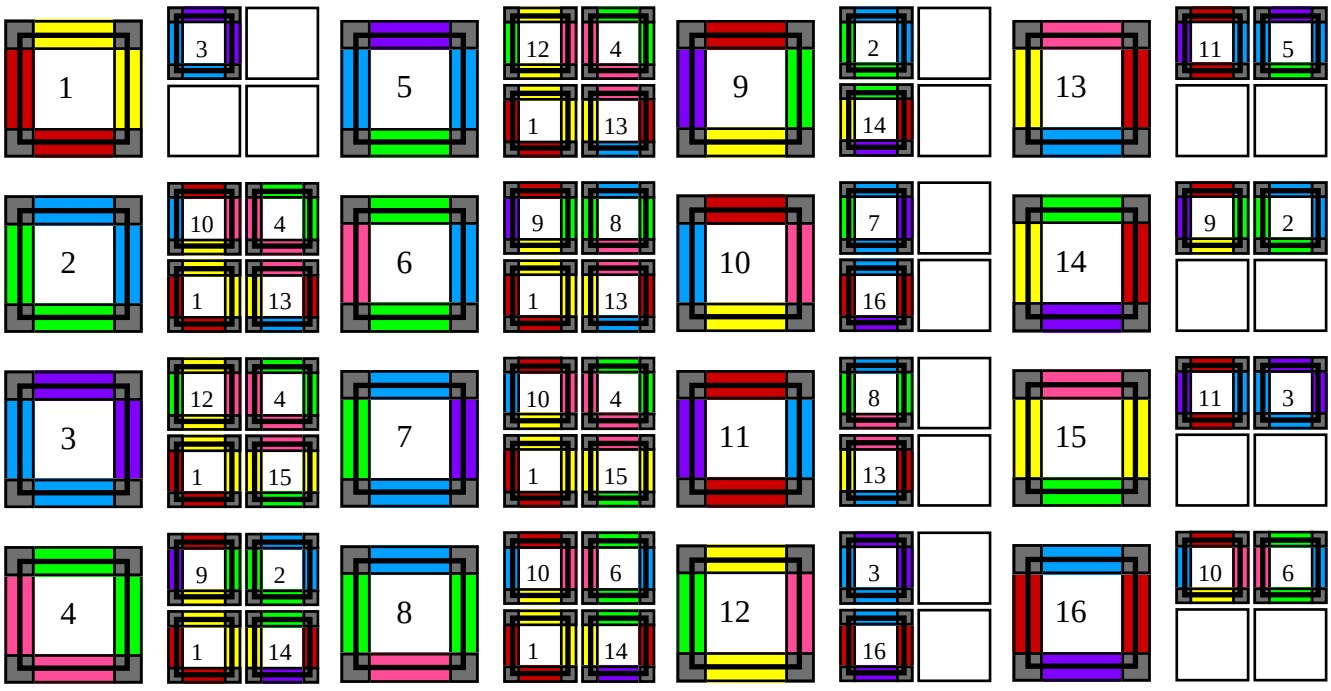

Figure 2: The 16 substitution rules. At each iteration the larger tile is replaced by the smaller tiles directly adjacent to it.

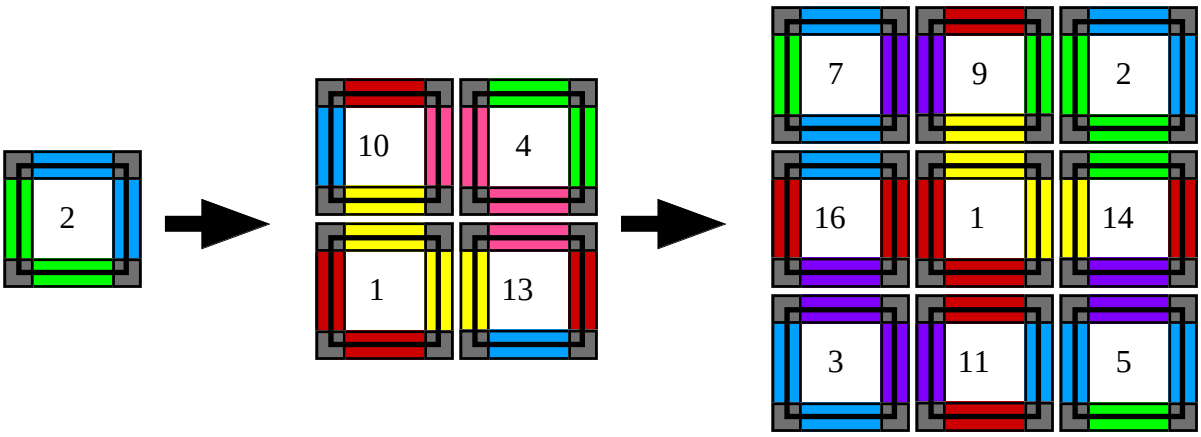

Figure 3: Substitution rules applied three times starting with tile number 2. See Figure 2 for the corresponding substitution rules.

