Vis Comput (2012) 28:869–875
DOI 10.1007/s00371-012-0698-8

ORIGINAL ARTICLE

Procedural models for cartoon cracks and fractures animations

Jing Liao · Jinhui Yu

Published online: 19 April 2012

Abstract We present an approach for animating cracks and fractures in cartoon style. In our method we take a 2D hand-drawn object as input and then construct a 2.5D model of the object in order to approximate the object volume. Next, we generate the Voronoi textures on the 2.5D object model for visual abstraction of cartoon cracks. Further, cracking gaps on the Voronoi textures are widened progressively until Voronoi cells split apart and finally fall onto ground according to simplified physical rules. With minimum user intervention, our model is able to generate cartoon cracks and fractures animations procedurally, as demonstrated by examples given in the paper.

Keywords Procedural modeling · Cracks and fractures · Non-photorealistic rendering · Cartoon animation

1 Introduction

In the Oxford dictionary some examples are given to show that a crack is used to refer to a very tiny or incompletely separated fracture. Thus, crack fracture is a process of breaking objects into small pieces which may then fall under gravity and reach their rest conditions when touching other objects such as ground as well as fallen pieces. In cartoon animation those effects are used for adding realism and drama in animation. Figure 1 shows an example of hand-drawn crack and fracture effects taken from TV commercial animations. Animating cartoon crack fractures is far from easy by hand, because it requires animators to draw many broken pieces in a frame and animate them with multiple frames in a visually convincing manner. Animator's work load can be reduced significantly if these effects can be generated by procedural models.

In recent years, some approaches of physically based modeling of crack fractures on 3D objects have been developed. However, these methods focus on realistic effects thus cannot be used straightforwardly in cartoon animation because of inconsistency in visual style. Thus, we have to seek for different solutions to animate crack fractures in cartoon style, using just 2D painted objects and the background. Pursuing this goal therefore imposes the following challenges different from those in 3D realistic crack fracture animation.

(1) Currently physically based approaches for animating crack fractures require 3D models of objects for physical computation. While in traditional cartoon animations objects are drawn in 2D, and it is very difficult to reconstruct 3D models from 2D cartoon drawings in general.
(2) Restricted by using 2D painted objects, we have to introduce 3D information such as thickness to broken shapes derived from 2D painted unbroken objects and animate them in a visually convincing manner.
(3) Sometimes, object shadows on the background maybe drawn before they crack. During the cracking process, object shadows should also vary as objects break into pieces. Our model should be able to handle shadow variations as well.

In this paper, we propose a procedural approach for modeling cartoon crack fractures. Our method takes 2D hand-drawn objects as input and then creates cracks on it and

Electronic supplementary material The online version of this article (doi:10.1007/s00371-012-0698-8) contains supplementary material, which is available to authorized users.

J. Liao · J. Yu (✉)
State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310027, China
e-mail: jhyu@cad.zju.edu.cn

J. Liao
e-mail: liaojing@cad.zju.edu.cn

Fig. 1 Hand-drawn crack and fracture effects

breaks the object into fragments. By observing cartoon crack fractures drawn by hand, we identify some simple patterns and parameters that will guide our procedural techniques and provide an interactive control to the animator.

2 Related work

The methods used in computer graphics for generating crack fractures can be loosely grouped into two categories: physical approach and non-physical approach (procedural model).

2.1 Physical approaches

Many researchers have undertaken physical simulation to model cracks. A mass–spring system was used to reproduce crack patterns in microsphere monolayers [1], and to model tree bark [2], cracks in surfaces [3], and in volumes [4]. Gobron and Chiba used cellular automata to crack multilayer surfaces [5] and simulated materials peeling off of surfaces [6]. Paint cracking and peeling were also simulated using a two-layered model on a 2D grid [7]. Federl and Prusinkiewicz used wedge-shaped finite elements to model cracks formed by drying mud and tree bark [8, 9]. Iben and O'Brien [10] extended the finite elements method to generate cracks from a stress field defined heuristically over a triangle discretization of the surface.

