

Procedural Generation of Rock Piles using Aperiodic Tiling

A. Peytavie¹, E. Galin², J. Grosjean³, S. Merillou⁴.

¹LIRIS - CNRS - Université Claude Bernard Lyon 1, France

²LIRIS - CNRS - Université Lumière Lyon 2, France

³LSIIT - CNRS - Université Louis Pasteur Strasbourg, France

⁴XLIM - CNRS - Université de Limoges, France

Abstract

In this paper, we present a tiling method for generating piles of rocks without any computationally demanding physically-based simulation. Previous techniques rely on a periodic tiling of rocks and generate unrealistic repetitive patterns. In contrast, our approach relies on a modified corner cube algorithm to generate a set of aperiodic tiles. We generalize the construction method so that the geometry of rocks should straddle corner cubes with a view to avoiding unrealistic gaps in the arrangement of rocks. Moreover, we propose an original technique to control the shape of rocks into contact by computing the Voronoï cells using a parameterized anisotropic distance. Our method has been successfully used to generate landscapes and stone huts and walls with thousands of rocks piled together.

Categories and Subject Descriptors (according to ACM CCS): [Computer Graphics]: Three-Dimensional Graphics and Realism

Keywords: Procedural modeling, natural phenomena, aperiodic tiling.

1. Introduction

Modeling and rendering realistic images of natural landscapes is an important problem in computer graphics. Beautiful images of complex terrains covered with vegetation have been produced by many computer graphics researchers and artists in the film industry. Unfortunately, the rendered scenes often lack many details, which betrays the synthetic nature of the scene.

A vast variety of techniques have been proposed for generating and rendering terrains, simulating ecosystems and creating realistic plant models, simulating aging and weathering phenomena. In contrast, generating details such as rocks, stones, fallen leaves or branches covering the ground remains an open area of research.

In this paper, we focus on rocks and stones that are conspicuous in nature and may be found everywhere in landscapes such as in rocky mountains, deserts, at the bottom of cliffs, sea shores or by river banks. They play an important part in the realism of a natural scene, as their presence provides the viewer with a hint about the age of a rocky scene as well as indirect indications about the erosion and the characteristics of the environment.



Figure 1: Dried stone huts created with aperiodic tiling

1.1. Related work

The challenge stems not only from the complexity and diversity of rock shapes, but also from the huge number of rocks in contact and piled together in a virtual scene.

Terrain generation There are three main approaches to generating synthetic terrains: fractal landscape modeling [MKM89, ST89], physical erosion simula-

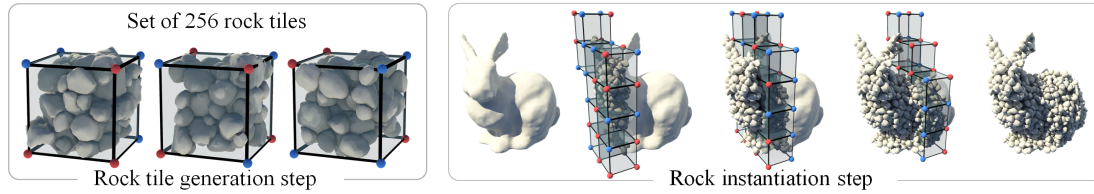


Figure 2: Synthetic overview of the rock generation process, the bunny model is used as the support object

tion [KMN88, NWD05, MDH07] and terrain synthesis from images or sample terrain patches [ZSTR07]. While those techniques are efficient for creating large scale terrains, they do not generate the geometry of rocks and rely on procedural texturing methods or displacement maps to render rocks.

Rock generation In general, a few rocks models are created by artists and located in the scene by hand, which is impractical for generating thousands of rocks. Relying on only a few instances produces unrealistic replication artifacts. While collision detection techniques can be used to create piles of rocks, this approach is computationally demanding and difficult to control.

[MIS01] proposed a texture based tiling method combined with a bump mapping scheme to create stones paving a path. The method creates stones by a displacing a two-dimensional map and can't create rock piles.

[Miy90] proposed a procedural model for generating stone walls. [LDG01] presented a general cellular pattern based technique for generating stone walls according to the architectural settings and the geometric features of the underlying shape. This method tends to generate regular patterns on surfaces suited for architectural models, whereas we focus on aperiodic volumetric distributions for natural sceneries.

Aperiodic tiling Several two-dimensional aperiodic tiling techniques [CSHD03, BPB09, SKP05] have been proposed to distribute natural objects such as trees over the surface of a terrain in a natural way with a view to avoiding repetitive patterns. Wang cubes [LEQ*07] have been successfully used for volume illustration applications. Poisson sphere distributions [LD06a] based on a three-dimensional aperiodic tiling of points have been successfully used to locate instances of objects in space. The main limitation of those techniques is that instances are separated by a minimum distance and can't be into contact.