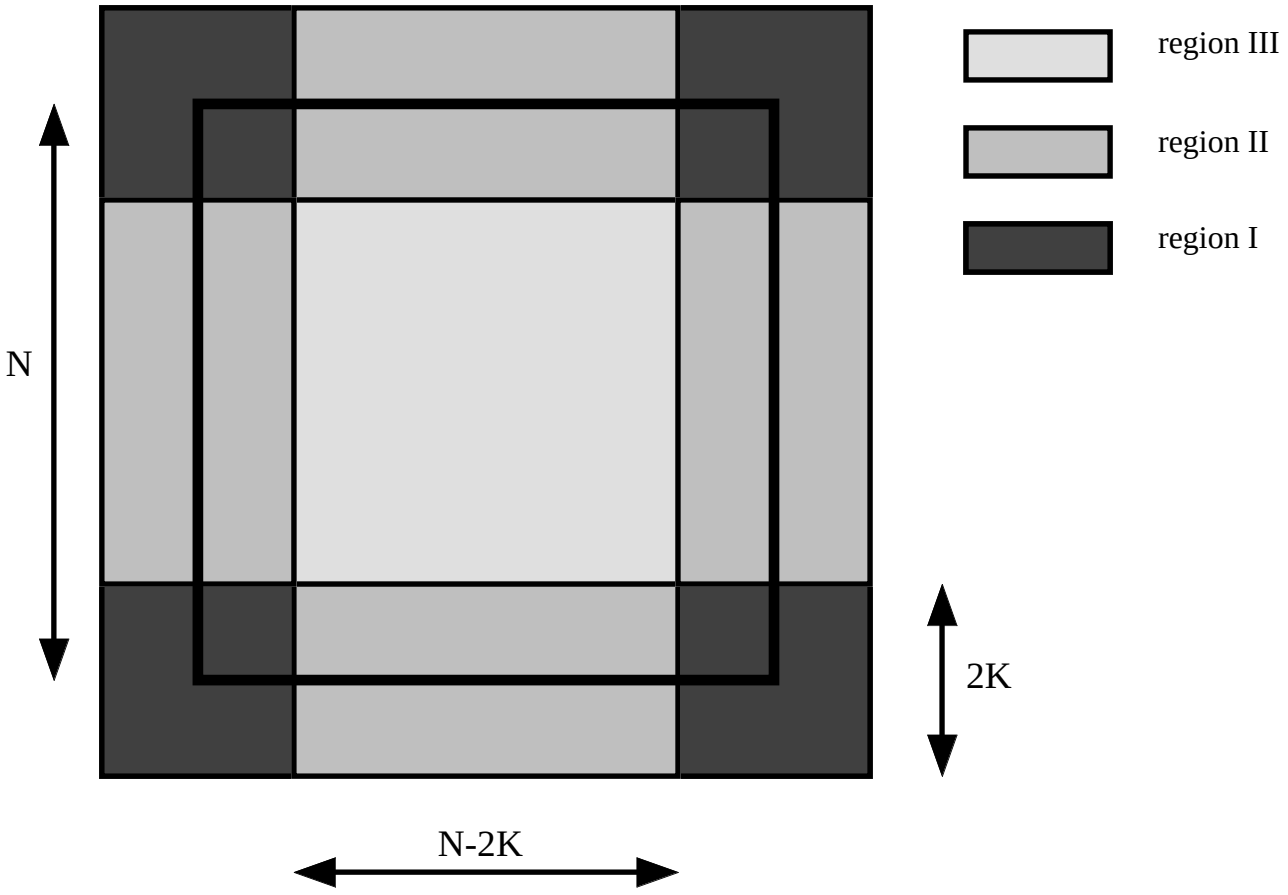

Figure 4: This figure shows the three different regions of input noises for each of the 16 tiles. The 2Kx2K noise blocks are the same for each tile. The (N-2K)x2K and 2Kx(N-2K) blocks of region II correspond to the different colours of the tiles. The (N-2K)x(N-2K) block of region III is specific to each tile.