In addition to being used for cracks, physically based methods have also been used to simulate objects that are being broken or torn apart. Terzopoulos et al. [11] modeled elastic deformation with mass–spring systems, and this work was a precursor to the later work by Terzopoulos and Fleischer [12], which extended the approach to plastic deformation and fracture: when a spring stretches beyond its elastic limit, it breaks. Norton et al. [13] and Mazarak et al. [14] took similar approaches by attaching voxels together with springs which break when the local pressure or force exceeds a designated yield limit. The model of Norton et al. was used to produce an animation of a teapot being smashed, while that of Mazarak et al. was used to generate 3D debris from explosions. Beside mass–spring systems, finite elements have also been widely used to simulate brittle fracture [15, 16], ductile fracture [17], elasto-plastic materials and interactive fracture [18], and the fracture and deformation of voxelized surface meshes [19]. Other algorithms include generating fracture on elastic and plastic materials with the virtual node algorithm [20], a membrane-bending model for thin shell objects [21], and a meshless framework [22].

2.2 Procedural methods

The physical models tended to be slow and difficult to control. Since our work focuses on cartoon simulation of crack fractures, non-physically based procedural methods such as Voronoi diagram-based simulation are quicker and easy to control, thus they appear quite adequate for our modeling task.

In the Voronoi-based methods, cracks are produced either by interpreting all Voronoi boundaries as cracks, or by explicitly tracking the progress of a crack through the network; fragments are abstained by tessellating the object with Voronoi polygon. Worley [23] postulated a general texture synthesis primitive based on the nth-order Voronoi diagram. Although Worley did not explicitly address the use of his primitive for cracks, it is clear from the examples he provided that crack patterns can be generated using his technique. Voronoi diagrams were also suggested by Raghavachary [24] as a lightweight way of producing realistic-looking cracks. In Raghavachary's method, Voronoi sites are scattered across the polygons of an input mesh, and the Voronoi diagram computed. Mould [25] further generates Voronoi diagram of a weighted graph based on an input image, resulting image-guided fractures.

Beside Voronoi diagram, procedural crack pattern also can be generated by other ways. Wyvill et al. [26] form cracks based on the Distance Transform algorithm on a 2D surface to replicate Batik painting cracks. Martinet et al. [27] learn crack patterns from a real picture, map them to an object's surface, carve out a volume to generate crack depth and then break the object into fragments.

Our work falls into the category of procedural methods. Here we focus on cartoon animation of crack fractures rather than realistic simulation of crack fractures. Figure 2 shows an overview of our algorithm. First we take as input a 2D hand-drawn object and, in order to preserve the volume of the fallen pieces, we then cut the input object into two parts, one is the front half seen by the viewers, the other is the back