Recently, [PGMG09] proposed an original tiling technique for generating rock piles from a material layer representation of terrains. The proposed method has two major weaknesses however. The periodic tiling scheme produces unrealistic patterns, and there is no control over the shape of rocks that are generated.

1.2. Contributions

In this paper, we present an original aperiodic tiling scheme for generating rocks into contact based on a modified corner cube [LD06a] generation algorithm. Our method proceeds in two steps (Figure 2). First, we generate a set of aperiodic tiles of rocks while guaranteeing contact between neighboring rocks. This step is performed once and for all as a pre processing step. Rock piles are then generated by virtually tiling space with the precomputed set of aperiodic tiles and instancing only those rocks that intersect a support object.

We present an original method for controlling the shape of rocks by computing the Voronoï cells generated by a set of seed points using a class of anisotropic distance function. Our technique enables us to generate different kinds of models, including flat sharp edged rock or smooth and round pebbles.

The remainder of this paper is organized as follows. Section 2 presents an overview of our aperiodic tiling generation algorithm and notations. Sections 3, 4 and 5 describe the computation of the set of aperiodic tiles, the computation of the bounding volumes of rocks and the erosion process used to compute the final rock shapes. Section 6 presents how rocks piles are created from a support shape. Section 7 presents some results before we finally conclude this work.

2. Overview and notations

In this section, we present an overview of our method for generating a set of aperiodic tiles of rocks into contact.

Periodic tiling Let us recall the fundamentals of the periodic algorithm presented in [PGMG09]. First, a set of points \mathbf{p}_k is distributed inside a cubic tile \mathcal{C} . By considering that the cube \mathcal{C} virtually tiles space, it is possible to construct closed Voronoï cells, denoted as \mathcal{V}_k for every point \mathbf{p}_k . The resulting set of cells \mathcal{V}_k periodically tiles space. The Voronoï cells give a good approximation of convex rock shapes into contact. The final geometry of the rocks is created by eroding the surface of Voronoï cells almost everywhere but at some random contact points located on the faces so that rocks should nicely pack together.

An important limitation of this approach is that unnatural repetitive patterns appear, the more so as the ground surface is smooth as illustrated in Figure 3.



Figure 3: Unrealistic patterns with periodic tiling



Figure 5: Aperiodic tiling avoids visible artifacts

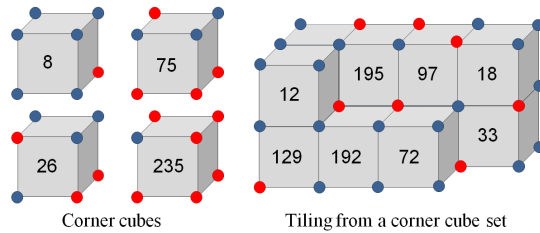


Figure 4: A number of different corner cubes from a complete set of 256 configurations, and a tiling generated from the set of corner cubes

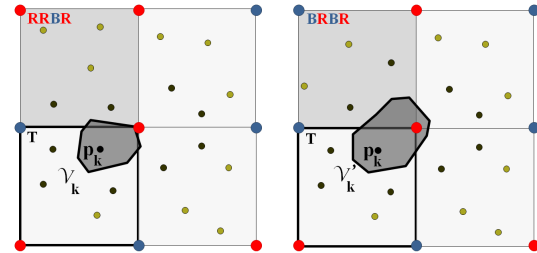


Figure 6: Two different rock volumes \mathcal{V}_k and \mathcal{V}'_k can be generated for a single seed point \mathbf{p}_k but with different compatible corner cubes

Corner Cubes Recall that corner cubes [LD06a] are an extension of corner tiles [LD06b] to three dimensions. Corner cubes are cubic tiles with colored corners. Similar to Wang tiles [CSHD03], the cubes have a fixed orientation. A complete set of corner cubes over 2 colors consists of $2^8 = 256$ cubes. One corner cube is characterized by the color of its 8 corner vertices (Figure 4). A tiling is generated by placing the cubes next to each other such that adjoining corners have matching colors.

The aperiodic tiling problem Corner cubes, which have been proposed to generate Poisson Sphere distributions, could be a means to generate a set of aperiodic tiles of points \mathbf{p}_k . Unfortunately, it is not possible to create the rock volumes \mathcal{V}_k directly from the set of aperiodic tiles in a consistent way. The reason for this is that the rock volume generation technique, although local, is based on the computation of the Voronoi cells of points \mathbf{p}_k . Let us consider a corner cube \mathcal{C} , the geometry of a volume \mathcal{V}_k depend on the neighbors of its seed point \mathbf{p}_k , which can belong to a different neighboring corner cube.

