Computer Simulation and Generation of Moving Sand Pictures

Mohan Zhang, Hongwei Lin, Kang Zhang and Jinhui Yu

Abstract—Moving sand pictures are interesting devices that can be used to generate an infinite number of unique scenes when repeatedly being flipped over. However, little work has been done on attempting to simulate the process of picture formulation. In this paper, we present an approach capable of generating images in the style of moving sand pictures. Our system defines moving sand pictures in a few steps, such as initialization, segmentation and physical simulation, so that a variety of moving sand pictures including mountain ridges, desert, clouds and even regular patterns can be generated by either automatic or semi-automatic via interaction during initialization and segmentation. Potential applications of our approach range from advertisements, posters, post cards, packaging, to digital arts.

Index Terms—Moving sand picture, tessellation, segmentation, color arrangement, physical simulation.

1 Introduction

Moving sand pictures are made by filling several kinds of silicon carbides (sand grains) with specific weights and colors, and a small amount of water between the two pieces of glass. Although the moving sand picture device is simple, when flipped over repeatedly, it can be used to generate an unlimited number of beautiful images of mountains, rivers, and deserts, etc, as shown by two examples in Fig. 1 (a) and (b).

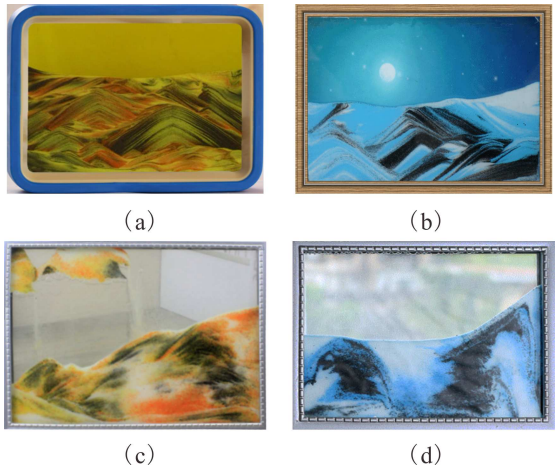


Fig. 1: *Real life moving sand pictures: satisfactory results (a) and (b), and unsatisfactory results (c) and (d).*

Real life moving sand pictures are attractive due to their interactive and visual appealing. However, the random properties involved in the interactions between sand grains and air bubbles formed by the water inside the devices make the resulting pictures unpredictable, as shown in Fig. 1 (c) and (d). One usually has to flip over the device many times to obtain a beautiful picture. Computer simulation of moving sand pictures could, however, support various controls over the generated results and offer almost unlimited room for creativity and imagination. For example, one could easily change sand colors and/or granularities, insert virtual obstacles, and possibly extend to three dimensions. Research in this direction could potentially generate a new type of visual communication media for a wide range of applications in design industry, such as card, desktop, packaging and textile pattern design.

In this study, we develop an approach that is able to generate moving sand pictures, ranging from mountain ridges, clouds, desert, to regular patterns. The main contributions of this study include: (1). Two means (halftoning and rectangle tessellation) to initialize the upper part of the drawing window; (2). Tools for users to effectively control the resulting pictures to suit various application needs; (3). Accelerated simulation with three layers of sand grains (dynamic, static and fixed layers).

- *M. Zhang is with the State Key Lab. of CAD&CG, Zhejiang University, Hangzhou, 310058, China.*
- *H. Lin is with the Department of Mathematics, and State Key Lab. of CAD&CG, Zhejiang University, Hangzhou, 310027, China.*
- *K. Zhang is with the Department of Computer Science, The University of Texas at Dallas, Richardson, TX 75080-3021, USA.*
- *J. Yu is with the State Key Lab. of CAD&CG, Zhejiang University, Hangzhou, 310058, China. He is also a guest professor at the Department of Computer Science, Harbin Finance University, Harbin, 150030, China.*

2 Related Work

Moving sand pictures are different from traditional arts because no strokes are involved in making pictures, the result of a moving sand picture is very much like non-photorealistic rendering, while its forming process is purely physical as a result of sand-bubble-liquid interaction. It is these similarities that inspired the proposed methodology, that contains both physical simulation of sand collision and falling, and artistic control over the layout and texture of sand pile shapes.

In this section we just review some works upon that our work draws, and omit the work fallen in non-photorealistic rendering where the traditional media simulation is the main concern, notably the choices of stroke placement and the mixing of paint. In addition, moving sand pictures show

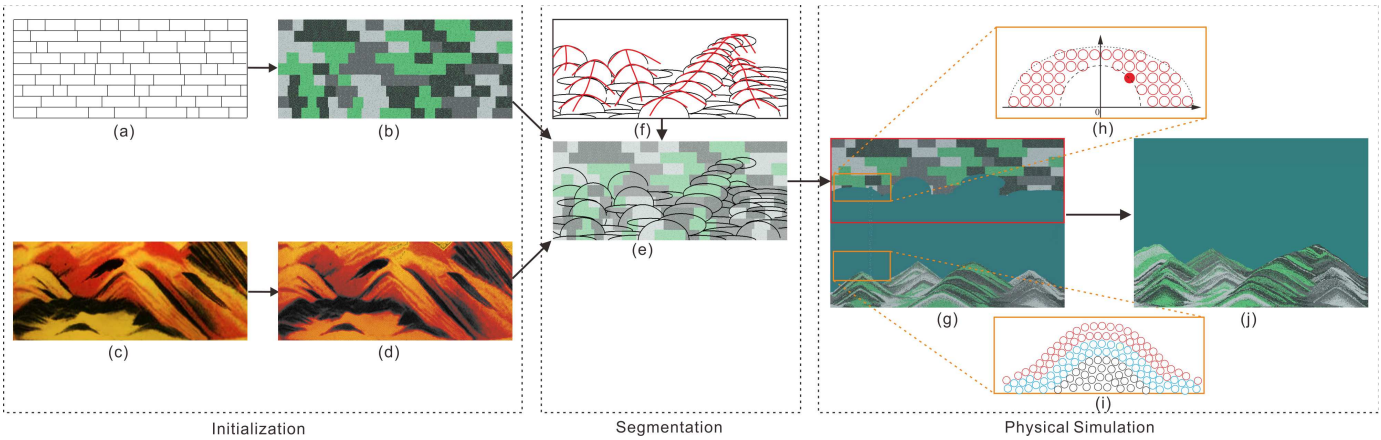


Fig. 2: *Overview of our system. (a) Tessellation of initialization region. (b) Filling tessellation cells with sand grains. (c) Input the moving sand image. (d) Halftoning. (e) Segmentation of initialization region. (f) Input the sketch (optional). (g) Simulation of sand grains falling and piling. (h) Determination of sand grains falling order. (i) Three layer model. (j) Result.*

surprisingly complex behaviors since they involve complicated interactions among sand grains, water and bubbles which are hard to simulate accurately by physical means.

A related work, which is relatively simpler than moving sand picture simulation, granular simulation has been studied extensively in geophysics and physics to engineering analysis and computer graphics. In the following we mainly comment on the most representative works from these communities. Existing works related to the granular materials can largely be divided as the following.

For visual applications, continuum methods which simulate granular materials as fluid can give fairly convincing results. The Smoothed Particle Hydrodynamics (SPH) method was proposed originally to model fluid dynamics in astrophysics [1]. Zhu and Bridson [2] modeled granular materials as incompressible fluid with the fluid-implicit-particle algorithm, and their approach relied on identifying rigidly moving regions of material to obtain stable piles. Lenaerts and Dutré [3] unified SPH framework where fluids and granular materials are two-way coupled. Improving the method of [2], Narain et al [4] developed a new fluid solver combining the strengths of both particles and grids with enhanced flexibility and efficiency. In addition, Wilkinson and Willemsen [5] developed the Invasion Percolation (IP) method to model complex fluids. Hawick and Ken [6] adapted the IP model to experiment with flow of immiscible fluids in model reservoir systems. A versatile and robust SPH simulation approach for multiple-fluid flow was proposed [7], and later extended to cover solid phases, including deformable bodies and granular materials [8].