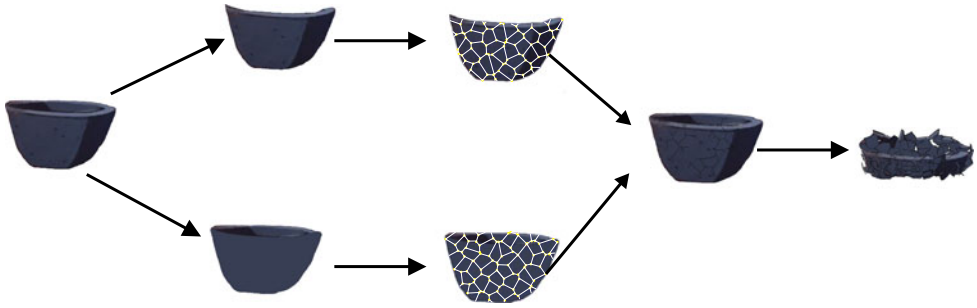


Fig. 2 Algorithm overview

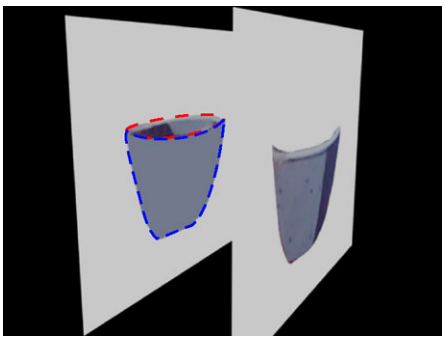


Fig. 3 2.5D model of the input object

half occluded by the front half Sect. 3. Next, the Voronoi textures are generated on these two parts (Sect. 4) to simulate the cartoon cracks. Further, the cracking gaps on the Voronoi textures are widened progressively until Voronoi cells split apart (Sect. 5) which finally fall onto ground according to simplified physical rules (Sect. 6).

3 Surface and volume approximation

In the 3D case when an object breaks into pieces, the total volume of the fallen pieces is equal to the volume of the unbroken object. While in our system the input object is drawn on 2D, thus, just breaking the drawn object into 2D broken pieces could not show the volumetric appearance of the unbroken pieces supposed to be in 3D. We have to therefore seek for a solution to introduce additional volume in the 2D broken pieces.

Usually it is very difficult to generate 3D object models from single 2D cartoon drawings, so we construct the front and back half of the input object to approximate its surface area instead. The front half (*FH*) is manually segmented from the drawn object (right of Fig. 3), and the back half (*BH*) is constructed by two parts: the first is the remaining part of the segmented input object (as indicated by red dash lines on the left of Fig. 3), and the second is the region occluded by the *FH* (as indicated by blue dash lines on the left of Fig. 3). This region is assigned the color of the original image on the non-shadow region in the remaining part of the segmented object. The two images of *FH* and *BH* are stored in two layers with depth order assigned as a 2.5D model for later use. Compared with hand-drawing many broken pieces involved in cartoon crack fractures, our interactive segmentation and construction of *FH* and *BH* significantly reduces the animator's workload.

In order to show the volumetric appearance of the broken objects in our model, it is necessary to introduce thickness to *FH* and *BH* derived above. Generally this is also a hard task as the broken objects may have varying structures which requires sufficient knowledge of the objects to add appropriate thickness to fit the 3D shape of the broken objects. Fortunately, in cartoon animation broken objects are often drawn with simple structures, which we can divide into two categories: the first includes solid objects such as sculptures and walls, etc., and the second includes objects that are vacant inside such as bowls. Details of adding thickness to these two kinds of object are given in Sect. 6.

4 Voronoi textures

Voronoi diagrams were commonly suggested in [23–25] as a way of producing realistic-looking cracks and fractures. We also adopt the well-known Voronoi diagram to simulate crack textures appearing in hand-drawn crack fractures as shown in Fig. 1.

Voronoi diagrams were first discussed by Peter Lejeune Dirichlet in 1850. However, it was more than a half of a century later in 1908 that these diagrams were written about in a paper by Voronoi, hence the name Voronoi diagrams. Given n distinct points (sites) $P = p_1, p_2, \ldots, p_n$ in the plane, the Voronoi polygon (cell) of a point p_i, $VP(p_i)$, is defined to be the set of all points q in the plane for which p_i is among the closest point to q in P. That is,

$$VP(p_i) = \{q : |p_i - q| \leq |p_j - q|, i \neq j\}. \quad (1)$$

The union of the boundaries of the Voronoi polygon is called the Voronoi diagram of P, denoted by $VD(P)$, as shown by solid lines in Fig. 4. A Voronoi edge is the boundary between two Voronoi polygons and a Voronoi vertex is the intersection of three or more Voronoi edges.

