# Real-Time Simulation of Brittle Fracture using Modal Analysis

Loeïz Glondu, Maud Marchal, Georges Dumont

# Real-Time Simulation of Brittle Fracture using Modal Analysis

Loeiz Glondu , Maud Marchal and Georges Dumont

**Abstract**—We present a novel physically-based approach for simulating realistic brittle fracture of impacting bodies in real-time. Our method is mainly composed of two novel parts: (1) a fracture initiation method based on modal analysis, (2) a fast energy-based fracture propagation algorithm. We propose a way to compute the contact durations and the contact forces between stiff bodies to simulate the damped deformation wave that is responsible for fracture initiation. As a consequence, our method naturally takes into account the damping properties of the bodies as well as the contact properties to simulate the fracture. To obtain a complete fracture pipeline, we present an efficient way to generate the fragments and their geometric surfaces. These surfaces are sampled on the edges of the physical mesh, to visually represent the actual fracture surface computed. As shown in our results, the computation time performances and realism of our method are well-suited for physically-based interactive applications.

**Index Terms**—Physical simulation, brittle fracture, modal analysis.

## 1 Introduction

Launched by the early works of Griffith [Gri24] and Irwin one century ago, the study of fracture by physicists led to a new research field called Fracture Mechanics [And95]. The term "fracture" embeds many meanings in the literature, such as: *tearing* (progressive separation of the material due to tensile stress), *breaking* or *shattering* (explosive separation of a body into numerous fragments), *cutting* (controlled separation of the material), *cracking* (formation of cracks inside the body, without separating it), and *crumbling* (separation of the matter into small fragments usually caused by friction or deterioration). The adjective "brittle" commonly refers to fractures occurring between stiff bodies that undergo only small and elastic deformations during both crack opening and crack propagation (as opposed to *ductile* fracture).

Numerical simulations of brittle fracture based on lattice networks were introduced in the 80's [AS88]. Now, brittle fracture simulation is commonplace in the computer graphics community, and has numerous applications in medical simulation, computer animation or games industry. However, existing real-time approaches for simulating brittle fracture can still be improved for several reasons. (1) The damped deformation waves propagation are not simulated. At the location of impact, a local deformation occurs and is damped through the material. The properties of this wave have consequences on the fractures outcome (see *e.g.* Figure 4). Existing real-time approaches do not model this phenomenon. (2) The time steps used during contacts are not adaptive. Real-time methods use time steps that are not related with the vibrational frequencies of the stiff bodies, preventing the deformations of the bodies to be properly captured. (3) The visual fracture patterns use mesh elements boundaries. Existing real-time approaches rely on the elements boundaries of either a physical mesh or a specific visual mesh to represent the fracture paths. The fractures are not allowed to propagate in any direction or in a straight manner.

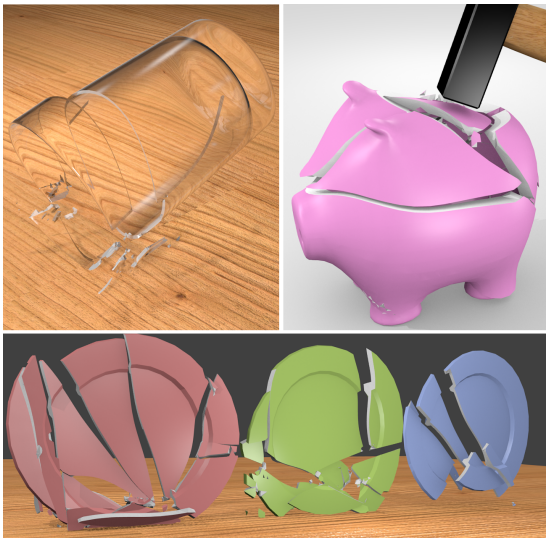

Fig. 1. Examples of brittle fracture simulations with our approach: (Left) a water glass breaks (with partial fractures) at the second bounce, (Right) a piggy bank smashed by a hammer, (Bottom) three plates dropped with different material properties.

*Loeiz Glondu, Maud Marchal and Georges Dumont are with IRISA/INRIA (Campus universitaire de Beaulieu, Rennes, France).*
*L. Glondu and G. Dumont are also with ENS Cachan, Antenne de Bretagne, France.*
*M. Marchal is also with INSA Rennes, France. E-mail: {loeiz.glondu, maud.marchal, georges.dumont}@irisa.fr*

We propose a new approach for simulating realistic physically-based brittle fracture of impacting rigid bodies, targeting interactive applications. Our method leverages previous work on modal analysis to efficiently simulate the deformations of stiff bodies, and to determine the starting points of fracture. As opposed to existing real-time methods, we adapt the time step depending on the materials and contacts properties, and we manage the fracture propagation and the fragments to generate a visual representation of the fracture that do not rely on elements boundaries of a mesh. However, we do not manage the recursive fracture of the generated fragments in this paper.