Figure 6 illustrates the Voronoi cell straddling problem in two dimensions: two different rock volumes \mathcal{V}_k and \mathcal{V}'_k are generated for the same seed point \mathbf{p}_k depending on the neighboring corner cubes whose configurations, denoted as **BRBR** and **RRBR** respectively, are different albeit compatible with the corner colors of the corner cube \mathcal{C} .

Corner cube grids We propose a new structure, corner cubes grids, as a means to solve the Voronoi cell straddling problem. By decomposing the corner cubes into a grid of n^3 cells \mathcal{C}_{ijk} , we can keep track of the neighborhood of seed points into the corner cube generation scheme and propose a construction process that avoid ambiguities in the creation of the Voronoi cells.

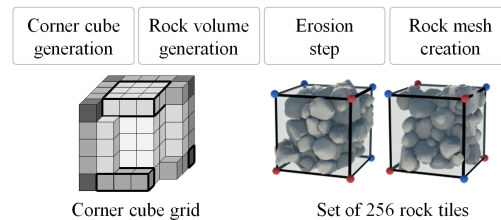


Figure 7: Overview of our rock generation technique

Our method enables us to create a set of aperiodic rock tiles, with rocks straddling the corner cubes and consistent together. This results in more natural rock distributions, as illustrated in Figure 5.

Overview Our method, as illustrated in Figure 7, proceeds into four steps:

1. Generate a set of 256 corner cubes, which are decomposed into a grid of cells, denoted as \mathcal{C}_{ijk} , and create a set of seed points \mathbf{p}_k for each cell.

2. Create a rock volume \mathcal{V}_k for every seed point \mathbf{p}_k by computing the Voronoi cells from the set of vertices \mathbf{p}_k using an anisotropic distance function.
3. Generate an implicit representation of the rocks by eroding the Voronoi cells \mathcal{V}_k using spheroidal erosion [BFO*07].
4. Finally, create the mesh representation by polygonizing the implicit surface model of the different rocks.

3. Corner cube grid generation

In this section, we present the corner cube grid generation process. Without loss of generality, we assume corners have 2 colors, thus we aim at generating a set of 256 cubic tiles.

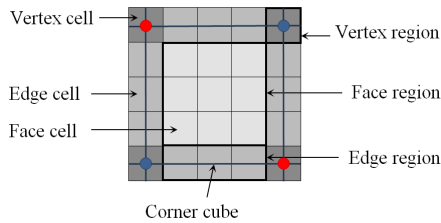


Figure 8: The corner cube grid structure ($n = 5$)

Let \mathcal{C} denote a generic corner cube. We decompose \mathcal{C} into a grid of n^3 cells denoted as \mathcal{C}_{ijk} . The different cells \mathcal{C}_{ijk} will be referred to as vertex, edge, face and center cells (Figure 8). We decompose the corner cube grid into a set of different regions which will be referred to as vertex, edge, face and center regions. They are defined as the 8 corner cells, the $n - 2$ cells along the 12 edges of the corner cube, the $(n - 2)^2$ cells on the 6 faces of the corner cube and the remaining $(n - 2)^3$ cells inside the cube respectively.

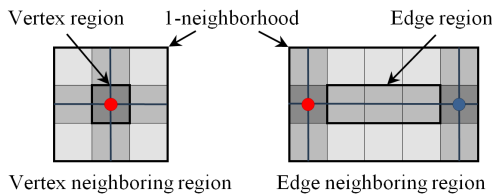


Figure 9: Two-dimensional representation of the neighboring regions of a vertex region and an edge region

We define the neighboring cells of a given region as the cells in the 1-neighborhood of the cells of this region (Figure 9). Note that the 1-neighborhood of a region contains cells outside of the corner cube. This property, along with the seed point generation algorithm, play an important part in the construction of the 256 corner cube grids as they are used to keep track of the points \mathbf{p}_k already created to avoid ambiguous configurations in the definition of volumes \mathcal{V}_k .

Creation of seed points During the generation of the 256 corner cubes, we create the random distribution of seed points \mathbf{p}_k in the different cells \mathcal{C}_{ijk} of the neighboring regions. One or more seed points may be created in the cells. Let s denote the maximum number of seed points per cell and let n_k denote the number of seed points in a cell. Seed points \mathbf{p}_k are created using a low discrepancy sequence in every cell. As we will see in the next section, the parameter n_k defines the relative density of seed points within a cell, thus the relative size of the Voronoi volumes \mathcal{V}_k .

