Visual Comput (2005) 21: 717–726
DOI 10.1007/s00371-005-0317-z

ORIGINAL ARTICLE

Brett Desbenoit
Eric Galin
Samir Akkouche

Modeling cracks and fractures

Published online: 31 August 2005
© Springer-Verlag 2005

B. Desbenoit (✉) · E. Galin · S. Akkouche
LIRIS, CNRS, Université Claude Bernard Lyon 1, France
{brett.desbenoit, eric.galin, samir.akkouche}@liris.cnrs.fr

Abstract This paper presents an interactive method for modeling cracks and fractures over a variety of materials such as glass, metal, wood, and stone. Existing physically based techniques are computationally demanding and lack control over the fracture propagation. Our approach consists in editing 2D fracture pattern and profile curves which are stored in an atlas according to material type. The fracture model is then automatically mapped onto the surface of the object and fractures are created by carving out a procedurally generated swept volume. Because the objects need not be voxelized or tetrahedralized as with physically based techniques, we are not limited in resolution when creating the geometry of cracks, which enables us to model small or very thin fractures.

Keywords Modeling · Fractures · Cracks

1 Introduction

Modeling complex and realistic shapes is still a challenging and important problem in computer graphics. The challenge stems not only from the complexity of the geometry and texture of the shapes, but also from the many surface details such as cracks, erosion, or patina produced by the aging and weathering of objects interacting together and with their environment. Beautiful and realistic images of complex natural environments have been produced by many computer graphics researchers and artists in the film industry. Unfortunately, the rendered models are often too perfect, which betrays the synthetic nature of the scene.

Several techniques simulating the physical and chemical phenomena that age stone [2, 7] and metallic structures [5, 6] in a natural environment have been proposed. In this paper, we present an interactive technique for modeling cracks and fractures over solid objects. Cracks and fractures are conspicuous in nature and may be found on wood, tree bark, stone, glass, ice, or dried clay. They play an important role in the realism of a natural scene, as their presence provides the viewer with a hint about the age of an object as well as indirect indications about the weather and the characteristics of the environment. Our approach is phenomenological: by observing real-world fractures and cracks, we identify some simple patterns and parameters that will guide our procedural techniques and will provide an interactive control to the designer.

1.1 Related work

Realistic animation of breaking objects is a challenging task in computer animation. Breaking an object often creates many small and interlocking pieces. The complexity of these fragments makes modeling by hand impossible. Consequently, the simulation of cracking, breaking, and shattering has received some attention in the computer graphics community.

Animating and simulating fractures. Most existing techniques for animating and simulating fractures rely on involved and computationally demanding physically based simulations to compute crack propagation and create fragments [18, 21, 22]. Such methods are indispensable for

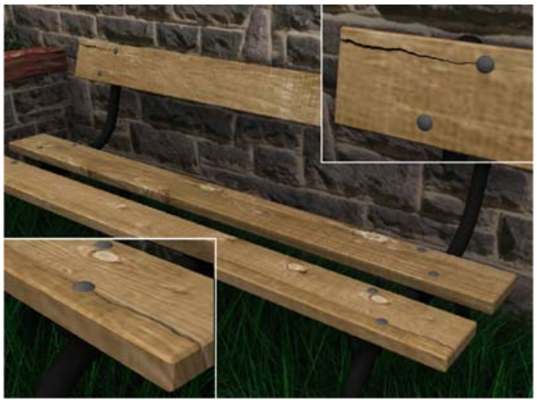

Fig. 1. Cracks on a wood bench

correct and accurate simulation of shattering and breaking and have produced animations of striking realism.

O'Brien and Hodgins [21] proposed a method for modeling and animating brittle fracture by analyzing the stress tensors computed over a finite element model. This model was extended in [22] to model ductile fractures for a wider range of materials. Smith et al. [23] presented a technique for shattering brittle objects using a set of point masses connected by distance-preserving linear constraints. Mûller et al. [18] used a hybrid implicit–explicit integration scheme to compute deformations and fracture of stiff materials in real time.

Several specific techniques have been proposed to address the fragmentation of solid objects induced by explosions. Mazarak et al. [16] used a voxel-based approach to model solid objects that break apart when they encounter a blast wave. Nef and Fiume [19] proposed a recursive pattern generator to divide a planar region into polygon shards.

Modeling static crack patterns. Several techniques have been proposed for modeling static crack patterns on dry mud, tree bark or even ceramics. Gobron et al. [10] described a method for modeling the propagation of cracks on the surface of objects using a cellular automata. Hirota et al. [11, 12] developed a mass-spring system for simulating the static crack and fracture patterns created by drying mud. Mass-spring techniques have also been used to simulate cracks on tree bark [8]. While those methods can create small cracks, they cannot represent large open fractures. Hybrid approaches combining procedural and physically based techniques have been proposed as well [14].

Some recent research addresses fracture pattern formation induced by growth or shrinkage [9], which occurs in bark formation or drying clay. The proposed method improves finite element methods to efficiently capture growth and fracture. Wang et al. [25] proposed an alternative image-based technique to model cracks in bark as a textured height field after processing an input image.

Modeling and rendering scratches. In the same area of surface defects, several methods have been proposed for modeling and rendering scratches [1, 17]. Those methods do not explicitly model the true microgeometry of a crack. Physically based simulations often require the discretization of objects into voxels or tetrahedral cells to compute internal forces. This discretization often leads to some artifacts in the crack pattern, which makes fragments look rather unrealistic. Those artifacts are the more visible as fractures are propagated along the boundaries of the initial mesh or voxel grid. Moreover, it is difficult to control a simulation so that breaking should occur only in a given region and so that some fragments should have a specific user-defined shape. Therefore, their usage may be cumbersome for interactive modeling applications.