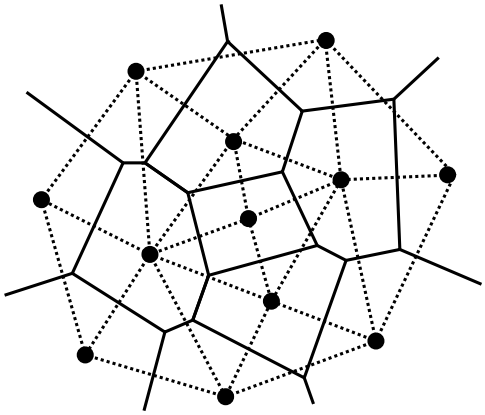


Fig. 4 Voronoi diagram (*solid*) and Delaunay triangulation (*dot*)

There are many algorithms for constructing the Voronoi diagram [28], including the calculation of Voronoi diagrams using graphics hardware [29]. We adopt the simple algorithm by Lischinski [30] for the incremental construction of the Delaunay triangulation and the Voronoi diagram. For implementation details of the algorithm, please refer to [30].

In our system, we synthesize a Voronoi diagram on two layers of FH and BH as the basic model for crack fractures. This involves seeding the interior of contours with seed points distributing evenly or centrically, triangulating the resulting point set, and constructing the Voronoi network. The $VD(P)$ constructed above only models underlying structures of the cartoon crack fractures. To animate these cartoon effects we need to incorporate appropriate rendering and controlling mechanisms, as detailed in the following two sections.

5 Modeling cracking

Cracking is a process that usually starts from some positions where the object is hit, cracks advance on the object and the gaps of cracks widen progressively until the object is broken into pieces. Thus modeling cracking involves simulation of crack advancing and gap widening along the edges of Voronoi cells in the Voronoi network.

Firstly, several Voronoi vertices are chosen by the user as activating seeds, as shown by the red points in Fig. 5(a). Each activating seed nucleates a cracking line for each connecting edge, and the cracking line zigging along its corresponding edge at a certain speed, set by $L = L_0 + \Delta L * t$, where L_0 is the initial length, ΔL the speed and t the time, as shown by the dark bold lines in Fig. 5(b). When a cracking line reaches an inactivated vertex, this vertex became a new activating seed, which trigs more cracking lines, as illustrated in Fig. 5(c). The cracking propagation process terminates until all cracking lines have reached the contours or vertices activated previously.

As for crack widening, the earlier emerged crack lines are wider than those new ones. We model this by assigning age to the cracking line width; the older, the wider. In our system we set the speed of cracking line widening by $W = W_0 + \Delta W * t$, where W_0 is the initial width, ΔW the speed and t the time. Cracks animating starts on the *FH* layer first, and then to the *BH* layer, as shown in Fig. 6.

6 Modeling fracturing

When cracking lines transverse the whole object, the second phase begins: the object shatters into small fragments which then fall on other objects such as floor and ground. During the falling period, fragments may interact with each other under the gravity, causing rotation, collision and piling up of fragments, such as those happened in 3D.

Fragments obtained with above procedure are, however, just 2D shapes of Voronoi cells, which could not be used straightforwardly to produce realistic rotations, collision and piling up of fragments as they appear in 3D. To add realism to cartoon fracturing, we add thickness to each fragment to

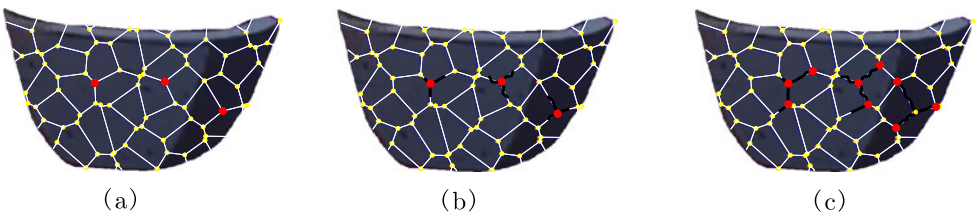


Fig. 5 Algorithm of dynamic cracking texture

Fig. 6 Cracking texture animated in three different frames

obtain its 3D counterpart, which is far easier than constructing the complex 3D model for the broken object. We construct a 3D fragment by copying a Voronoi cell and move it along the direction of the cell normal to a distance denoted by d, which equals the thickness of the fragment specified by the user, as illustrated in Fig. 7. Faces other than the front one is assigned by the user.