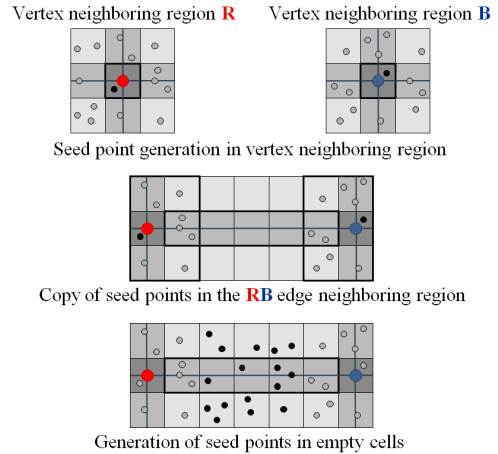


Figure 10: Overview of the first two steps of the seed point generation algorithm for an RB edge

Recall that we present the algorithm for the corner cube grid structure built from 2 different colors. By combinations, to create the 256 corner cube configurations, we need to generate 2 vertex regions, 12 edge regions, 48 face regions and 256 center regions. The seed points \mathbf{p}_k are sequentially created in the vertex, edge, face and center regions. The algorithm proceed as follows:

1. Generate the seed points \mathbf{p}_k in the 2 corner regions and in the neighboring cells.
2. For all the 12 edge regions, copy the seed points \mathbf{p}_k of the vertex neighboring cells. Generate new seed points only within the empty cells of the edge region and the edge neighboring cells.
3. For all the 48 face regions, copy the seed points \mathbf{p}_k of the vertex and edge neighboring cells. Generate new seed points only within the empty cells of the face region and the face neighboring cells.
4. Finally, for all the 256 center regions, copy the seed points \mathbf{p}_k of the vertex, edge and face neighboring cells before generating the new seed points in the empty cells of the center region.

Figure 10 illustrates the process for an RB edge. At the end of the construction process, we create a set of 256 corner cube tiles by assembling the corresponding vertex, edge,

face and center regions. The set of corner cube tiles has a consistent distribution of seed points that guarantee that the rock volumes \mathcal{V}_k will perfectly match, even when straddling the borders of the cubes.

4. Rock volume generation

In this section, we address the computation of the rock volumes \mathcal{V}_k from the seed points.

We define the volumes \mathcal{V}_k as the Voronoi cells of points \mathbf{p}_k . Instead of using the Euclidean distance which creates round shaped rocks, we propose to use an anisotropic distance. Several anisotropic distance functions have been proposed in [CBS96] for modeling implicit primitives. We propose to rely on an ellipsoidal anisotropic distance function to control the overall shape and orientation of stones.

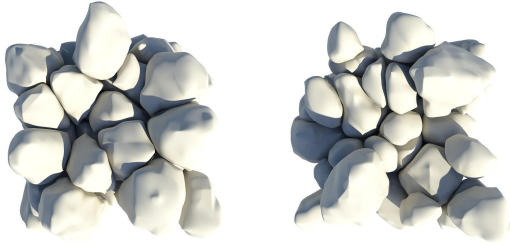


Figure 11: Round shaped rocks generated with $\alpha = 1$, $\beta = 1$, $\gamma = 1$ and $g_k \in [1, 2]$ and one seed point per cell

For every seed point \mathbf{p}_k , we define an anisotropic ellipsoidal distance function, denoted as $d_k(\mathbf{p})$, as follows:

$$d_k(\mathbf{p}) = \frac{1}{g_k} f_k(\mathbf{p})$$

Let \mathbf{x} , \mathbf{y} , \mathbf{z} denote the vectors of the local frame attached to every seed point \mathbf{p}_k , and α , β , γ refer to the parameters that characterize the relative size along the frame axes. The function $f_k(\mathbf{p})$ is defined as:

$$f_k(\mathbf{p}) = \alpha_k((\mathbf{p} - \mathbf{p}_k)\mathbf{x})^2 + \beta_k((\mathbf{p} - \mathbf{p}_k)\mathbf{y})^2 + \gamma_k((\mathbf{p} - \mathbf{p}_k)\mathbf{z})^2$$

The parameters g_k and the number of seeds per cell n_k are used to control the relative size of rocks: the higher n_k , the smaller the Voronoi volumes \mathcal{V}_k . The parameter $g_k > 0$ tunes the relative size of the rocks. In our system, the values of g_k are defined by using a uniform random distribution over a control interval $[1, g_{\max}]$. In our own implementation, we define $g_{\max} = 4$ so as to limit the range of the relative size of rocks.