1.2 Overview of our method

Our technique may be split into two steps:

Modeling template fracture models. First, template fracture models are edited after real-world images and stored into an atlas of generic models according to their corresponding type of material. Fractures are characterized by a 2D fracture pattern defining their branching structure and by a set of profile curves that defines the cross sections of the cracks.

In our interactive template fracture editor, the designer simply needs to specify a few control points for the crack pattern and some control profile curves. The final crack pattern is generated automatically by perturbing or smoothing the coarse crack pattern according to the type of the material, whereas the intermediate profile curves are automatically generated by a simple curve interpolation scheme.

Creating cracks and fractures. The designer first selects a fracture model in the atlas. This model is automatically mapped onto the surface of the original object to create a 3D crack volume that adapts to the surface of the object.

The designer can control the orientation of the fracture on the object, edit its profile curve, or change some of its shape parameters with interactive feedback. Cracks are created by using a Boolean difference operation between the original input model and the carving volume as presented in [15].

1.3 Contributions

The main contributions of this paper are as follows.

Template fracture models. Our approach enables us to capture the complexity and diversity of crack patterns and profiles. Template fracture models are stored in an atlas according to material type. The designer can quickly se-

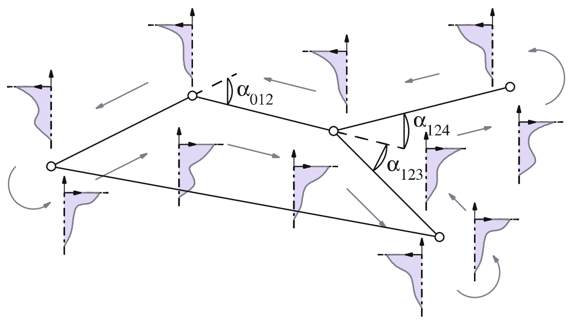

Fig. 2. Overview of structure of a fracture model. The *arrows* illustrate the traversal of the graph

lect and apply fracture models and interactively edit complex shapes.

Fracture geometry. Because the objects need not be voxelized or tetrahedralized as with physically based techniques, we are not limited in precision or resolution when creating fragments. Therefore, we avoid shape artifacts resulting from the discretization preprocessing step required by physically based techniques and we can easily create the true geometry of small or very thin fractures such as cracks in glass (Fig. 12).

Animating cracks. We propose a technique for animating the propagation of a fracture over a surface by parameterizing the crack pattern by time steps that trigger and stop crack growth. This method enables us to control the propagation speed easily and efficiently.

The remainder of this paper is organized as follows. Section 2 describes the structure of our template fracture models. Section 3 addresses the creation of the carving volume. Section 4 describes a technique for animating the propagation of a fracture over an object. We conclude the paper by a presentation of several images illustrating realistic cracks over a variety of objects of different materials in Sect. 5, followed by a discussion of our results and open problems for future research.

2 Template crack models

A template crack model $\mathcal{F} = \{\mathcal{P}, \mathcal{C}\}$ is characterized by a crack pattern $\mathcal{P}$ and a set of profile curves $\mathcal{C}$ that represent the cross sections of the crack (Fig. 2).

2.1 Crack patterns

The crack pattern is implemented as an oriented graph, denoted as $\mathcal{G}$. The connectivity of the graph represents the branching structure of the crack pattern. The nodes of the graph, denoted as $\mathcal{N}$, store the profile curves that define the cross section of the crack. The corresponding theoretical crack volume is produced by recursively traversing the graph and interpolating the crack profile curves along the arcs of the graph. This step will be addressed in detail in Sect. 3.

Every node in the graph stores two or more profile curves, depending on its connectivity. More precisely, unary nodes, i.e., leaf nodes, and binary nodes store two profile curves, whereas n-ary nodes store n profile curves (Fig. 2). The profile curves at the nodes may be of a different geometry, which enables us to model a vast variety of crack patterns for different types of materials.

Let $\mathcal{N}_j$ denote a leaf node and $\mathcal{N}_i$ its parent node. $\mathcal{N}_j$ stores two profile curves which will be denoted as $\mathcal{C}_{ij}$ and $\mathcal{C}_{ji}$ (Fig. 2). Otherwise, in the general case, $\mathcal{C}_{ijk}$ will refer to the profile curve at node $\mathcal{N}_j$ when traversing the graph from $\mathcal{N}_i$ to $\mathcal{N}_k$.

The arc between two nodes $\mathcal{N}_i$ and $\mathcal{N}_j$ will be referred to as $\mathcal{A}_{ij}$. The relative angle between the directions of two consecutive arcs $\mathcal{A}_{ij}$ and $\mathcal{A}_{jk}$ will be denoted as α_{ijk}. The arcs $\mathcal{A}_{ij}$ are valuated by the distance between two nodes $\mathcal{N}_i$ and $\mathcal{N}_j$ denoted as δ_{ij}. The distance between two nodes in the graph is obtained by traversing the graph and evaluating the minimum distance.

2.2 Crack profile curves

A profile curve is defined as a piecewise cubic spline curve whose control points will be referred to as $\boldsymbol{p}_i$, $i \in [0, n-1]$, where n denotes the number of control points. The profile curves may be of any type and may define nonconvex or even more complex cross sections (Fig. 3).

Interpolating profile curves. The theoretical crack volume is produced by recursively traversing the graph and interpolating the crack profile curves along the arcs of the graph. Therefore, we need to be able to compute a set of curves interpolating an initial and final profile curve $\mathcal{C}_i$ and $\mathcal{C}_j$. In our system, we rely on the curve morphing technique described in [3]. This method enables us to create nonintersecting curves interpolating an initial and final control curve easily.