Natural methods have also been proposed for animating granular materials that directly simulate the interactions between individual grains. Luciani et al [9] developed a particle system model for granular materials using damped nonlinear springs. An efficient discrete element method (DEM) was proposed to capture granular phenomena in [10], and Bell et al [11] exhibited a two-way coupling between granular materials and rigid bodies by using DEM. Alduán et al [12] used an adaptive resolution version of the method [11] to improve performance. Furthermore, DEM and smoothed particle hydrodynamics were utilized to simulate the intersection of fluids and granular materials [13]. Recently, Macklin et al [14] presented a unified dynamics framework for real-time visual effects that can be applied by casting granular interactions as hard constraints.

Combining aspects of particle-based and grid-based methods, Sulsky et al [15] developed the Material Point Method that uses two representations of the continuum, one based on a collection of material points and the other on a computational grid. It has been used in modeling granular materials, such as snow [16] and sand [17]. Recently, Gergely et al. [18] used the material point method to discretize the governing equations for its natural treatment of contact, topological change and history dependent constitutive relations. Built upon this method, Gilles et al. [19] derived a compact model for the rheology of the material.

Other works in granular materials focus on improving computational efficiency. Granular materials were modeled using height fields in [20] and [21]. Sumner et al [22] took a similar height fields approach with simple displacement and erosion rules to model footprints, track, etc. Pla et al [23] applied cellular automata to model granular terrains interactively. In the work by Bouchaud et al [24] two populations of grains, immobile and rolling, are recognized so that only rolling grains need to be calculated. The model proposed by Bouchaud et al [24] was later used for improving the efficiency of the granular materials [25], [26] and [27].

In contrast to granular simulation, only a few attempts have been made on moving sand picture simulation. The first attempt was by Pearce et al [28] who constructed a lattice-based simulation of a sand picture based around the Kawasaki spin-exchange model [29] and IP model [6] with empirical couplings between cells. The generated pictures however visually differ greatly from those in Fig. 1. In this study, we develop an approach for generating moving sand pictures, which are visually similar to those in Fig. 1, and also aesthetically pleasing.

3 Approach Overview

The overall architecture of our system is presented in Fig. 2. First, at the initialization stage, we arrange sand grains with different attributes, such as weight, size, and color in

the working window. We offer users two options at this stage. One is to take a moving sand picture or a photograph as input (Fig. 2(c)) and then apply halftoning on the input image (Fig. 2(d)). The other is to tessellate the entire region with small rectangles (called *cells*) of varying sizes (Fig. 2(a)) and fill the cells with sand grains of different colors (Fig. 2(b)). Users may further edit colors in the cells for personal preferences.

Next, at the segmentation stage, we segment the entire region using ellipses of varying sizes. Fig. 2(e) shows the result of segmenting the image in Fig. 2(d). Adoption of ellipses aims at simulating holes where sand grains fall through, and the ellipses eventually correspond to mountains in the resulting pictures. Thus, in order to offer more control over the composition of mountains, users may optionally input a sketch (red lines in Fig. 2(f)) to guide the placement of ellipses (black lines in Fig. 2(f)) for the desired composition .

Finally, at the physical simulation stage, we set the falling order of sand grains in each segmented cell in Fig. 2(e). In each cell, our approach sorts sand grains in a fixed order of falling (Fig. 2(h)) and calculates each sand grain by Physx library [30](Fig. 2(g)). We use a three-layer model to speed up the simulation process (Fig. 2(i)). When all sand grains fall down the final moving sand picture (Fig. 2(j)) is obtained.

4 Initialization of Sand Grains

Since the gap between the enclosing glass plates in the moving sand picture device is usually small, it is sufficient for us to simulate moving sand pictures on a two dimensional space [28]. The first step of simulation is to arrange sand grains of different attributes such as weight, size and color in the so-called initialization region over the upper part of the simulation window. This section describes the working window, the initialization region in the window, and attributes of sand grains (Section 4.1). We offer two initialization strategies, i.e. halftoning (Sections 4.2) and tessellation (Section 4.3), and then describe how to fill the colored sand grains in the tessellated region (Section 4.4).

4.1 Working window and attributes of sand grains

We set the size of the working window at 1000×700 pixels, which is enough to produce aesthetically pleasing moving sand pictures. The initialization region is a sub-window in the upper part of the working window, thus of the same width as the working window, but varying height, subject to the user's aesthetic preference. The default height of the initialization window is set at 350 pixels.

To simulate sand grains of different weights in moving sand pictures, we employ four classes of sand grains, as suggested in [31], and denote them by S_1, S_2, S_3, and S_4, respectively. Inspired by [31] and [32], we assign the following densities, $3.2g/cm^3$, $3.2g/cm^3$, $2.7g/cm^3$, and $2.7g/cm^3$, and diameters, $0.2mm$, $0.2mm$, $0.1mm$, and $0.1mm$ to S_{1-4} to model the sand grains as small spheres. The number of sand grains corresponding to S_{1-4} is set at $3/8$, $3/8$, $1/8$, and $1/8$ of the total sand grains, as suggested in [31]. The colors of sand grains corresponding to S_{1-4}, denoted by C_{1-4}, can be specified by the user.

4.2 Initialization by halftoning

One strategy of initialization is to take a moving sand picture or a photograph with the same size of the initialization region as input. Since pixel values in the input image are usually much larger than the 4 color options used in our system, we first convert the input image (Fig. 3(a)) into gray scale one (Fig. 3(b)), and then perform halftoning on the gray scale image to obtain an image with pixel values of four classes (Fig. 3(c)), and assign corresponding attributes, such as weights, sizes, and colors, to the halftone image (Fig. 3(c)), finally obtain color halftone image as in Fig. 3(d).

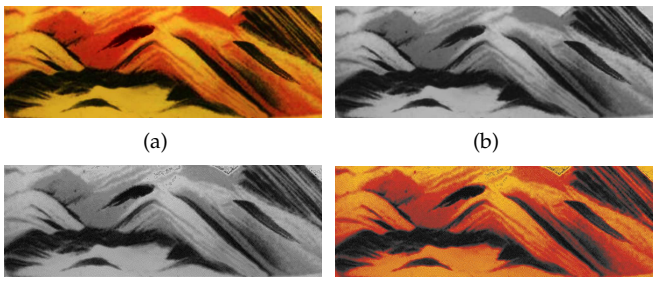


Fig. 3: *Initialization by halftoning. (a) Input an image. (b) Convert the input image into gray scale. (c) Halftoning the gray scale image. (d) Colored halftoning image.*

Halftoning has been an active research area for many years, and thus several halfoning algorithms are available, such as Floyd-Steinberg error diffusion [33], ordered dithering [34], and Knuth's dot-diffusion [35], etc. We employ the Floyd-Steinberg error diffusion algorithm that achieves dithering using error diffusion. The algorithm pushes the residual quantization error of a pixel onto its neighboring pixels, to be handled later. Specifically, it spreads the residual quantization errors out using the following distribution mask:

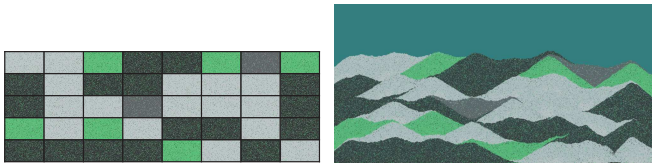
$$\begin{pmatrix} 0 & P & \frac{7}{16} \\ \frac{3}{16} & \frac{5}{16} & \frac{1}{16} \end{pmatrix}$$

where P represents the pixel currently being scanned, and the numbers represent the portion of the error that is distributed to the pixel in the specified position. The algorithm scans the image top down and left to right, quantizing pixel values one by one. When the quantization error is transferred to the neighboring pixels, the pixels that have already been quantized are not affected. For more details on the Floyd-Steinberg algorithm, please refer to [33].

When the moving sand pictures are not available, our approach offers an alternative solution to initialize sand grains. Unlike initializing sand grains by halftoning on an input image, where the distribution of sand grains is provided by the input image itself, our new method of sand grains initialization must solve the following two problems: (1) tessellating the initialization region with small cells and (2) filling the cells with sand grains of different attributes. We propose two simple strategies for tessellating and filling sand grains to generate scenes similar to those seen in the moving sand picture device, described in the following two sections.