If n_k has the same value for every cell, then the generated rock volumes \mathcal{V}_k and therefore the final rock models will have approximately the same size (Figure 11). In contrast, if n_k is obtained by randomly sampling the range of values $n_k \in [1, s]$, we create rock volumes with very different sizes, such as rocks and gravel (Figure 12).

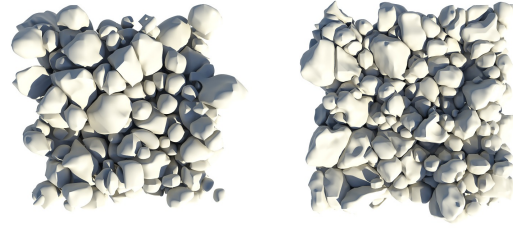


Figure 12: Rocks of different sizes obtained with $\alpha = 1$, $\beta = 1$, $\gamma = 1$, $g_k \in [1, 3]$ and from 1 to $s = 20$ seed points per cell

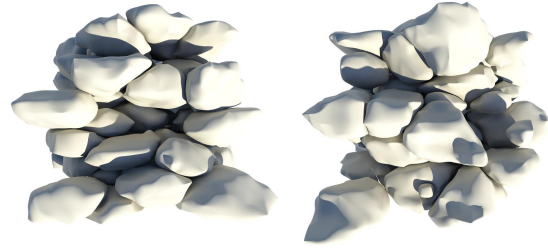


Figure 13: Flat shaped rocks obtained by using $\alpha = 5$, $\beta = 5$, $\gamma = 1$ as scaling coefficients, $g_k \in [1, 3]$ and one seed points per cell

The local frame system $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is generated by uniform random distribution. The coefficients α_k , β_k and γ_k are obtained by using a uniform random distribution over an input interval characterizing the maximum extent of the rocks in every direction. Figure 13 shows several flat rock models (after erosion).

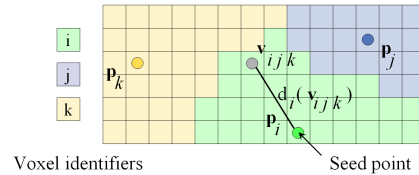


Figure 14: Erosion of the discrete Voronoi volumes

Discrete implementation Since creating the Voronoi cells using anisotropic distances is a very complex problem, we perform the rock volume generation step by computing a discrete Voronoi diagram. We subdivide the corner cubes \mathcal{C} into a grid \mathcal{G} of $n_{\mathcal{G}}^3$ voxels. The parameter $n_{\mathcal{G}}$ defines the resolution of the grid that will be used to compute the discrete Voronoi cells and perform the erosion process which will be presented in the next section. Therefore, $n_{\mathcal{G}}$ should be set high enough to guarantee that the grid can capture all the Voronoi volumes. This is performed by computing the

inverse of the minimum distance between all seed points \mathbf{p}_k .

Let \mathcal{G}_{ijk} denote a voxel of this grid and \mathbf{v}_{ijk} denote the center of the voxel. For every voxel, we first compute a unique identifier that stores the index k of its corresponding seed point \mathbf{p}_k (Figure 14). This step is efficiently performed by evaluating the minimum distance between the center of the voxel and the seed points $d_k(\mathbf{v}_{ijk})$. This discrete representation will enable us to perform the erosion of the rock volumes to generate the final rocks shapes into contact.

5. The erosion process

The rock volume generation process provides us with a voxel based representation of the rock volumes \mathcal{V}_k . The erosion step aims at carving and sculpting the rock shapes out of those coarse volumes while maintaining contact points between the final rock shapes with a view to generating convincing rock piles.

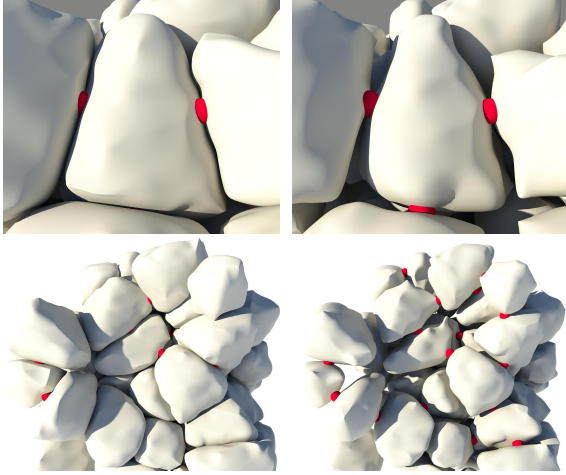


Figure 15: Rocks into contact after the erosion process. The contact points have been represented as red shaded spheres

The erosion process proceeds in four steps. First we generate a set of contact points between the coarse rock volumes. Then we define an attenuation field that will dampen erosion in the neighborhood of contact points (Figure 16). We perform a user controlled number of spheroidal erosion steps [BFO*07] and generate an implicit representation of the rocks. Finally, we create the meshes of the rocks by using standard implicit surface meshing techniques [LC87].