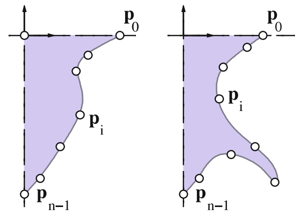

Fig. 3. Some crack profile curves

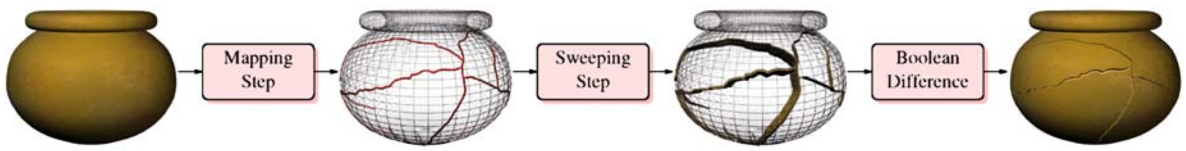

Fig. 4. Overview of crack modeling pipeline

3 Modeling fractures onto objects

In this section, we describe our method for creating cracks on an object. Given an initial object $\mathcal{O}$ and a crack model $\mathcal{F}$, the overall algorithm may be outlined as illustrated in Fig. 4. First, we map the crack pattern $\mathcal{P}$ onto the surface of the object so as to create a 3D skeleton $\mathcal{S}$. Then, we create the profile curves at the vertices of the skeleton by interpolating the profile curves $\mathcal{C}_i$ of the crack model and orienting them according to the local normal of the surface of the object. We define the crack volume $\mathcal{V}$ as a piecewise generalized cylinder produced by sweeping the interpolating profile curves along the skeleton. Eventually, we carve the crack volume $\mathcal{V}$ out of the original object by computing the Boolean difference $\mathcal{O} - \mathcal{V}$. In our system, we use standard techniques for computing the Boolean operations between two meshes.

3.1 Creation of 3D skeleton

The creation of the 3D skeleton is performed by mapping the crack pattern $\mathcal{P}$ onto the surface of the object. Given an anchor point on the object $\boldsymbol{c}_0$ and an initial direction $\boldsymbol{u}_0$, the algorithm progressively marches along the triangle mesh and simultaneously traverses the graph $\mathcal{G}$ using the distance and angle parameters δ_{ij} and α_{ijk} stored at the nodes and the arcs to compute the shape of the skeleton.

The nodes $\mathcal{N}_i$ of the crack pattern are transformed into vertices $\boldsymbol{c}_i$. The arcs $\mathcal{A}_{ij}$ are transformed into a set of n line segments $[\boldsymbol{s}_k, \boldsymbol{s}_{k+1}]$, $k \in [0, n-1]$, where $\boldsymbol{s}_0 \equiv \boldsymbol{c}_i$ and $\boldsymbol{s}_n \equiv \boldsymbol{c}_j$. Vertices $\boldsymbol{s}_k$ are the intersections between the edges of the triangle mesh of the object and the crack pattern propagating over the object (Fig. 5).

For every vertex $\boldsymbol{s}_k$ of the skeleton we compute a local reference frame $\mathcal{R}_k = (\boldsymbol{s}_k, \boldsymbol{d}_k, \boldsymbol{t}_k, \boldsymbol{n}_k)$ according to the local curvature of the surface and according to the direction of the propagation of the crack. Vector $\boldsymbol{d}_k$ denotes the direction of the propagation of the crack at vertex $\boldsymbol{s}_k$, and $\boldsymbol{n}_k$ denotes the normal of the surface at this point. If $\boldsymbol{s}_k$ lies inside a triangle, then $\boldsymbol{n}_k$ is the normal of the triangle. Otherwise, if $\boldsymbol{s}_k$ is on an edge or a vertex, then $\boldsymbol{n}_k$ is the average of the normals of the neighboring triangles. The surface tangent vector is defined as $\boldsymbol{t}_k = \boldsymbol{n}_k \wedge \boldsymbol{u}_k$.

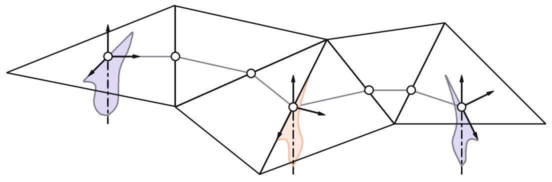

Fig. 5. Structure of 3D skeleton produced by mapping crack pattern onto surface

3.2 Creation of profile curves

The cross sections $\mathcal{X}_k$ of the crack volume at vertices $\boldsymbol{s}_k$ of the skeleton are obtained by interpolating the profile curves located at vertices $\boldsymbol{s}_0 \equiv \boldsymbol{c}_i$ and $\boldsymbol{s}_n \equiv \boldsymbol{c}_j$ and locating them in the local reference frame $\mathcal{R}_k$ (Fig. 5). Recall that the length of the skeleton between $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$ is equal to δ_{ij}. We define δ_{ik} as the sum of the lengths of the line segments between vertices $\boldsymbol{s}_0 = \boldsymbol{c}_i$ and $\boldsymbol{s}_k$:

$$\delta_{ik} = \sum_{j=0}^{j=k-1} \|\boldsymbol{s}_{j+1} - \boldsymbol{s}_j\|$$

Thus, for all vertices $\boldsymbol{s}_k$, we define its corresponding cross section as the linear interpolation:

$$\mathcal{X}_k = \frac{\delta_{ik}}{\delta_{ij}} \mathcal{C}_i + \left(1 - \frac{\delta_{ik}}{\delta_{ij}}\right) \mathcal{C}_j$$

3.3 Carving volume generation

The overall carving volume is obtained by traversing the skeleton structure and generating swept volumes for every line segment and every terminating vertex.