4.3 Tessellation of initialization region

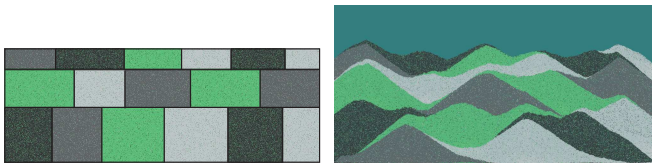
The first issue associated with tessellation is the choice of cell shapes. Tessellating the initialization region with cells of irregular shapes is extremely difficult because it requires that the user manually draws a picture at the quality of actual moving sand pictures. Because the initialization region is a large rectangle occupying the upper part of the working window, it is natural to choose small rectangles as tessellation cells, as shown by the black lines in Fig. 4(a). After all cells are filled with different kinds of sand grains (sand grains filling strategy will be described in Section 4.4), sand grains from the cells fall downward to form a scene which looks like mountain ranges (the sand grains falling procedure will be described in Sections 6 and 7), as shown in Fig. 4(b). However, the ranges appear unnatural and even mechanical.



(a) (b)

Fig. 4: *Tessellation of initialization region by uniform rectangles (a) and corresponding simulation result (b).*

Since sand grains fall down cell by cell in our approach, sand grains falling from their corresponding cell pile up at the bottom part of the working window to form a sandpile. Obviously, the larger the cell is, the larger the corresponding sandpile. It can then be expected that more natural mountain ranges could be obtained with varied sizes of rectangular cells in the tessellation region. For instance, because distant mountains may appear lower than the mountains nearby, we tessellate the initialization region using rectangles with heights that increase progressively from the top to the bottom. On each row, widths vary randomly, as illustrated by the black lines in Fig. 5(a). The corresponding generated result is given in Fig. 5(b), where mountain sizes change more naturally.



(a) (b)

Fig. 5: *Tessellation of initialization region by rectangles of varying sizes (a) and corresponding simulation result (b).*

4.4 Filling tessellation cells with sand grains of different colors

After the initialization region is tessellated, all tessellation cells must be filled with the four classes of sand grains, S_{1-4}, rendered with the corresponding four colors C_{1-4}. Fig. 1 shows that in the moving sand picture device the four classes of sand grains are distributed in such a complex way that different patterns and shapes can be formed. Thus, simply filling all rectangular cells with uniformly mixed S_{1-4} sand grains is not a good idea, because the resulting scene may be generated without visible patterns. Therefore, it is necessary to fill tessellation cells with mixed sand grains, S_{1-4}, yet with notably dominant colors, from cell to cell so that visible patterns can be generated after sand grains move down from the cells.

We propose the following strategy to fill S_{1-4} in tessellation cells. A cell is filled in such a way that it appears to have one color dominant over other three colors, and the ratio between sand grains with dominant color and sand grains of the other three colors is set empirically to 0.8:0.2 in one cell. The four classes of sand grains S_{1-4} are set to $\frac{3}{8}$, $\frac{3}{8}$, $\frac{1}{8}$, and $\frac{1}{8}$ of the total amount of sand grains, respectively (Section 4.1). The algorithm that determines the color of a tessellation cell's sand grains is given in the following pseudo code:

Algorithm 1 Fill tessellation cells

```
for each cell in the tessellation region do
    Select S_m ∈ S_{1-4} with probabilities 3/8, 3/8, 1/8, 1/8;
    Choose C_m as the dominant color;
    Select S_n ∈ ∁_{S_{1-4}} S_m with probabilities 1/3, 1/3, 1/3;
    choose C_n as the non-dominant color;
    for each sand grain s in the rectangle cell do
        if a random number R ∈ [0, 1] ≤ 0.8 then
            Assign s with color C_m;
        else
            Assign s with color C_n;
        end if
    end for
end for
```

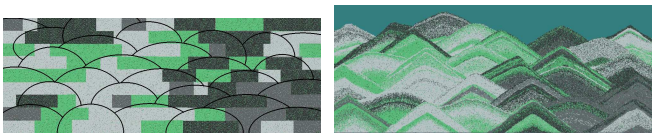
5 Segmentation of Tessellated Region

In a physical moving sand picture device, the falling process of sand grains is complex due to interactions among sand grains, bubbles and water. Accurate simulation of this process is almost impossible. Sand grains falling directly from rectangular tessellation cells one by one would be a simple strategy to simulate the process, and two simulated results shown in Fig. 4(b) and Fig. 5(b) are not satisfactory, due to the lack of strip patterns seen in Fig. 1. It is necessary then to re-segment the tessellated region with different geometric primitives so that strip patterns can be produced over mountain shapes.

5.1 Segmentation with varying ellipses

We adopt ellipses to approximate shapes formed by sand grains falling downward and use them to re-segment the tessellated region. Most of ellipses are bigger in area than those of rectangular cells, thus one ellipse may overlap with several rectangular cells, as shown in Fig. 6(a). We take each ellipse as the unit and let sand grains located in the ellipse unit fall, sand grains with different dominant colors in the small tessellation cells covered by an ellipse produce strip patterns over the mountain shapes formed by over all sand grains in the ellipse. We continue this procedss until all

ellipses are visited, the resulting picture shown in Fig. 6(b) looks similar in style to those in Fig. 1.



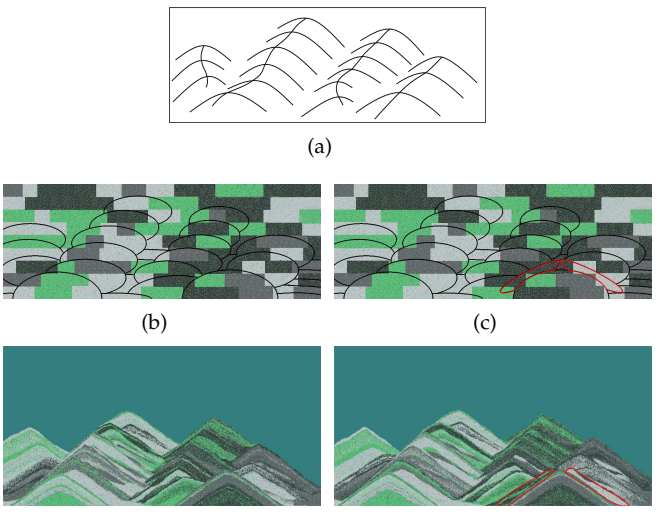
(a) (b)

Fig. 6: *Segmentation of the tessellated region with varying ellipses (a) and corresponding simulation result (b).*

When halftoning is used for initialization (Section 4.2), tessellating the initialization region with varying rectangles is unnecessary because the distribution of different sand grains is irregular in the input image. We therefore skip the tessellation step and perform segmentation with ellipses directly on the halftoned input image.

5.2 Sketch based segmentation

Our goal is to reproduce moving sand pictures similar to those in Fig. 6(b), and also to generate moving sand pictures which are aesthetically pleasing. It would be desirable then to arrange mountains so that they can show mountain ridges often seen in a natural scene. To this end, we design a tool in our UI that allows users to draw a sketch as in Fig. 7(a), where long lines suggest mountain ridges, thus are called as ridge lines, while arc-like curves along the ridge lines suggest mountain profiles, they also depict how close mountains are to each other in depth. Next, we detail how to place ellipses of varying sizes according to the sketch.



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

Fig. 7: *Segmentation guided by the sketch. (a) The sketch. (b) Segmentation without long and narrow ellipses. (d) Corresponding simulation result. (c) Segmentation with long and narrow ellipses. (e) Corresponding simulation result.*

Detection of intersections in the sketch. Intuitively, an ellipse should be placed under an arc-like curve crossing the ridge lines so that a mountain like pattern could be produced in the scene. It is necessary then to detect the intersections between ridge lines and arc-like curves first in the sketch, this can be done with traditional line intersection detection algorithm, and intersection points detected are put into $\Psi_P\{P_1, P_2, ...P_n\}$, where n is the total number of intersection points detected.

Placement of ellipses. To place an ellipse under the ith arc-like curve, we need to determine the center, major and minor axes of the ellipse. Also, considering that nearby mountains appear larger than those further away, we should place ellipses such that their sizes decreasing progressively bottom up. In our implementation, we set the semi-major axis of the ellipse by $a_i = 0.5 * A_m - \alpha P_i(y)$, where A_m is the maximum major axis estimated by dividing the width of the initialized region by the number of ridge lines near the bottom of the sketch, $P_i(y)$ is the y coordinate of the ith detected intersection point, $i = 1, 2, 3, ...n$, and α is a parameter set at 0.3. We let the eccentricity of ellipses vary in the range [0.45,0.65], with which the semi-minor axis b_i for all ellipses to be used in segmentation can be obtained.