For the solid objects such as sculptures and walls, the thickness of the fragments should be increased so that they appear reasonably bigger than the thinner fragments derived from some objects such as bowls. In our system we simply increased the thickness of fragments for solid objects to $1.5d$, which is sufficient to simulate cartoon fragments derived from solid objects.

Once the 3D models for fragments are obtained, they are animated physically. Suppose all fragments are made of the same materials and the same density, they all fall as the solid bodies to the ground plane under gravity. When collision of fragments arises during falling, we treated it as the elastic collision or inelastic collision according to the material defined. In our implementation, fragments falling, colliding and piling up are simulated by use of the Physx library [31].

Since the background picture in cartoon is also painted on 2D, there is no 3D information available for calculating the collision between falling fragments and the ground. We therefore set up a virtual 3D ground plane interactively on the background picture to support the physical simulation of fragments falling and piling up process.

We also set some mechanical characteristics required by Physx, including the density of the object, the gravity, the restitution parameter, friction parameter. With these settings, the position of every fragment in each frame can be calculated by Physx library. In order to speed up the process, we use the bounding box instead of the 3D mesh of each fragment in colliding detection.

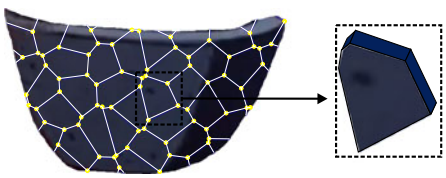


Fig. 7 Modeling 3D fragment

7 Treatment of shadows

Shadows of broken objects are treated in two phases: the first is cracking phase in which objects just begin to crack, thus objects still remain their shapes, so do their shadows. Thus, no shadow animation is required in this phase. The second is a fracturing phase in which the objects break down into pieces, consequently shadows of broken pieces vary as they fall. Since we have obtained 3D fragments using the procedure described in Sect. 6, shadow animation can be done with the traditional shadow map algorithm.