For every line segment $[\boldsymbol{s}_k, \boldsymbol{s}_{k+1}]$ of the skeleton, the carving volume is defined as the generalized cylinder connecting the corresponding cross sections $\mathcal{X}_k$ and $\mathcal{X}_{k+1}$ (Fig. 6). We generate the carving volume at the end vertices of the skeleton by sweeping and interpolating the two cross sections around the axis passing by the end vertex $\boldsymbol{s}_k$ and with a direction $\boldsymbol{n}_k$ (Fig. 6).

3.4 Adaptive resampling of skeleton

In some cases two consecutive cross sections $\mathcal{X}_k$ and $\mathcal{X}_{k+1}$ located at vertices $\boldsymbol{s}_k$ and $\boldsymbol{s}_{k+1}$ of the skeleton

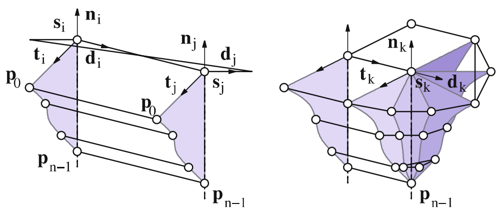

Fig. 6. Swept volumes defining carving volumes

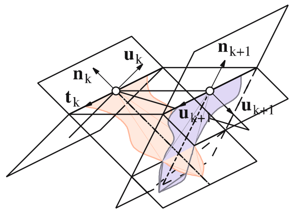

Fig. 7. Intersecting cross sections that may occur in high curvature regions

may self intersect, which produces an inconsistent swept volume for the corresponding segment $[\boldsymbol{s}_k, \boldsymbol{s}_k+1]$. Such cases arise in particular whenever the local curvature of the surface is too high compared to the size of the cross sections, or whenever the crack pattern makes a sharp turn (Fig. 7).

Inconsistent swept volumes should be avoided as they prevent the correct computation of Boolean differences. Therefore, we propose a technique that adapts the skeleton to the local curvature of the object to avoid self intersecting cross sections.

We perform an adaptive resampling of the skeleton by merging or splitting vertices so as to adapt the local density of vertices to the curvature of the mesh.

Merging process. This step aims at adapting the number of vertices on the skeleton to the size of the cross sections. Let r_i and r_j denote the radius of the circle centered at vertices $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ on the skeleton and enclosing the corresponding cross sections $\mathcal{X}_i$ and $\mathcal{X}_j$. The two cross sections may intersect if

$$\delta_{ij} < r_i + r_j$$

Thus, if $\delta_{ij} < r_i + r_j$, we merge the two vertices $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ into a single new vertex $\boldsymbol{s}_k$. The corresponding cross section is evaluated as a linear interpolation:

$$\boldsymbol{s}_k = \frac{r_i\, \boldsymbol{s}_i + r_j\, \boldsymbol{s}_j}{r_i + r_j} \qquad \mathcal{X}_k = \frac{r_i\, \mathcal{X}_i + r_j\, \mathcal{X}_j}{r_i + r_j}$$

Splitting process. The splitting process aims at inserting new vertices in the skeleton if the length of a line segment δ_{ij} is too large compared to the size of the cross sections. This process produces more regular skeletons, which in turn results in more evenly sampled carving volumes. A line segment between two vertices $\boldsymbol{s}_i$ and $\boldsymbol{s}_j$ is split if

$$r_i + r_j < 2\,\delta_{ij}$$

In that case, we insert a new vertex $\boldsymbol{s}_k$ and a corresponding cross section $\mathcal{X}_k$ over the line segment as for the merging process.

3.5 Handling noisy bumpy surfaces

Although our previous technique can handle surfaces with varying triangle densities and regions of high curvature, they may fail at creating a consistent carving volume for very rough or noisy surfaces. Such cases arise when creating fractures over highly bumped surfaces such as tree bark or rough stone. In those cases, mapping the crack pattern onto the exact surface of the object produces a very complex skeleton with many interlocking segments which cannot be simplified by the split and merge algorithm described in the previous paragraphs.

Our approach consists in using a smooth carrier surface, denoted as $\widetilde{\mathcal{O}}$, that approximates the surface of the original object (Fig. 8). The new skeleton $\widetilde{\mathcal{S}}$ is generated by mapping the crack pattern $\mathcal{P}$ onto the carrier surface. The carving volume $\widetilde{\mathcal{V}}$ is generated as described in Sect. 3.2 by sweeping the profile curves along the skeleton, and the final cracked object is defined as $\mathcal{O} - \widetilde{\mathcal{V}}$.

The approximating surface should be smooth and should embed the original object so as to avoid the formation of tunnels when carving the crack volume out of the object. The carrier surface is created after the original object by using standard mesh smoothing techniques [4, 13]. The vertices of the new mesh located inside the original object are relocated toward the original surface so that $\widetilde{\mathcal{O}}$ should embed $\mathcal{O}$.

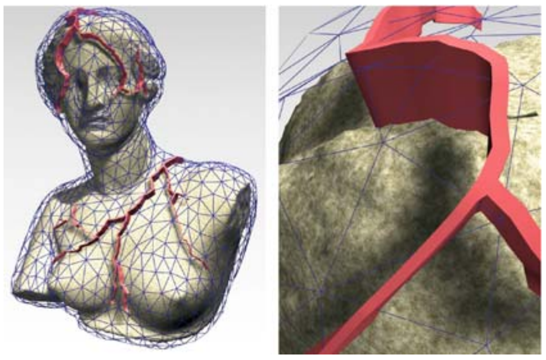

Fig. 8. Mapping crack pattern on approximating smooth mesh embedding original Aphrodite stone statue

Figure 8 illustrates this process. To create cracks on the highly detailed and bumped mesh of the stone statue (92188 triangles), we first created a corresponding smooth carrier surface (blue transparent mesh with only 3042 triangles). The carving volume (pink shaded) was generated by projecting the crack pattern on the carrier surface.