The center $(O_i(x), O_i(y))$ of the ith ellpise placed under the detected intersection point $(P_i(x), P_i(y))$ can be obtained by $O_i(x) = P_i(x)$ and $O_i(y) = P_i(y) - b_i$, as illustrated by the lower ellipse in Fig. 8.

Sketch-guided elliptical segmentation. With all attributes obtained for n ellipses, we can place them in the initialization region for segmentation, as illustrated by two black ellipses centered at O_i and O_j in Fig. 8. Via experiments we find that, when the distance between centers of two intersection points along one ridge line is larger than 1.2 times of the semi-minor axis of the ellipse to be placed, the sand pile shapes generated are away from the ridge line in the sketch, as shown by Fig. 7(b) and Fig. 7(d). In order to preserve the ridge line in the generated sand pile shapes, the long and narrow ellipses should be inserted on the left and right sides on the top of the lower ellipse, so that sand grains inside these two ellipses may produce strip patterns beside the sand pile shape produced by sand grains in the lower ellipse.

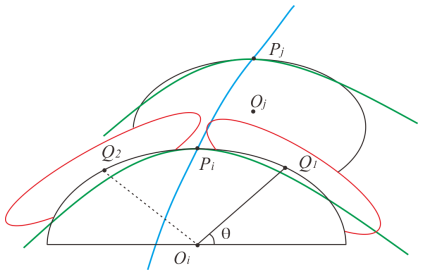


Fig. 8: *Blue line indicates the mountain ridge, green lines indicate arc-like curves in the sketch, and red ellipses indicate inserted ellipses.*

The two inserted ellipses are highlighted in red in Fig. 8, we determine their centers Q_1 and Q_2 on the lower ellipse simply by letting θ varying in the predefined ranges, their orientations are the same as the tangent at Q_1 and Q_2 on the lower ellipse, the semi-major axes of two inserted ellipses are set at 0.6 times of the semi-major axis of the lower ellipse, and their eccentricities are set in the range [0.96,0.98]. With the elliptical segmentation modified by inserting two long and narrow ellipses (red ones in Fig. 7(c)), the ridge line

in the sketch can be preserved in the generated sand pile shapes as expected (Fig. 7(e)).

6 Determination of Sand Grains Falling Order

After the tessellated region is segmented by a set of ellipse cells, we sort the segmentation cells by row order according to the y-coordinates of their centers and then place the segmentation cells on each row randomly, until all cells in the row have been visited.

For each cell, we must determine the falling order of sand grains in the cell. There are many possible orders for thousands of sand grains to fall. We thus observe how sand grains fall in a physical moving sand picture device and choose a method that is able to mimic the natural falling order.

When the moving sand picture device is flipped over, the air bubbles formed by water rise from the bottom. When rising up, large air bubbles begin to break into small ones. Due to the gravity of sand grains above large air bubbles, small gaps begin to appear between air bubbles, and sand grains above air bubbles begin to fall through the gaps. An accurate physical simulation of the air bubble's rising and breaking is beyond the scope of this paper. Nevertheless, this behavior inspires us to propose a simple strategy to set the order of sand grains falling as follows.

In each segmentation cell that is selected, we first take the center of the bottom line of the cell as a reference, and then we sort sand grains in the cell in ascending order of the distances between each sand grain and this center. For sand grains with equal distance between their static positions and the center, they are randomly sorted one by one until they are all visited. Fig. 10 shows a half circle with a radius d equal to the distance between a sand grain and the center of the bottom line of the segmentation cell.

7 Simulation of Sand Grains Falling and Piling

Our approach simulates the process of sand-piling physically. All sand grains fall as solid bodies toward the ground due to gravity. When a collision between sand grains occurs while falling, we consider it an elastic collision or inelastic collision depending on the weights assigned to the sand grains. We simulate the falling, collision, and piling of sand grains using Physx [30].

In Physx, all physical objects have at least one material, which defines the friction and restitution properties used to resolve a collision with the objects. Friction uses the Coulomb Friction Model, which involves 2 coefficients, i.e., *static friction coefficient* and *dynamic friction coefficient* (sometimes called *kinetic friction*). Friction resists relative lateral motion of two solid surfaces in contact. These two coefficients define a relationship between the normal force exerted by each surface on other surfaces and the amount of friction force applied to resist lateral motion. Static friction defines the amount of friction applied between non-moving surfaces. Dynamic friction defines the amount of friction applied between surfaces that are moving relative to each-other.

The coefficient of restitution of two colliding objects is a fractional value representing the ratio of speeds after and before an impact, taken along the line of impact. A coefficient of restitution of 1 is said to collide elastically, while a coefficient of restitution < 1 is said to be inelastic.

In our implementation, we set the parameters of dynamic friction, static friction, and restitution to $0.4, 0.4$, and 0.1, respectively, based on our experiments. Changing the friction coefficient will affect the final look of generated sandpile shapes, as demonstrated in Fig. 9, sandpile shapes vary progressively from flatter to steeper when values of friction coefficients increase. Moreover, the size of each grain also affects the simulation, smaller sand grains may result in flatter sand piles and bigger sand grains may result in steeper sand piles.

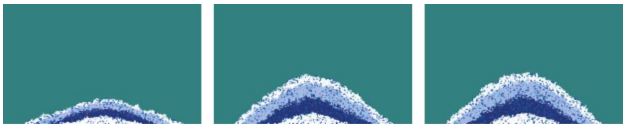


Fig. 9: *Sand pile shapes generated with dynamic friction and static friction set to* 0.1 *in (a),* 0.4 *in (b), and* 0.9 *in (c). The restitution remains to be* 0.1 *in (a), (b) and (c).*

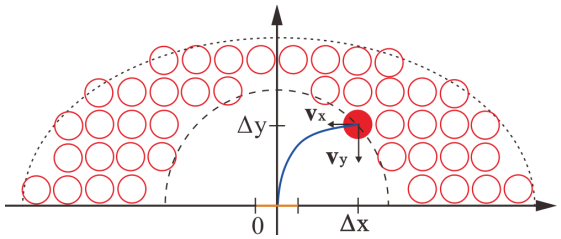


Fig. 10: *Moving a sand grain from its current position to the outlet in a segmentation cell.*

Fig. 10 shows a segmentation cell with a dashed line ellipse laid on a local coordinate system with the origin in the middle of the bottom line in the cell. To simulate a small gap between air bubbles in the moving sand picture device, we put an outlet in the middle of the bottom line in the cell (orange line in Fig. 10), and set its width at 15 mm to ensure that all four classes of sand grains are able to pass through the outlet. We sort each sand grain in a cell according to the distance d_i (i=1,...n, where n is the number of sand grains contained in a cell) between its static position and the origin of the coordinate system and, from the minimum to the maximum values among d_i, we pick up a sand grain s_i (red solid circle in Fig. 10) according to its d_i, and assign a horizontal velocity $\boldsymbol{v_x}$, which starts at s_i and points to the Y axis (For those sand grains having the same value in d_i, we randomly pick up one until they all are picked up). For s_i to fall out of the outlet from its static position, v_x should satisfy the following equations:

$$\begin{cases} \Delta x = v_x t \\ \Delta y = g t^2/2 \end{cases} \tag{1}$$

where Δx and Δy are coordinates of s_i in the local coordinate system, t is time, and g is the acceleration due to gravity. Using equations in Eq. 1, we can solve v_x and find:

$$v_x = \Delta x \sqrt{\frac{g}{2\Delta y}} \tag{2}$$

The vertical velocity v_y, pointing downwards, is assigned to s_i with $v_y = gt$, where t starts when s_i begins to move. As t increases, the sand grain s moves to the outlet along the trajectory indicated by the blue line in Fig. 10.