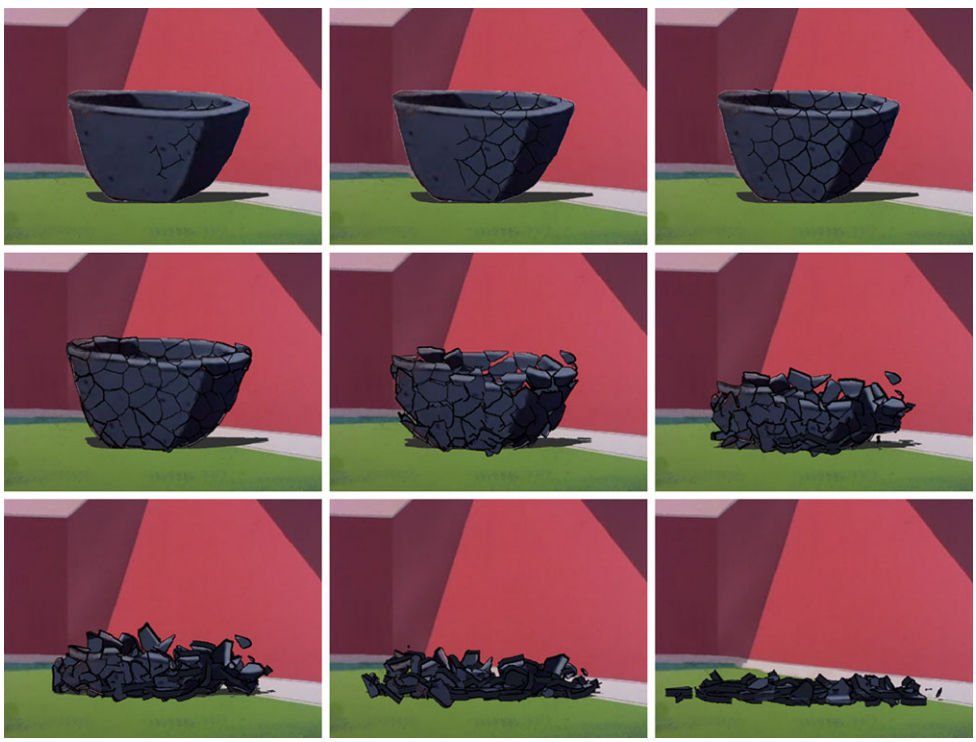


Fig. 8 Cracking and fracturing of a bowl

Fig. 9 Cracking and fracturing of a house

Fig. 10 Cracking and fracturing of a cartoon character

8 Results

To demonstrate the usability of our system, we have created three animations of cartoon crack fractures under varying conditions. Our system is implemented with Microsoft Visual C++ and OpenGl Libraries. All cartoon cracking and fracturing animations are generated in real time on a 2.5 GHz Pentium PC with 2048 MB of RAM.

In the first example we show a cracking and fracturing bowl. The bowl first cracks and then shatters into pieces, and finally falls on the ground, as shown by a strip of 9 frames sampled from the animation in Fig. 8. The second example is a cartoon house breaking, as shown by a strip of eight frames in Fig. 9.

The last example is a cartoon character breaking, as shown by the example of Cinderella in Fig. 10. Such effects may be used in cartoon, say Cinderella becomes a sculpture made of mud when cursed by the witch, and then breaks.

9 Conclusion

In this paper we present a theme of modeling cracking and fracturing effects for cartoon animation. Our work is the first attempt to model cracking and fracturing effects for 2D cartoon objects, procedurally.

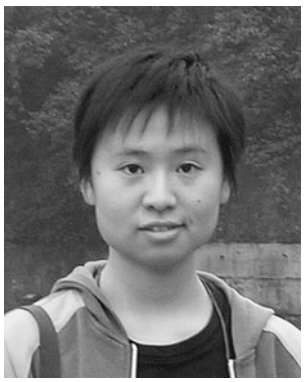
The limitation of our method is that the resultant animation of crack fractures cannot be seen from arbitrary views in 3D. This is because we only reconstruct the 2.5D model for cartoon objects instead of their full 3D model. Nevertheless, this is not a serious problem because in cartoon animation they are often depicted most expressive content seen from certain view angles and our model seems adequate for the task at hand.

In future work we intend to model more complex effects in cartoon animation including the explosions, earthquake, etc.

Acknowledgements This work is supported by the State Key Program of National Natural Science Foundation of China (No. 60933007), the Key Technologies R&D Program of China (No. 2007BAH11B02).