4 Animating crack propagation

In this section, we present a technique for animating the propagation of a fracture over the surface of an object. Given a static fracture model $\mathcal{F} = \{\mathcal{P}, \mathcal{C}\}$, we define the time-varying fracture model $\mathcal{F}(t) = \{\mathcal{P}(t), \mathcal{C}(t)\}$ that characterizes the propagation of the crack. The overall animation is simply defined by instantiating the generic fracture model at given time steps t_0.

4.1 Crack propagation model

Our approach consists in parameterizing the nodes $\mathcal{N}_i$ of the graph by a tuple $(t_i, \Delta t_i)$. The parameter t_i represents the time step at which a crack starts growing at node $\mathcal{N}_i$, whereas Δt_i denotes the amount of time needed for this crack to grow and reach its final size (Fig. 9).

In our implementation, the time steps can be automatically computed as follows. Let v denote the constant crack propagation velocity and τ the time needed for a cross section to start growing and achieve its final state. Recall that δ_{ij} denotes the distance between node $\mathcal{N}_i$ and node $\mathcal{N}_j$, and the time intervals are incrementally defined as

$$t_j = t_i + \frac{\delta_{ij}}{v} \qquad \Delta t_j = \tau$$

Time parameters t_i and Δt_i can be edited for every node of the graph so as to control the speed on propagation of the fracture on its different parts. In our system, we have implemented some higher-level control tools for editing the crack propagation process.

The formation of a whole subtree can be delayed (or anticipated) by incrementing (or decrementing) the growth-starting time steps t_i by a user-defined constant amount of time. In a similar way, the crack formation can be accelerated or slowed by increasing or decreasing the parameters Δt_i of a whole subtree.

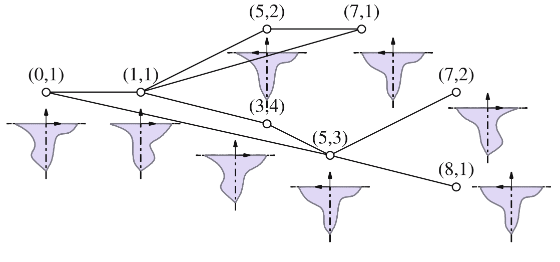

Fig. 9. Simplified representation of generic fracture propagation model

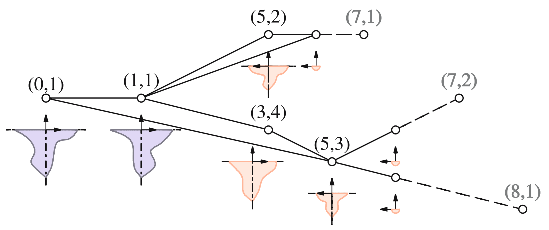

Fig. 10. An instance of generic fracture model in Fig. 9 with time step $t_0 = 6$

4.2 Instantiation scheme

The creation of a crack is performed by instantiating the generic fracture model at a given time step. The creation of $\mathcal{F}(t_0)$ is performed in two steps. First, we create the subgraph $\mathcal{G}(t_0)$ only keeping the nodes $\mathcal{N}_i$ whose starting time satisfies the condition $a_i > t_0$. The arcs $\mathcal{A}_{ij}$ that satisfy $t_0 > a_j$ are preserved, whereas the arcs $\mathcal{A}_{ij}$ that satisfy $a_i < t_0$ and $t_0 < a_j$ are cut (Fig. 10).

Thus, for every arc $\mathcal{A}_{ij}$ that satisfies the condition $a_i < t_0 < b_j$, we create a new leaf node $\mathcal{N}_k(t_0)$ in the subgraph $\mathcal{G}(t_0)$. The arc $\mathcal{A}_{ik}$ stores the same relative angle $\alpha_{ik} = \alpha_{ij}$ and the distance δ_{ik} is computed as follows:

$$\delta_{ik}(t_0) = \frac{t_0 - a_i}{a_j - a_i} \delta_{ij}$$

The profile curves $\mathcal{C}_i(t_0)$ at the nodes of $\mathcal{G}(t_0)$ are computed as follows. For every node $\mathcal{N}_i$:

1. If $t_0 < a_i$, then the propagation has not reached node $\mathcal{N}_i$; thus $\mathcal{N}_i(t_0)$ is not added to $\mathcal{G}(t_0)$.
2. If $a_i < t_0 < b_i$, then the propagation has reached node $\mathcal{N}_i$ but the formation of the crack at this point is not finished. We define the modified profile of the node by scaling it as follows:

$$\mathcal{C}_i(t_0) = \frac{t_0 - a_i}{b_i - a_i} \mathcal{C}_i$$

3. Otherwise, $t_0 > b_i$, so the node has stopped growing and $\mathcal{N}_i(t_0) = \mathcal{N}_i$.

The profile curves at the leaf nodes of $\mathcal{G}(t_0)$ are defined as single-point profile curves.

5 Results

We have applied our method to create cracks and fractures over a variety of objects of different materials including stone, wood, glass, ice, or clay. The corresponding images are shown throughout this paper.