8 Acceleration of Simulation

In Physx, the simulation of sand grain's falling, colliding, and piling up must take all sand grains in the scene into account. Thus, as more sand grains come into the scene, the time involved in the simulation calculations increases dramatically. To reduce the computational cost, the BCRE model was proposed in [24]. The model treats the sand pile as two layers, the *dynamic layer* and the *static layer* (described later on). The visible and rolling part of the sand flow is simulated using the discrete element method and the invisible and static layer is represented by a height field.

Because the number of sand grains used in our system is about $3 \sim 5$ times that used in the BCRE model [27], it may take more than 2 hours to complete the moving sand picture simulation even when the two layer BCRE model is used (see statistics in Table. 1 in Section 10). As such, we desire to reduce the simulation time. In this section, we introduce an additional layer, the *fixed layer*, to the BCRE model in order to accelerate the simulation process:

- *dynamic layer*: consists of sand grains that can move and are fully controlled by Physx, as indicated by the red circles in Fig. 11;
- *static layer*: consist of sand grains that are almost immobile due to their speeds and are also fully controlled by Physx, shown by the blue circles in Fig. 11;
- *fixed layer*: consists of sand grains that are immobile because they are under the static layer, shown by the black circles in Fig. 11.

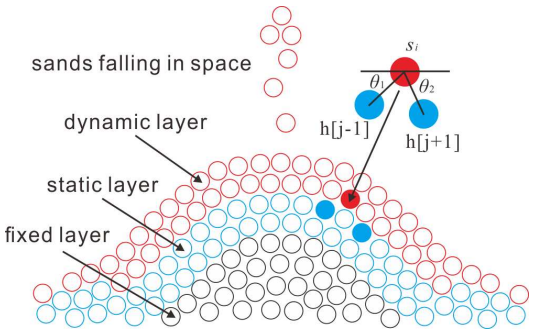


Fig. 11: *A sand pile subdivided into three layers.*

8.1 Classification of three layers

In order to classify the piled sand grains into three layers, we first map the four classes of sand grains S_{1-4} from the scene, handled in Physx, to the pixels in the working window. We then use an array $H_s[j]$, $(j = 1, ..., n)$, initialized with zeros, to record the maximum height of the static layer, where j corresponds to the jth pixel in the working window, and n is the width of the window. In our implementation, n is also equal to the width of the scene handled in Physx.

We devise several simple rules to determine whether a specific sand grain on the sand pile belongs to the dynamic, static, or fixed layer. A sand grain is in the dynamic layer when it is moving in space (falling). When it falls on the sand pile, it is added to a set Ψ_d.

In Physx, the time step is controlled by the parameter *elapsed time*. In our implementation we adopt the default setting provided in Physx, 0.016s, which is adequate for both simulation and rendering in our application. We realize that updating all data used in the three layers at every time step slows down the simulation significantly. Based on our experiments, we take 40 Physx time steps as a *simulation cycle*, and update all the data in the three layers a the end of each cycle.

After Ψ_d is filled with sand grains that have fallen on the sand pile during one simulation cycle, we traverse sand grains in Ψ_d. For a sand grain s_i, $(i = 1, ..., m_d$, where m_d is the number of sand grains in $\Psi_d)$, at position (x, y) in Ψ_d (shown by a lower red solid circle, enlarged in the upper right of Fig. 11), we first truncate x with $\lfloor x + 0.5 \rfloor$ to obtain an index j, and then pick two sand grains, recorded in $H_s[j + 1]$, and $H_s[j - 1]$ (shown by the two blue solid circles near the lower red solid circle, enlarged in the upper right of Fig. 11) to calculate an angle defined by $\theta(x, y) = 0.5 * (\theta_1 + \theta_2)$, where θ_1 and θ_2 are calculated by $\theta_1 = \arctan(H_s[j + 1] - y)/((j + 1) - x)$ and $\theta_2 = \arctan(y - H_s[j - 1])/(x - (j - 1))$, respectively, as illustrated by the two angles shown in Fig. 11. Here $\theta(x, y)$ approximately describes the slope of the static layer at $H_s[j]$. By setting a parameter α to confine $\theta(x, y)$ it is possible to control the slope defined by $H_s[j]$. In our implementation we set α at $\pi/6$ by default. Users can however tune it to obtain desired effects.

If a sand grain s satisfies following two conditions:

- its velocity is below $3 \times 10^{-5} m/s$
- $\theta(x, y)$ is less than α

it should be in the static layer, and thus moved from Ψ_d to a set Ψ_s. Furthermore, if the height of s_i is larger than the current height $H_s[j]$ of the static layer, i.e, $y > H_s[j]$, we update $H_s[j]$ with y.

We use another array $H_f[j]$, which is initialized to store zeros, to record the maximum height of the fixed layer. Since the fixed layer is below the static layer, we need to first set a suitable static layer thickness before $H_f[j]$ is updated. Through experiments, we find that a thick static layer would consume consierable time in the Physx simulation because it would involve a large number of static sand grains in the computation. A too thin static layer would, however, cause sand grains from the dynamic layer go through small gaps too fast. We therefore set the thickness of the static layer to be $2.0mm$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2017.2779799, IEEE Transactions on Visualization and Computer Graphics

For a sand grain s_i, ($i = 1, ..., m_s$, where m_s is the number of sand grains in Ψ_s), at position (x, y) in Ψ_s, we truncate x with $\lfloor x + 0.5 \rfloor$ to obtain an index j and. If the following two conditions are satisfied:

- $H_s[j] - H_f[j] > 2.0mm$
- $y < H_s[j] - 2.0mm$

then the sand grain s_i is in the fixed layer, and moved from Ψ_s to Ψ_f. We update $H_f[j]$ to y if $y > H_f[j]$.

We repeat the above procedure to update Ψ_d, Ψ_s, Ψ_f, $H_s[j]$ and $H_f[j]$ at the end of each simulation cycle in Physx until all the sand grains in the initialized region fall down. The above algorithm is outlined in the pseudo code below (Algorithm 2).

Algorithm 2 Three-layer model

for all sand grains $s_i(x, y) \in \Psi_d$ **do**
  index $j \Leftarrow \lfloor x + 0.5 \rfloor$;
  angle $\theta_1 \Leftarrow \arctan(H_s[j+1] - y)/((j+1) - x)$;
  $\theta_2 \Leftarrow \arctan(y - H_s[j-1])/(x - (j-1))$;
  slope $\theta(x, y) \Leftarrow 0.5 * (\theta_1 + \theta_2)$;
  if its velocity $v_s < 3 \times 10^{-5} m/s$ & $\theta(x, y) < \alpha$ **then**
    move s_i from Ψ_d to Ψ_s;
    if $y > H_s[j]$ **then**
      update $H_s[j] \Leftarrow y$;
    end if
  end if
end for
for all sand grains $s_i(x, y) \in \Psi_s$ **do**
  index $j \Leftarrow \lfloor x + 0.5 \rfloor$;
  if $H_s[j] - H_f[j] > 2.0mm$ & $y < H_s[j] - 2.0mm$ **then**
    move s_i from Ψ_s to Ψ_f;
    if $y > H_f[j]$ **then**
      update $H_f[j] \Leftarrow y$;
    end if
  end if
end for

8.2 Comparison between BCRE model and our three layer-model

Once a sand grain s is considered part of the fixed layer, it is no longer part of the Physx computation. Since most of sand grains are in the fixed layer while the moving sand picture is being formed, our three-layer model can reduce the computational cost significantly. Fig. 12 compares the time costs against the number of sand grains in the scene using the BCRE model and our three-layer model, where the run-time data is recorded using our three-layer model and the (two-layer) BCRE model respectively when the number of sand grains increases from 5000 to 225000 with the interval of 20000.

In the BCRE model, much time is spent on the generation of the height field and the mesh as sand grains enter the scene. The simulation time increases almost linearly as the number of sand grains in the scene increases, as shown by the blue line in Fig. 12. In our three-layer model, the simulation time also tends to converge to a constant when the number of sand grains increases, as shown by the red line in Fig. 12. This is because most of sand grains in the scene is in the fixed layer, not involved in the Physx computations. We can find the time cost of BCRE is larger than time cost of three-layer model when the number of sand grains is more than 45000. Since the number of sand grains ranges from 3.5×10^5 to 5×10^5, which is much larger than 45000, introducing the fixed layer can reduce the simulation time dramatically.