References

1. Skjeltorp, A.T., Meakin, P.: Fracture in microsphere monolayers studied by experiment and computer simulation. Nature **335**, 424–426 (1988)
2. Federl, P., Prusinkiewicz, P.: A texture model for cracked surfaces, with an application to tree bark. In: Proceedings of the 7th Western Computer Graphics Symposium (1996)
3. Hirota, K., Tanoue, Y., Kaneko, T.: Generation of crack patterns with a physical model. Vis. Comput. **14**(3), 126–137 (1998)
4. Hirota, K., Tanoue, Y., Kaneko, T.: Simulation of three-dimensional cracks. Vis. Comput. **16**, 371–378 (2000)
5. Gobron, S., Chiba, N.: Crack pattern simulation based on 3d surface cellular automaton. In: Proceedings of the International Conference on Computer Graphics (2000)
6. Gobron, S., Chiba, N.: Simulation of peeling using 3d-surface cellular automata. In: Proceedings of the 9th Pacific Conference on Computer Graphics and Applications (2001)
7. Paquette, E., Poulin, P., Drettakis, G.: The simulation of paint cracking and peeling. In: Proceedings of Graphics Interface (2002)
8. Federl, P., Prusinkiewicz, P.: Modelling fracture formation in bi-layered materials, with applications to tree bark and drying mud. In: Proceedings of the 13th Western Computer Graphics Symposium (2002)
9. Federl, P., Prusinkiewicz, P.: Finite element model of fracture formation on growing surfaces. In: Lecture Notes in Computer Science, vol. 3037, pp. 138–145 (2004)
10. Iben, H., O'Brien, J.: Generating surface crack patterns. Graph. Models **91**, 198–208 (2009)
11. Terzopoulos, D., Platt, J., Barr, A., Fleischer, K.: Elastically deformable models. Comput. Graph. **21**, 205–214 (1987)
12. Terzopoulos, D., Fleischer, K.: Modeling inelastic deformation: viscoelasticity, plasticity, fracture. Comput. Graph. **22**, 269–278 (1988)
13. Norton, A., Turk, G., Bacon, B., Gerth, J., Sweeney, P.: Animation of fracture by physical modeling. Vis. Comput. **7**(4), 210–219 (1991)
14. Mazarek, O., Martins, C., Amanatides, J.: Animating exploding objects. In: Proceedings of the Graphics Interface, pp. 211–218 (1999)
15. Neff, M., Fiume, E.: A visual model for blast waves and fracture. In: Proceedings of Graphics Interface, pp. 193–202 (1999)
16. O'Brien, J., Hodgins, J.: Graphical modeling and animation of brittle fracture. Comput. Graph. **33**, 137–146 (1999)
17. O'Brien, J., Bargteil, A., Hodgins, J.: Graphical Modeling and Animation of Ductile Fracture. In: Proceedings of ACM SIGGRAPH (2002)
18. Muller, M., Gross, M.: Interactive virtual materials. In: Proceedings of Graphics Interface (2004)
19. Muller, M., Teschner, M., Gross, M.: Physically-based simulation of objects represented by surface meshes. In: Proceedings of the Computer Graphics International (2004)
20. Molino, N., Bao, Z., Fedkiw, R.: A virtual node algorithm for changing mesh topology during simulation. ACM Trans. Graph. **23**(3), 385–392 (2004)
21. Gingold, Y., Secord, A., Han, J.Y., Grinspun, E., Zorin, D.: A discrete model for inelastic deformation of thin shells. In: Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (2004)
22. Pauly, M., Keiser, R., Adams, B., Dutre, P., Gross, M., Guibas, L.J.: Meshless animation of fracturing solids. In: Proceedings of the ACM SIGGRAPH (2005)
23. Worley, S.: A cellular texture basis function. In: Proceedings of SIGGRAPH, pp. 291–294 (1996)
24. Raghavachary, S.: Fracture generation on polygonal meshes using Voronoi polygons. In: Proceedings of SIGGRAPH (Sketches), p. 187 (2002)
25. Mould, D.: Image-guided fracture. In: Proceedings of Graphics Interface (2005)
26. Wyvill, B., van Overveld, K., Carpendale, S.: Rendering cracks in batik. In: Proceedings of the 3rd International Symposium on Nonphotorealistic Animation and Rendering (2004)
27. Martinet, A., Galin, E., Desbenoit, B., Akkouche, S.: Procedural modeling of cracks and fractures. In: Proceedings of International Conference on Shape Modeling and Applications (2004)
28. Aurenhammer, F.: Voronoi diagrams—a survey of a fundamental geometric data structure. ACM Comput. Surv. **23**(3), 345–405 (1991)
29. Hoff, K.E., Culver, T., Keyser, J., Lin, M., Manocha, D.: Fast computation of generalized Voronoi diagrams using graphics hardware. In: Proceeding of the 26th Annual Conference on Computer Graphics and Interactive Techniques, pp. 277–286 (1999)
30. Lischinski, D.: Incremental Delaunay triangulation. In: Heckbert, P. (ed.) Graphics Gems IV, pp. 47–59. Academic Press, Boston (1994)
31. Nvdia: PhysX. http://developer.nvidia.com/physx (2012). Accessed 22 March 2012



Jing Liao is currently a graduate student in the State Key Lab of CAD&CG, Zhejiang University, China. Her research interest includes computer animation and non-photorealistic rendering.

Jinhui Yu is a professor of computer science at Zhejiang University, China. He received his Ph.D. in computer graphics from the University of Glasgow in 1999. His research interest includes stylized computer animation and computer graphics art.