Almost all the template crack patterns presented throughout this paper were created in less than 5 min by using our interactive editor. Thus, it took us only a few hours to create an atlas including different types of cracks and fractures for stone, wood, glass, ice, or clay.

All the final cracked models presented through out this paper were created in less than 5 min. This amount of time includes the selection of the crack pattern in the atlas and the application of the fracture model to the original object. The complex scene (Fig. 18), which includes many different models including several cracked stone walls (Fig. 17), a fractured wood bench (Fig. 1) and gate (Fig. 19), and broken glass light bulbs, was created in approximatively 2 h.

Eggshell. The crack pattern on the eggshell was created after a stone crack pattern. We modified its shape by inserting twice as many nodes and adding a random displacement so as to get a very irregular crack.

Mushroom. The crack pattern of the fairy ring model (Fig. 14) was created procedurally by generating radial and logitudinal cracks over the cap and the stem. The cracks follow the direction of the fibers of the mushroom.

Wine glass and vase. The fracture pattern on the wine glass (Fig. 12) was created by first editing a coarse pattern and progressively smoothing some of its parts while preserving some obtuse sharp angles. The crack profiles are very thin rectangular shapes whose width is less than 10 microns. The very thin geometry of the crack enabled us to capture the complex reflection and refraction phenomena and to produce very realistic lighting effects. The cracks in the glass vase (Fig. 13) were created using the same technique.

Pan glass. The original pan glass of the Mona Lisa painting was composed of 12 triangles. The star-shaped crack pattern on the pan glass covering the Mona Lisa painting (Fig. 20) was made of 6776 control nodes. Only 50 of theses were edited by hand; the others were automatically generated using our procedural smoothing technique that captured the curved geometric shapes of the crack. The generated crack pan glass is made of 95, 618 triangles. This example demonstrates that our method can correctly handle objects and fracture models with very different resolutions.

Wood. The cracks on the wood ramp (Fig. 17), the gate (Fig. 19), and the bench (Fig. 1) were created after the tex-

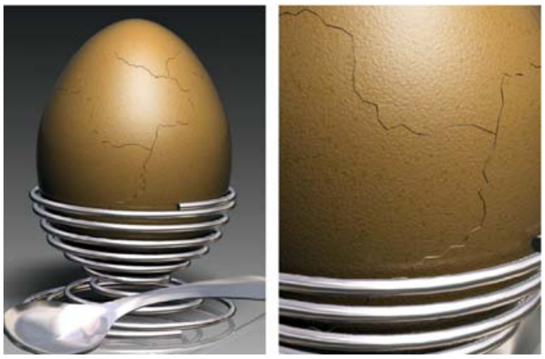

Fig. 11. Broken eggshell

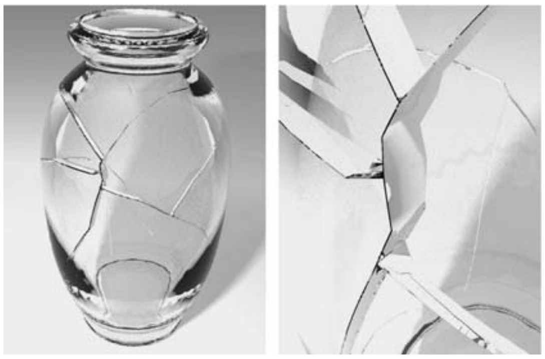

Fig. 13. Broken glass vase

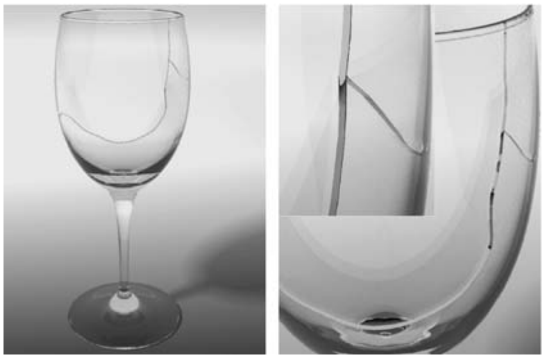

Fig. 12. Broken wine glass

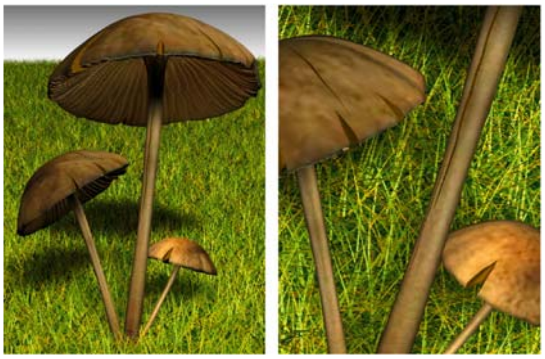

Fig. 14. Mushrooms with cracked steps and stems

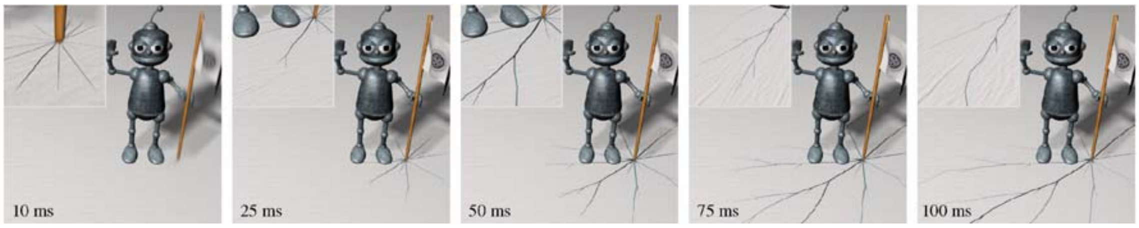

Fig. 15. Synthetic character (© Laurence Boissieux, INRIA) strikes and breaks icefield with staff