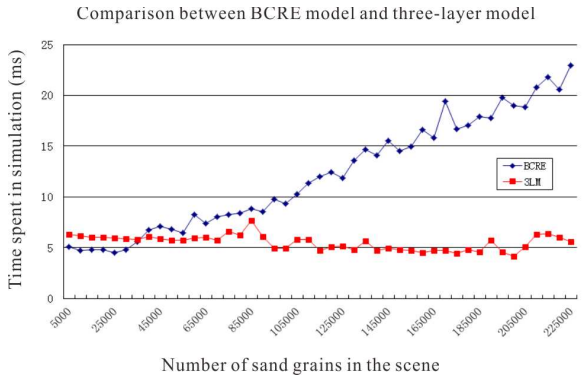
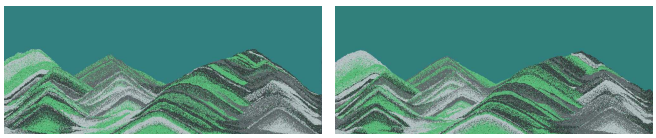


Fig. 12: *Number of sand grains in the scene v.s. Time cost in generating moving sand picture with BCRE model and our three-layer model.*



(a) (b)

Fig. 13: *Sand simulation using the BCRE model (a) and our approach (b).*

Fig. 13 shows mountain like patterns generated by the the BCRE model (left) and our three-layer model. The results show that the two methods achieve very similar simulation effects.

9 Heuristic Setting Rules

In our approach, sand moving pictures are composed of sand piles covered by ellipses used for segmentation, thus, the sizes of ellipses have direct effects on sizes of sand piles and. Semi-major axes and semi-minor axes of ellipses determine the shapes of sand piles, specifically, steep sand piles could be generated by ellipses of smaller eccentricities, and flat sand piles by ellipses of large eccentricities.

Strip patterns appearing as textures in sand piles are produced by sand grains inside rectangular cells covered by ellipses. Hence if more strip patterns are desired in sand piles, rectangular cells used for tessellation should be smaller than ellipses used for segmentation. Additionally, the colors of sand grains inside rectangular cells have effects on the contrasts of textures inside sand piles.

We could use the above observations as heuristic rules to in creating desired moving sand pictures. For instance, we could set large ellipses of smaller eccentricities to generate mountains and use smaller ellipses of large eccentricities to generate scenes like sea surfaces and deserts.

Although our approach is originally designed for generating irregular mountain-like patterns, it is able to generate interesting regular patterns with intricate details, as shown in the next section. This is achieved using rectangles with horizontal sides equaling the width of the initialization window and vertical sides of varying sizes at the tessellation stage, and curves of varying widths and heights repetitively along the horizontal axis at the segmentation stage.

10 Results

This section presents several results of moving sand pictures generated using the aforementioned approach.

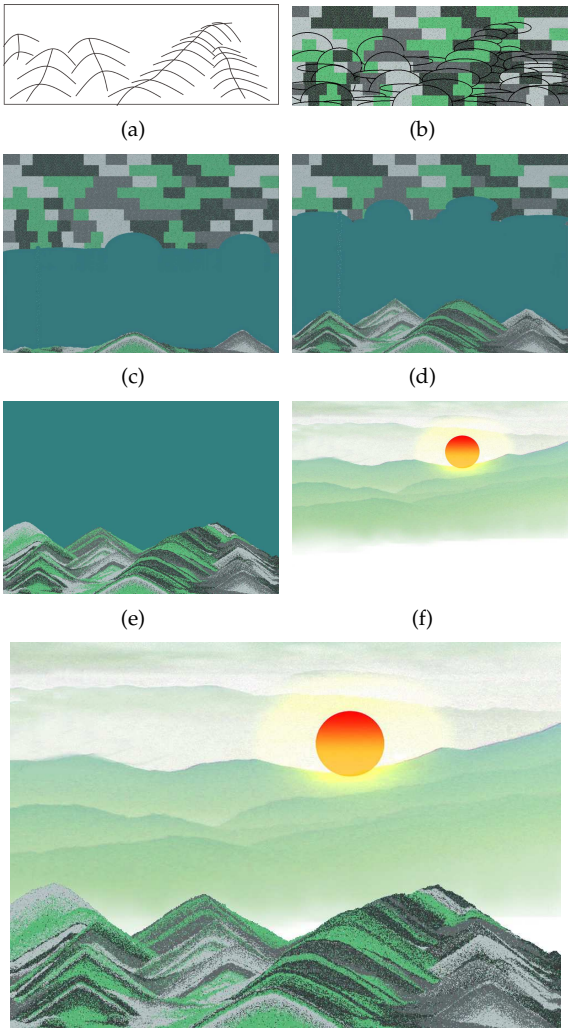


Fig. 14: *Mountain ranges. (a) The sketch. (b) Initialization and segmentation. (c) ∼ (d) Two screen shots captured during the generation process. (e) Generated mountain ranges. (f) Background picture. (g) Generated mountain ranges with the background picture added.*

Mountain ranges. Fig. 14 shows a sketch (Fig. 14(a)), segmentation cells drawn in black lines over the tessellated cells in (Fig. 14(b)), with colors of sand grains set to (51,55,56), (98,102,105), (90,190,120) and (190,198,200), respectively (RGB). Two screen shots of our working window captured during the simulation process are shown in Fig. 14(c) and 14(d), and the moving sand picture generated in Fig. 14(e)). Some moving sand picture devices enhance the realism by overlaying a background picture in the upper part of the image, rather than leaving it blank after all the sand grains have fallen to the bottom part. We could also add a background image, such as that in Fig. 14(f), to the generated moving sand pictures to enhance the realism. The enhanced result is shown in Fig.14(g).

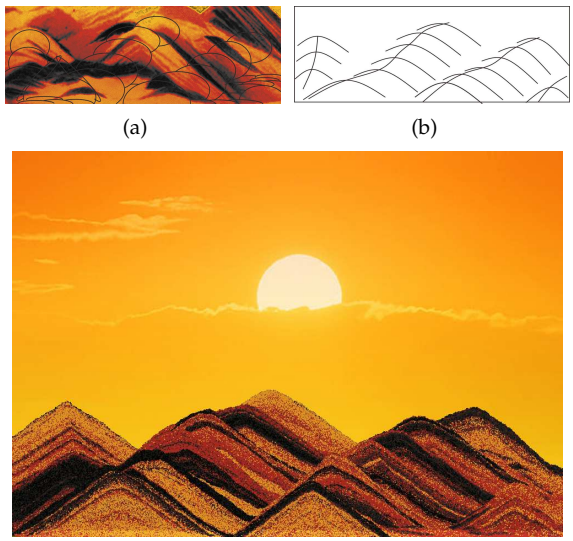
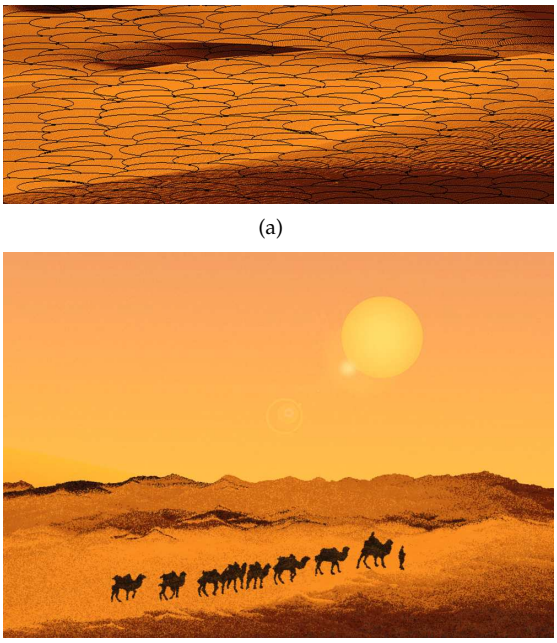


Fig. 15: *Mountains in sunset. (a) Initialization with a moving sand picture and segmentation. (b) Sketch. (c) Final moving sand picture with a sun background picture added.*

Mountains in sunset. Fig. 15 shows a moving sand picture generated by taking another moving sand picture of size 1000×350 pixels (Fig. 3(a)) as input image for initialization. After halftoning is performed on Fig. 3(a), colors of sand grains in C_{1-4} are set at $(10, 10, 10)$, $(63, 63, 63)$, $(198, 53, 59)$, and $(240, 182, 40)$, respectively. We segment the resulting image (Fig. 3(d)) by several ellipse cells (black lines in Fig. 15(a)) on the basis of sketch in Fig. 15(b). Fig. 15(c) shows the moving sand picture generated with a sunset image added.