Contact points The contact points between the different rock volumes are created in two steps. We generate a contact point, denoted as \mathbf{c}_{ij} , for every two rock volumes \mathcal{V}_i and \mathcal{V}_j into contact. The contact point is created on the line segment $[\mathbf{p}_i \mathbf{p}_j]$ by finding the position that satisfies $d_i(\mathbf{c}_{ij}) = d_j(\mathbf{c}_{ij})$ using a bisection algorithm.

Note that it is possible to randomly discard a few contact points so as to create more space between rocks. However this may results in some floating rocks. In our system, we keep all the contact points between rock volumes so as to obtain visually plausible stable configurations.

Generation of the erosion damping field The rocks are generated by eroding the rock volumes \mathcal{V}_k while preserving the contact points between those volumes. Thus, erosion must be dampened in the neighborhood of the contact points (Figure 16). For every contact point \mathbf{c}_{ij} , we define the compactly supported damping function $\omega_{ij}(\mathbf{p})$ as a piecewise linear function of the distance to the contact point, $\omega_{ij}(\mathbf{p}) = \omega_{ij}(r)$ with $r = \|\mathbf{p} - \mathbf{c}_{ij}\|$ and:

$$\omega_{ij}(r) = \begin{cases} 0 & \text{if } r < a \\ (r-a)/(b-a) & \text{if } a \leq r < b \\ 1 & \text{otherwise} \end{cases}$$

The parameter a defines the contact radius of the spherical region where no erosion occurs. This parameter enables us to control the size of the contact surfaces between rocks. The parameter b controls the size of the blending region between full erosion and no erosion.

The final damping field $\omega(\mathbf{p})$ is defined as the combination of all the damping fields $\omega_{ij}(\mathbf{p})$ as follows:

$$\omega(\mathbf{p}) = \min_{(i,j) \in [0, n_G - 1]^2} (\omega_{ij}(\mathbf{p}))$$

By using the minimum function we guarantee that no erosion occurs within the spherical region $r < a$.

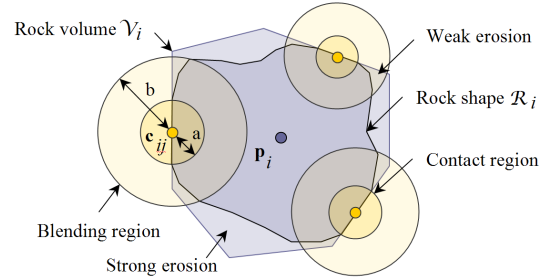


Figure 16: Definition of the erosion damping regions around contact points

Erosion process The erosion process is performed in two steps. First, we initialize the voxels \mathcal{G}_{ijk} of the voxel grid with a parameter denoted as $\rho_{ijk} = 1$ that characterizes the erosion state of the voxel. If $\rho_{ijk} = 0$, then the voxel is completely eroded and becomes empty (Figure 18).

The erosion process is performed by iteratively computing which cells have to be eroded, and evaluating the amount of erosion. Only the voxels at the surface of the rock volume, i.e. that share a face with either another empty voxel or a



Figure 17: A virtual canyon with piles of rocks and boulders lying in the river (25313 rocks)

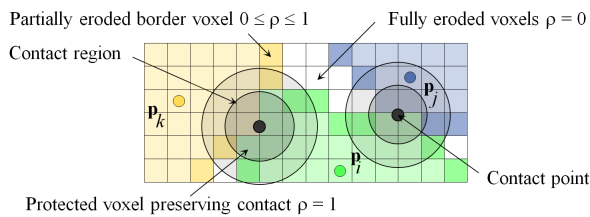


Figure 18: Erosion of the discrete Voronoi volumes

voxel of different identifier, should progressively erode. Let $\epsilon(\mathbf{p})$ denote the amount of erosion at a point \mathbf{p} in space at a given iteration. We define the evolution of the state of a voxel \mathcal{V}_{ijk} as:

$$\rho_{ijk} \leftarrow \rho_{ijk} - \epsilon(\mathbf{p}) \omega_{ij}(\mathbf{p})$$

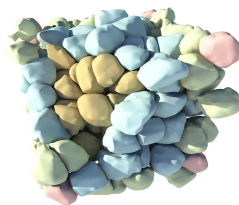


Figure 19: Example of rocks generated into a corner cube grid structure, the rocks have been shaded with same color as the corner cube grid cells and some rocks removed to see the interior of the tile

Mesh generation process For every rock, we identify the voxels in the grid with the corresponding identifier and define an implicit surface representation of its volume. We define the field function values at the centers of the cells \mathbf{v}_{ijk} as:

$$r(\mathbf{v}_{ijk}) = 2\rho_{ijk} - 1 \quad r(\mathbf{v}_{ijk}) \in [-1, 1]$$

The field function $r(\mathbf{p})$ for every point in space is obtained as a tri-linear interpolation of those values. Thus, we easily create the mesh of the rocks by standard implicit surface meshing techniques [LC87].