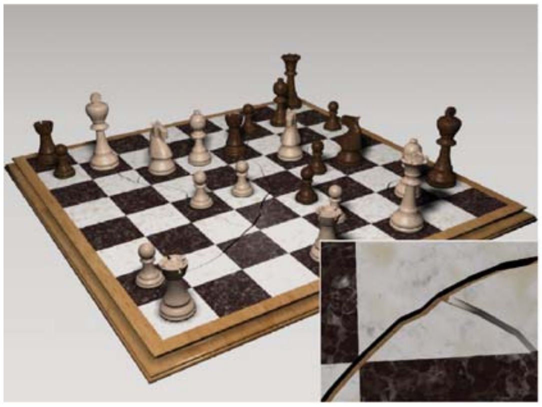

Fig. 16. Cracked chessboard

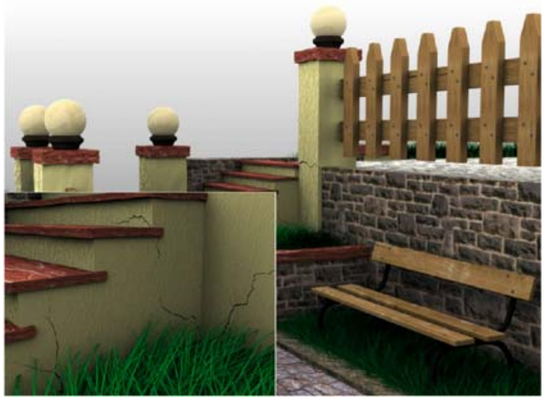

Fig. 18. Complex scene with many cracked objects

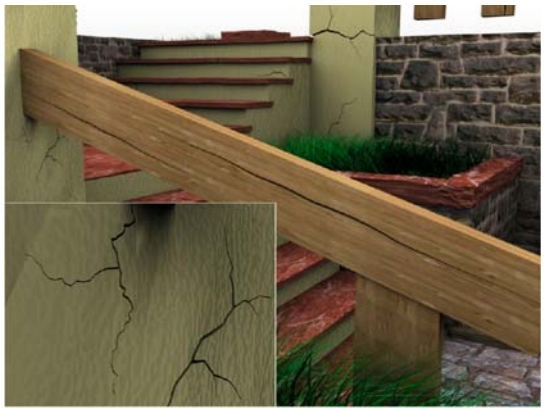

Fig. 17. Wood ramp with some fractures produced by dilatation of wood

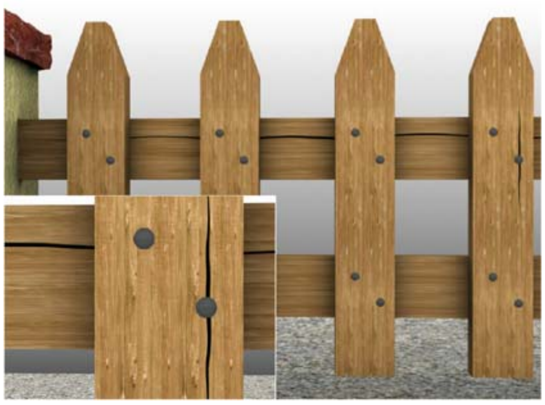

Fig. 19. Wood gate with cracks created by weathering of nails

ture pattern of the wood so that the fractures would follow the wood fibers.

Stone. The fracture patterns for the stone objects were created procedurally by perturbing an otherwise coarse crack pattern by inserting new nodes and modifying their location using small random displacements. Figure 17 shows a closeup of cracks produced by a wood ramp on a concrete wall.

The marble crack pattern of the chessboard (Fig. 16) was created procedurally by successively perturbing and smoothing an initial random coarse crack pattern.

Ice. Figure 15 shows a synthetic character standing on an icefield and striking it with a staff. The cracks appear at the impact point and quickly propagate over the surface of the icefield. The creation of the static crack pattern took 5 min. An extra 5 min were necessary to tune the parameterization of the nodes.

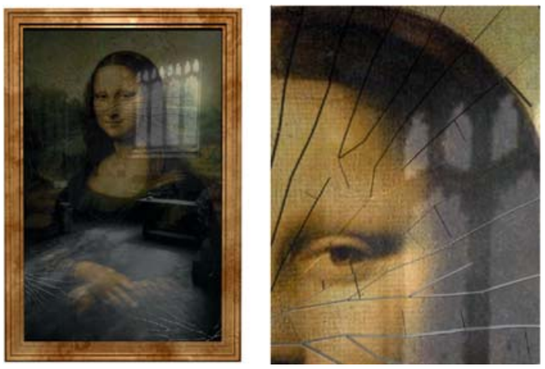

Fig. 20. Cracked pan glass covering Mona Lisa

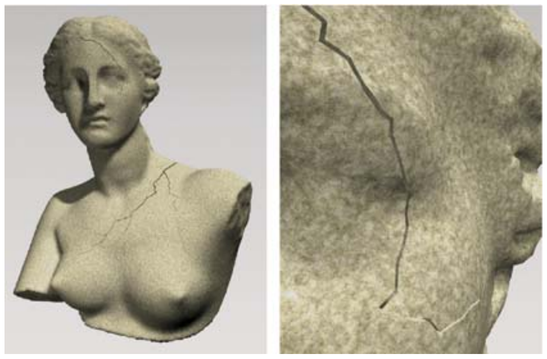

Fig. 21. Stone statue of Aphrodite with cracks

6 Conclusion

We have presented an efficient interactive technique for modeling a variety of cracks and fractures over objects of different materials. Our fracture model can handle very large or very thin fractures with a complex pattern and geometry easily. Our cracking algorithm can handle triangle meshes of different resolutions without producing artefacts.