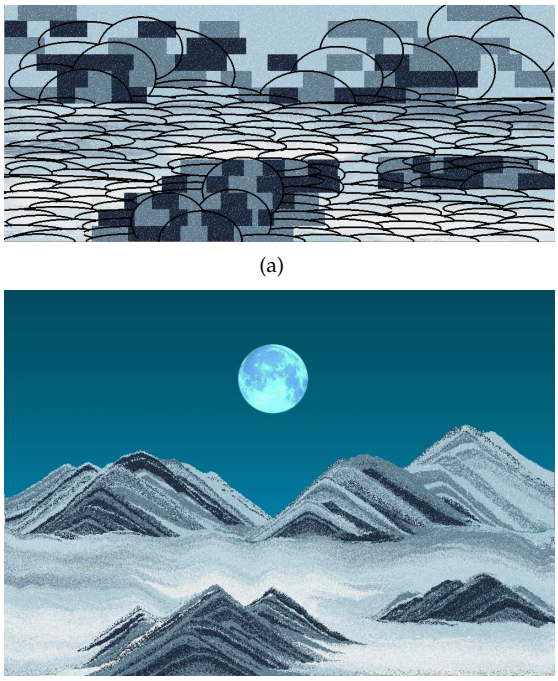
Desert with camels. Fig. 16 presents a desert-like moving sand picture generated using halftoning on an input image of a desert photograph of Fig. 16(a), the colors of sand grains in C_{1-4} are set to $(26, 12, 10)$, $(100, 30, 10)$, $(200, 100, 20)$, and $(250, 140, 30)$. In order to obtain flatten sand piles to depict desert surfaces, we adopt small segmentation cells as indicated by the black lines in Fig. 16(a) and place them over the input photograph in an automatic manner.

To make it aesthetically pleasing, we add figures/objects to the generated moving sand pictures. In order to offer additional control over the scene, we propose a simple method that places a binary mask on the scene, calculates the bounding box of the mask, and then scans the area covered by the box from top down left to right. We then render sand grains fallen on the mask using two dark colors within C_{1-4}, C_1 and C_2 with a probability of 0.7 and 0.3, respectively.



(a) (b)

Fig. 16: *Desert with camels. (a) Initialization with a photograph and segmentation. (b) Generated moving sand picture-desert with camels and human figures added.*



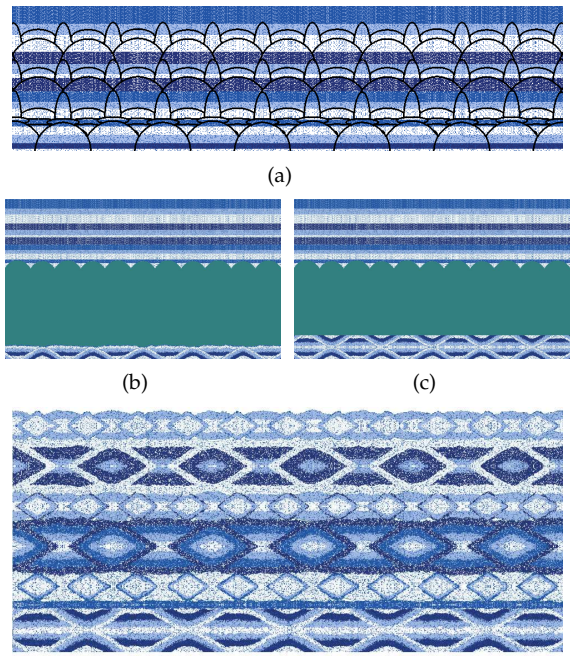
(a) (b)

Fig. 17: *Mountains & clouds. (a) Initialization and segmentation. (b) Final generated moving sand picture with a moon background picture added.*

Mountains & clouds. Fig. 17 shows an example in which different kinds of objects can be generated in a single picture, such as mountains in different distances, and the clouds between them. This is be achieved by simply dividing the initialization region into several areas for mountains and clouds, and then using large rectangles with big color variations to initialize the mountain areas and small rectangles with small color variations to initialize the cloud areas, and finally segment mountain areas with large ellipses and cloud areas with small ellipses, as indicated in Fig. 17(a). The resulting moving sand picture with a bright moon added is shown in Fig. 17(b).

Regular patterns. In addition to images of natural scenes depicted by previous moving sand pictures, our approach is also able to generate regular patterns via simple settings at the tessellation and segmentation stages, as shown in Fig. 18. At the tessellation stage, we place several horizontal rectangles of the same width and varied heights one by one vertically. At the segmentation stage, we first arrange two neighboring ellipses of different major and minor axis as a pattern unit and copy the unit along the horizontal axis (such a row corresponds to a pattern strip in the generated moving sand picture). Next, we repeat the above procedure upward to form the second row composed of different pattern units. This procedure continues until the entire initialization region is covered. To separate neighboring patterns, thin rectangles could be inserted between them.

During the generation process, each row produces a horizontal patterned strip which is asymmetric. To obtain a symmetric pattern, we could simply copy the asymmetric pattern and flip it vertically, and place flipped asymmetric pattern over the asymmetric pattern just produced. This can be done automatically in our system as indicated by Fig. 18(b) and Fig. 18(c).



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

Fig. 18: *Regular patterns. (a) Initialization and segmentation. (b)∼(c) Two screen shots captured during the generation process. (c) Final patterns produced.*

As discussed before, the sizes of ellipses have direct effects on the patterns of generated moving sand pictures. A rich variety of patterns could therefore be obtained by

simply changing the sizes of ellipses at the segmentation stage. The regular patterns generated by our approach may be hard to make using any existing painting software, thus could be used widely for textile or decorative patterns on various products, such as hats, carpets and mugs.

Our simulation system is implemented using Microsoft Visual C++, NVIDIA Physx and OpenGL, and run on a PC with 2.60 GHz CPU (Intel (R) Core), 4GB memory and NIVDIA GeForce GT 750M. Table 1 lists the statistics associated with our simulation model and the BCRE model for the previous 5 examples, where $Type$ is the type of initialization, $Initial$ is the time spent in initialization in seconds (s), $3LM$ and $BCRE$ are the time spent in our three-layer model in minutes (min) and BCRE models in hours (h), respectively. The table shows that introducing the fixed layer in our model, the simulation time is reduced by 2 to 4 folds compared with the BCRE model.

TABLE 1: Summary of processing time with different models and types of initialization.

	$\#of\ sand\ grains$	$Type$	$Initia$	$3LM$	$BCRE$
Fig. 14	3.5×10^5	Tessellation	2.373s	55m	$153m$
Fig. 15	3.5×10^5	Halftoning	0.17s	61m	$169m$
Fig. 16	3.5×10^5	Halftoning	0.145s	40m	$176m$
Fig. 17	4.3×10^5	Halftoning & Tessellation	3.223s	97m	$271m$
Fig. 18	2.6×10^5	Tessellation	2.935s	47m	$83m$

11 Conclusions and Future Work

This paper has demonstrated how moving sand pictures could be simulated graphically with parameterization and control. The modules described in different sections could be used alone or in a combinational manner. Thus, users could generate moving sand pictures similar to a moving sand picture device, and also highly artistic ones. Potential applications of our approach range from advertisements, posters, postcards, packaging, to digital arts. As a future work, we plan to extend the current method to three dimensional, so that sand sculptures in the style of moving sand pictures could be generated.

Acknowledgments

This paper is supported by Natural Science Foundation of China (No. 61772463) and (No. 61379069).