6. Rock instantiation

Rock piles are created by using a support object \mathcal{O} that defines the shape of the rock pile. We can use any kind of model that provides us with a point membership function, denoted as $s(\mathbf{p})$. In our system, we used implicit representations (Figure 17, 24), procedural models (Figure 22) and closed triangle meshes (Figure 23).

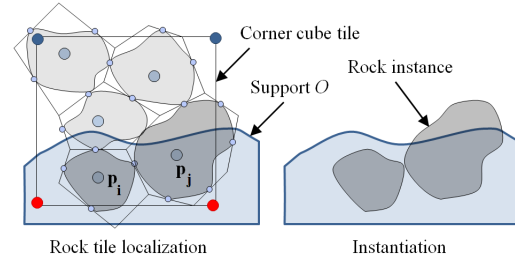


Figure 20: Rock instantiation process

Assuming that the corner cubes are placed with their corners on integer grid points, a stochastic tiling is obtained by assigning a random color to each integer grid point. We generate rocks from the precomputed set of tiles by analyzing corner cubes straddling the object \mathcal{O} (Figure 20).

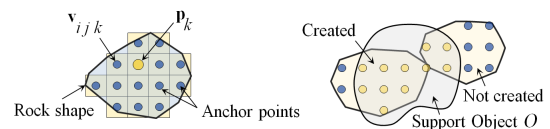


Figure 21: Rock selection process



Figure 22: A Zen garden scene featuring 94658 small stones organized into lines and circular patterns

For every rock, we associate a set of anchor points $\mathcal{A} = \{\mathbf{a}_i\}$ which are defined as the union of the centers \mathbf{v}_{ijk} of the voxels and the seed point \mathbf{p}_k . For all anchor points, we evaluate the point membership function $s(\mathbf{a}_i)$ and compute the number of anchor points inside and outside of the object \mathcal{O} respectively (Figure 21). We instantiate rocks that have a relative number of anchor points inside the object above a given threshold.



Figure 23: A meshed statue converted into rocks

This method enables us to avoid the creation of rocks with only a small intersection with the geometry of the support object. In practice, for large rock piles, we eliminate rocks whose distance to the surface is larger than twice the maximum radius of the Voronoï embedding spheres.

7. Results and discussion

We have implemented our aperiodic rock tiling algorithm into a modeling application coded in C++. We have applied our method to create the different scenes shown throughout this paper: a canyon (Figure 17), a Zen garden (Figure 22) and dried stone huts (Figure 24). Renderings were performed by using Mental Ray on the textured meshes produced by our method.

Realism Our aperiodic tiling method greatly enhances the overall realism of rock piles by avoiding unnatural repetitive distribution patterns that occur with periodic schemes.

Control Because we do not rely on physically based simulations for computing the collisions between rocks, our method lends itself for real time modeling of complex scenes. In particular, it is very efficient for controlling the location of stones and creating special effects. The volumetric model describing how rocks should fit line and circular patterns in the Zen garden was generated procedurally.

The ancient dried stone huts demonstrate that our method can efficiently instantiate thousands of stones for creating complex architectural models. The volumetric model representing the walls and roof was sketched using our implementation of the Arches system [PGMG09] in less than two minutes.

n	Stones	Time
5	7355	23
6	17166	73
7	33217	176

Table 1: Timings (in seconds) for generating tiles

Timings Our rock pile generation technique can create thousands of rocks piled together very efficiently. Table 1 reports timings for generating the set of aperiodic tiles as well as the number of cells n . The Voronoï volume computation and the erosion processes were performed using a finer grid \mathcal{G} , with $n_{\mathcal{G}} = 10 \times n$ so that the resolution should be fine enough to capture the details of smaller stones.

Note that the creation of the corner cube grid with seed points is negligible: most of the time is spent in the Voronoï volume generation, the erosion process and in the creation of the meshes of the rocks from their implicit representation.



Figure 24: Bories (ancient dry stone huts) generated with our rock tiling technique (28032 stones)

Recall that the geometry of rocks in the cubic tiles is generated once and for all as a preprocessing step. In contrast, instantiating the rocks in a scene can be performed very efficiently.