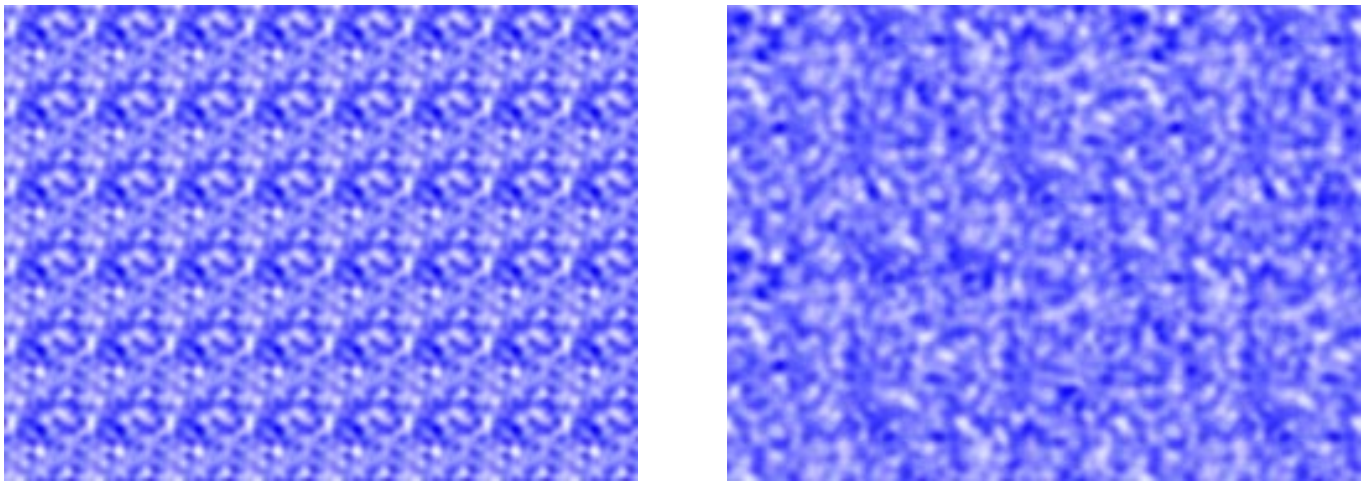

Figure 5: Periodic texture mapping versus aperiodic texture mapping. The texture on the left was generated by tiling the same periodic tile 48 times. The texture on the right was generated using 48 copies of our 16 aperiodic tiles. Notice that no artefacts are present in the aperiodic texture.

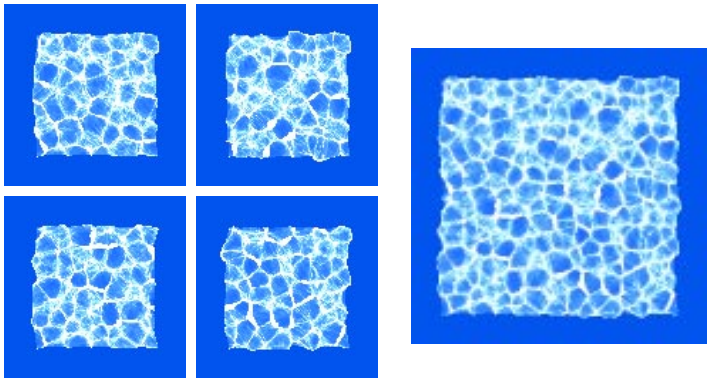

Figure 6: For each of the 16 tiles used to create the water surface, we compute the corresponding caustic texture. The caustic map on the right is obtained by compositing the four maps on the right from rule 3−>{12,4,1,15}.

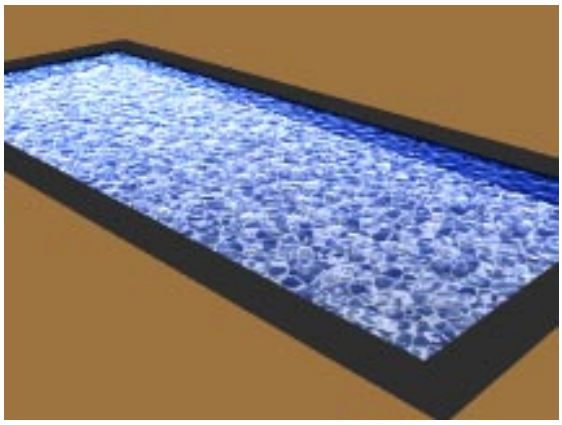

Figure 7: Ray traced image of a pool. We tiled the textured shown in Figure 6 aperiodically on the bottom of the pool.

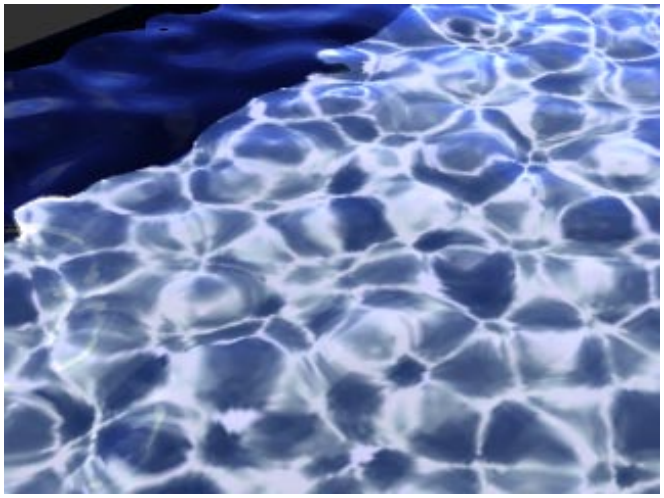

Figure 8: Closer view of the beautiful caustics on the bottom of the pool.

Figure 9: "The Wine−Dark Sea". Ray traced image of an ocean. We instantiated 3600 tiles to create the ocean's surface.