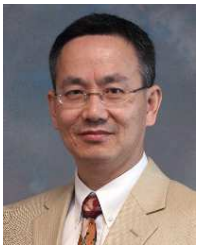
References

[1] J. J. Monaghan, "Smoothed particle hydrodynamics," *Reports on Progress in Physics*, vol. 68, no. 8, p. 1703, 2005.
[2] Y. Zhu and R. Bridson, "Animating sand as a fluid," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 965–972, 2005.
[3] T. Lenaerts and D. Philip, "Mixing fluids and granular materials," *Computer Graphics Forum*, vol. 28, no. 2, pp. 213–218, Apr. 2009.
[4] R. Narain, A. Golas, and M. C. Lin, "Free-flowing granular materials with two-way solid coupling," *ACM Transactions on Graphics*, vol. 29, no. 6, pp. 173:1–173:10, Dec. 2010.
[5] D. Wilkinson and J. F. Willemsen, "Invasion percolation: a new form of percolation theory," *Journal of Physics A: Mathematical and General*, vol. 16, no. 14, p. 3365, 1983.
[6] K. Hawick, "Gravitational and barrier effects in d-dimensional invasion percolation reservoir models," in *Proceedings of the IASTED International Conference on Power and Energy Systems and Applications*, Dec. 2011, pp. 259–266.
[7] B. Ren, C. Li, X. Yan, M. C. Lin, J. Bonet, and S.-M. Hu, "Multiple-fluid sph simulation using a mixture model," *ACM Transactions on Graphics*, vol. 33, pp. 1–11, Aug. 2014.
[8] X. Yan, Y.-T. Jiang, C.-F. Li, R. R. Martin, and S.-M. Hu, "Multiphase sph simulation for interactive fluids and solids," *ACM Transactions on Graphics*, vol. 35, no. 4, pp. 79:1–79:11, Jul. 2016.
[9] A. Luciani, A. Habibi, and E. Manzotti, "A multi-scale physical model of granular materials," *Graphics interface*, pp. 136–146, Jan. 1995.
[10] N. Bićanić, "Discrete element methods," *ncyclopedia of Computational Mechanics*, vol. 1, Oct. 2004.
[11] N. Bell, Y. Yu, and P. J. Mucha, "Particle-based simulation of granular materials," in *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, New York, USA, 2005, pp. 77–86.
[12] I. Alduán, A. Tena, and M. A. Otaduy, "Simulation of high-resolution granular media," in *Proceedings of Congreso Español de Informática Gráfica*, 2009.
[13] I. Alduán and M. A. Otaduy, "Sph granular flow with friction and cohesion," in *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, New York, USA, 2011, pp. 25–32.
[14] M. Macklin, M. Müller, N. Chentanez, and T.-Y. Kim, "Unified particle physics for real-time applications," *ACM Transactions on Graphics*, vol. 33, no. 4, pp. 153:1–153:12, Jul. 2014.
[15] D. Sulsky, S.-J. Zhou, and H. L. Schreyer, "Application of a particle-in-cell method to solid mechanics," *Computer Physics Communications*, vol. 87, pp. 236–252, Aug. 1995.
[16] A. Stomakhin, C. Schroeder, L. Chai, J. Teran, and A. Selle, "A material point method for snow simulation," *ACM Transactions on Graphics*, vol. 32, no. 4, p. 102, 2013.
[17] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin, "The affine particle-in-cell method," *ACM Transactions on Graphics*, vol. 34, pp. 1–10, Aug. 2015.
[18] G. Klár, T. Gast, A. Pradhana, C. Fu, C. Schroeder, C. Jiang, and J. Teran, "Drucker-prager elastoplasticity for sand animation," *ACM Transactions on Graphics*, vol. 35, no. 4, pp. 1–12, Jul. 2016.
[19] G. Daviet and F. Bertails-Descoubes, "A semi-implicit material point method for the continuum simulation of granular materials," *ACM Transactions on Graphics*, vol. 35, no. 4, p. 13, Jul. 2016.
[20] X. Li and J. M. Moshell, "Modeling soil: Realtime dynamic models for soil slippage and manipulation," in *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, New York, USA, 1993, pp. 361–368.
[21] B. Chanclou, A. Luciani, and A. Habibi, "Physical models of loose soils dynamically marked by a moving object," in *Proceedings of the Computer Animation*, Washington, USA, 1996, pp. 27–35.
[22] R. Sumner, J. O'Brien, and J. Hodgins, "Animating sand, mud, and snow," *Computer Graphics Forum*, vol. 18, no. 1, pp. 17–26, Mar. 1999.
[23] M. Pla-Castells, I. García-Fernández, and R. J. Martínez, *Interactive Terrain Simulation and Force Distribution Models in Sand Piles*. Berlin, Heidelberg: Springer, 2006, pp. 392–401.
[24] J.-P. Bouchaud, M. Cates, J. R. Prakash, and S. Edwards, "A model for the dynamics of sandpile surfaces," *Journal de Physique I*, vol. 4, no. 10, pp. 1383–1410, 1994.
[25] A. Aradian, É. Raphaël, and P.-G. De Gennes, "Surface flows of granular materials: a short introduction to some recent models," *Comptes Rendus Physique*, vol. 3, no. 2, pp. 187–196, 2002.
[26] K. Onoue and T. Nishita, "Virtual sandbox," in *Proceedings of the 11th Pacific Conference on Computer Graphics and Applications*, Washington, USA, 2003, pp. 252–259.
[27] B. Zhu and X. Yang, "Animating sand as a surface flow," in *Eurographics*, 2010.
[28] B. Pearce and K. Hawick, "Interactive simulation and visualisation of falling sand pictures on tablet computers," in *Proceedings of the 10th International Conference on Modeling, Simulation and Visualization Methods*, 2013.
[29] K. Kawasaki, "Diffusion constants near the critical point for time-dependent ising models. iii. self-diffusion constant," *Physical Review*, vol. 150, no. 1, pp. 375–381, Oct. 1966.
[30] Nvdia, "Physx. http://developer.nvidia.com/physx," Accessed 22 March 2012, Mar. 2012.
[31] B. Fusheng, "Method for making new moving sand picture," Patent, CN, 1172808 C., Oct. 2004.
[32] Wikipedia, "http://en.wikipedia.org/wiki/silicon_carbide," Accessed 2015, 2015.

[33] R. W. Floyd, "An adaptive algorithm for spatial gray-scale," in *Proc. Soc. Inf. Disp.*, vol. 17, Jan. 1976.
[34] B. E. Bayer, "An optimum method for two-level rendition of continuous-tone pictures," *SPIE MILESTONE SERIES MS*, vol. 154, pp. 139–143, 1999.
[35] D. E. Knuth, "Digital halftones by dot diffusion," *ACM Transactions on Graphics*, vol. 6, no. 4, pp. 245–273, 1987.

Mohan Zhang received the BSc degree in software engineering from Sichuan University, China, in 2013. Currently, he is working toward the PhD at the State Key Lab of CAD&CG, Zhejiang University, China. His research interests include computer animation, physically based modeling and non-photorealistic rendering.

Hongwei Lin received the BSc degree from the Department of Applied Mathematics at Zhejiang University, China, in 1996, and the PhD degree from Department of Mathematics at Zhejiang University in 2004. He worked as a communication engineer from 1996 to 1999. He is a professor in the Department of Mathematics, State Key Laboratory of CAD&CG, Zhejiang University. His current research interests include geometric design, computer graphics, and computer vision. He is a member of the IEEE.



Kang Zhang received the B.Eng. degree in computer engineering from the University of Electronic Science and Technology, Chengdu, China, in 1982, and the Ph.D. degree from the University of Brighton, Brighton, U.K., in 1990. He is currently a Professor and Director of Visual Computing Lab, Department of Computer Science, University of Texas at Dallas (UT-Dellas), Richardson. He is also an Adjunct Professor of the UT-Dallas Computer Engineering Program and GIS Program. Prior to joining UT-Dallas, he held academic positions in the U.K., Australia, and China. His current research interests include visual languages, aesthetic computing, and software engineering. He has published more than 180 papers in these areas. He has authored and edited five books.

Jinhui Yu received the BSc and MSc degrees in electronics engineering from Harbin Shipbuilding Engineering Institute, Harbin Engineering University, China, in 1982 and 1987, respectively. He received the PhD degree in computer graphics from the University of Glasgow in 1999. He is a professor of computer science at the State Key Lab of CAD&CG, Zhejiang University, China. He is also a guest professor at the Department of Computer Science, Harbin Finance University, China. His research interests include image-based modeling, non-photorealistic rendering, computer animation, and computer graphics art.