8. Conclusion

We have proposed an original aperiodic tiling technique for generating piles of thousands of rocks. Our method not only avoids repetitive patterns that occur when using periodic tiling, but also provides simple control over both the shape of rocks and the shape of rock piles. This enables us to create a vast variety of models (including gravels or pebbles) and improve the overall realism of natural sceneries.

Acknowledgements

This work was supported by Agence Nationale de la Recherche as ANR-06-MDCA-004-01 project. We wish to credit Eden Games and Widescreen Games for participating to the project. Special thanks go to Pascal Bouvier and Robert Foriel (Widescreen Games) for developing the real time HLSL and Mental Ray shaders, and to Etienne Duranton (LIRIS) for his assistance in the implementation into the framework.

References

- [BFO*07] BEARDALL M., FARLEY M., OUDERKIRK D., SMITH J., JONES M., EGBERT P.: Goblins by spheroidal weathering. In *Eurographics Workshop on Natural Phenomena* (2007), pp. 7–14. 4, 6
- [BPB09] BOULANGER K., PATTANAİK S. N., BOUATOUCH K.: Rendering grass in real time with dynamic lighting. *IEEE Computer Graphics and Applications* 29, 1 (2009), 32–41. 2
- [CBS96] CRESPIAN B., BLANC C., SCHLICK C.: Implicit sweep objects. *Computer Graphics Forum* 15, 3 (1996), 165–174. 5
- [CSHD03] COHEN M., SHADE J., HILLER S., DEUSSEN O.: Wang tiles for image and texture generation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)* 22, 3 (2003), 289–294. 2, 3

- [KMN88] KELLEY A. D., MALIN M. C., NIELSON G. M.: Terrain simulation using a model of stream erosion. In *Proceedings of ACM SIGGRAPH* (1988), pp. 263–268. 2
- [LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of ACM SIGGRAPH* (1987), pp. 163–169. 6, 7
- [LD06a] LAGAE A., DUTRÉ P.: Poisson sphere distributions. In *Vision, Modeling, and Visualization* (2006), pp. 373–379. 2, 3
- [LD06b] LAGAE A., DUTRÉ P.: An alternative for Wang tiles: Colored edges versus colored corners. *ACM Transactions on Graphics* (2006), 1442–1459. 3
- [LDG01] LEGAKIS J., DORSEY J., GORTLER S.: Feature-based cellular texturing for architectural models. In *Proceedings of ACM SIGGRAPH* (2001), ACM Press, pp. 309–316. 2
- [LEQ*07] LU A., EBERT D. S., QIAO W., KRAUS M., MORA B.: Volume illustration using wang cubes. *ACM Transactions on Graphics*. 26, 2 (2007), 11. 2
- [MDH07] MEI X., DECAUDIN P., HU B.: Fast hydraulic erosion simulation and visualization on GPU. In *Proceedings of Pacific Graphics* (2007), pp. 47–56. 2
- [MIS01] MIYATA K., ITOH T., SHIMADA K.: A method for generating pavement textures using the square packing technique. *The Visual Computer* 17, 8 (2001), 475–490. 2
- [Miy90] MIYATA K.: A method of generating stone wall patterns. In *Proceedings of ACM SIGGRAPH* (1990), pp. 387–394. 2
- [MKM89] MUSGRAVE F. K., KOLB C. E., MACE R. S.: The synthesis and rendering of eroded fractal terrains. In *Proceedings of ACM SIGGRAPH* (1989), pp. 41–50. 1
- [NWD05] NEIDHOLD B., WACKER M., DEUSSEN O.: Interactive physically based fluid and erosion simulation. In *Eurographics Workshop on Natural Phenomena* (2005), pp. 25–32. 2
- [PGMG09] PEYTAIVIE A., GALIN E., MERILLOU S., GROSJEAN J.: Arches: a Framework for Modeling Complex Terrains. *Computer Graphics Forum* 28 (2009), 457–467. 2, 8
- [SKP05] SHAH M., KONTINEN J., PATTANAİK S.: Real-time rendering of realistic-looking grass. In *Proceedings of GRAPHITE* (2005), pp. 77–82. 2
- [ST89] SZELISKI R., TERZOPOULOS D.: From splines to fractals. In *Proceedings of ACM SIGGRAPH* (1989), pp. 51–60. 1
- [ZSTR07] ZHOU H., SUN J., TURK G., REHG J. M.: Terrain synthesis from digital elevation models. *IEEE Transactions on Visualization and Computer Graphics* 13, 4 (2007), 834–848. 2