We have developed a crack pattern editor that generates realistic complex models after real-world images. Fracture models can be archived into an atlas of shapes and interactively applied to any kind of object. The designer can control the shape of the cracks easily and with interactive feedback using our cracking editor.

In the near future, we plan to further investigate the creation of cracks with different levels of detail. Instead of creating the true geometry of the cracks, which would generate many triangles to capture the small details of the crack pattern, our method could directly generate a texture map and a corresponding bump map after the description of the crack pattern. This approach would enable us to create realistic approximations of cracks for models that only need a coarse representation with a low level of detail.

References

1. Bosch, C., Merillou, S., Pueyo, X., Ghazanfarpour, D.: Surface scratches: measuring, modeling and rendering. Visual Comput. **17**(1), 30–45 (2001)
2. Cutler, B., Dorsey, J., McMillan, L., Müller, M., Jagnow, R.: A procedural approach to authoring solid models. In: Proceedings of SIGGRAPH, pp. 302–311 (2002)
3. Desbenoit, B., Vanderhaghe, D., Galin, E., Grosjean, B.: Interactive modeling of mushrooms. In: Eurographics short papers, pp. 37–40 (2004)
4. Desbrun, M., Meyer, M., Schroder, P., Barr, A.H.: Implicit fairing of arbitrary meshes using diffusion and curvature flow. In: Proceedings of SIGGRAPH, pp. 317–324 (1999)
5. Dorsey, J., Pedersen, H.K., Hanrahan, P.: Flow and changes in appearance. In: Proceedings of SIGGRAPH, pp. 411–420 (1996)
6. Dorsey, J., Hanrahan, P.: Modeling and rendering of metallic patinas. In: Proceedings of SIGGRAPH, pp. 387–396 (1996)
7. Dorsey, J., Edelman, A., Legakis, J., VanJensen, H., Pedersen, H.K.: Modeling and rendering of weathered stone. In: Proceedings of SIGGRAPH, pp. 225–234 (1999)
8. Federl, P., Prusinkiewicz, P.: A texture model for cracked surfaces, with an application to tree bark. In: Proceedings of Western Computer Graphics Symposium, pp. 23–29 (1996)
9. Federl, P., Prusinkiewicz, P.: Finite elements models of fracture formation on growing surfaces. Lecture notes in computer science, **3037**, 138–145 (2004)
10. Gobron, S., Chiba, N.: Crack pattern simulation based on 3d surface cellular automata. Visual Comput. **17**(5), 287–309 (2001)
11. Hirota, K., Tanoue, Y., Kaneko, T.: Generation of crack patterns with a physical model. Visual Comput. **14**(3), 126–137 (1998)
12. Hirota, K., Tanoue, Y., Kaneko, T.: Simulation of three-dimensional cracks. Visual Comput. **16**(7), 371–378 (2000)
13. Jones, T., Durand, F., Desbrun, M.: Non-iterative, feature-preserving mesh smoothing. In: Proceedings of SIGGRAPH, pp. 943–949 (2003)
14. Lefebvre, S., Neyret, F.: Synthesizing bark. In: Eurographics Workshop on Rendering, pp. 105–116 (2002)
15. Martinet, A., Galin, E., Desbenoit, B., Akkouche, S.: Procedural modeling of cracks and fractures. In: Proceedings of Shape Modeling International, pp. 346–349 (2004)
16. Mazarak, O., Martins, C., Amanatides, J.: Animating exploding objects. In: Proceedings of Graphics Interface, pp. 211–218 (1999)
17. Merillou, S., Dischler, J.M., Ghazanfarpour, D.: A physically based model for rendering realistics scratches. Comput. Graph. Forum **23**(3), 361–370 (2004)
18. Müller, M., McMillan, L., Dorsey, J., Jagnow, R.: Real-time simulation of deformation and fracture of stiff materials. In: Eurographics Workshop on Animation and Simulation, pp. 113–124 (2001)
19. Neff, M., Fiume, E.: A visual model for blast waves and fracture. In: Proceedings of Graphics Interface, pp. 193–202 (1999)
20. Norton, A., Turk, G., Bacon, B., Gerth, J., Sweeney, P.: Animation of fracture by physical modeling. Visual Comput. **7**, 210–219 (1991)
21. O'Brien, J., Hodgins, J.: Graphical modeling and animation of brittle fracture. In: Proceedings of SIGGRAPH, pp. 137–146 (1999)
22. O'Brien, J., Bargteil, A., Hodgins, J.: Graphical modeling and animation of ductile fracture. ACM Trans. Graph. **21**(3), 291–294 (2002)

23. Smith, J., Witkin, A., Baraff, D.: Fast and controllable simulation of the shattering of brittle objects. In: Proceedings of Graphics Interface, pp. 27–34 (2000)

24. Terzopoulos, D., Fleischer, K.: Modeling inelastic deformation: viscoelasticity, plasticity, fracture. In: Proceedings of SIGGRAPH, pp. 269–278 (1988)

25. Wang, X., Wang, L., Liu, L., Hu, S., Guo, B.: Interactive modeling of tree bark. In: Proceedings of Pacific Graphics, pp. 83–91 (2003)

BRETT DESBENOIT Ph.D's Thesis.

ERIC GALIN is an Assistant Professor of Computer Science at the University Claude Bernard Lyon 1 (UCBL), France. A graduate from Ecole Centrale de Lyon, he received a PhD in Computer Science from UCBL in 1997.

SAMIR AKKOUCHE Professor