Our main contributions are:

- **A fracture initiation step based on modal analysis**. We propose a method based on modal analysis that takes into account the geometry of the impacted body, estimates contacts duration, and chooses appropriate time steps to simulate the deformations during contacts. Results show the impact of our simulation method on the fracture outcomes.
- **An efficient energy-driven algorithm for fracture propagation**. We present an efficient propagation algorithm that manipulates implicit surfaces to represent the fracture surfaces. The intersections between the fracture surfaces are handled robustly, and the implicit surfaces are sampled along the mesh elements edges, avoiding to rely on the elements boundaries.
- **A complete system for interactive applications**. We present an efficient way to generate fragments with a small memory footprint. Combined with our fracture initiation and fracture propagation, it forms a complete pipeline for realistic brittle fracture simulation that fits well with interactive application needs.

The paper is organized as follows: Section 2 sums up related work in fracture simulation. In Section 3, we present the main components of our approach. Section 4, 5 and 6 detail our fracturing method. Results and discussion are presented in Section 7.

## 2 Related Work

In this section, we survey the existing approaches for the simulation of brittle fracture. For clarity purposes, the section is divided in three parts: fracture initiation, fracture propagation and a focus on the existing real-time approaches.

### 2.1 Fracture Initiation

In the computer graphics community, first attempts to model fracture can be found in [TF88], where the authors used finite differences to simulate a 2-D mesh representing a tearing sheet of paper. The authors proposed to remove the link between the nodes of the mesh if their distance exceeds a threshold. Spring networks have also been used to model fractures [NTB+91], [DM95]. In [NTB+91], blocks of springs are removed at the same time to avoid strings of springs hanging. In [HTK98], the authors modeled surface cracks using layers of spring networks. More recently, [OH99] brought a new standard, proposing a rigorous simulation of deformation using the Finite Element Method (FEM). In order to determine whether a fracture should open, they defined a separation tensor derived from the deformation of the body. Later, their work was extended to model ductile fracture [OBH02]. A different approach consists in using a quasi-static analysis to study the stress state of the body as a consequence of an applied loading [MMDJ01]. When the maximal principal stress of an element exceeds a threshold, a fracture is initiated in this element. Quasi-static analysis leads to a rank deficient problem. This problem can be solved by anchoring points that are far from the fracture location [MMDJ01]. In [BHTF07], the quasi-static analysis is carried out thanks to a null space elimination technique, while [ZJ10] proposed a new specific solver. Another approach called cohesive zone model [SWB01], [EGGP02] proposes to consider cohesive forces between the elements of the physical mesh. Elements are separated when cohesive forces exceed a threshold (producing mesh dependent fracture patterns).

### 2.2 Fracture Propagation and Patterns

Purely graphical approaches have been proposed to model fracture for computer animation [NF99], [DGA05]. In [BHTF07], [ZJ10], the authors indirectly define the fracture surfaces thanks to Voronoi diagrams. Voronoi cell centroids are sampled with probabilities proportional to strain energy density, producing smaller pieces in regions of higher strain. In [ZJ10], the authors add an energy criterion to their method, limiting the sampling of Voronoi centroids if the energy of the indirect created surface exceeds the initial strain energy available. The energy test we use is similar to theirs, but in the context of propagating cracks rather that in a context of Voronoi-based patterns. Other approaches in computer graphics for computing crack propagation consist in iteratively propagating the crack along element boundaries [SWB01], [MMDJ01] or along arbitrary path through elements using remeshing techniques [OH99]. Remeshing produces fracture paths independent of the mesh, but involves an unpredictable growth of the number of elements. On the other side, modeling the fracture path with element boundaries produces visually not appealing fracture patterns. In order to avoid too many cracks to open and spurious branching, [SWB01] used a weakness mechanism around the separated elements to encourage the fracture to propagate where it

has already been opened. Meshless methods for brittle fracture have also been introduced by [PKA+05].

### 2.3 Real-Time Fracture Simulation

The first physically-based real-time system for fracture has been proposed in [MMDJ01], where bodies composed of a few number of elements breaking in real-time. Later, Müller *et. al.* presented another framework for real time fracture that leverages the computational gains of corotational formulation of FEM techniques [MG04]. The authors demonstrated their techniques for ductile fracture. However, using their method for brittle fracture would imply to use smaller time steps, that would compromise the performances. Recently, [PO09] proposed a complete solution targeting game industry, in which the graphical and physical meshes are dissociated through the use of so-called splinters. When fracture surfaces are defined along the mesh boundaries or when remeshing techniques are used, there is no need for a fragment generation step (the fragments are already defined by subsets of elements of the initial mesh, or cut elements). However, it is also possible to define fragments as subsets of the nodes of the initial body [MBF05], and to generate the geometric surface of fracture separately.

Existing real-time approaches do not use appropriate time steps to model the deformations of the stiff bodies during short contacts. However, the properties of the damped propagation waves from impact location have consequences on the simulation of brittle fracture. Also, current real-time methods rely on element mesh boundaries to generate the fracture geometry. This can be undesired if one wants the fracture to propagate in a straight way, and in any direction. In this paper, we propose a novel approach that uses adaptive time steps for each contact and simulates the damped deformation wave due to the contacts forces. We also present a fast propagation algorithm to simulate realistic fracture patterns in real-time.

## 3 Overview of the Fracture Simulation

An overview of our brittle fracture simulation algorithm is presented in Figure 2. This scheme stands for one body, and each dynamic body of our scene follows the same rules.

We use a third-party rigid body engine to simulate the bodies of our scene and perform collision detection. As soon as a contact is detected on a body, our fracture algorithm is performed on it. Our fracture algorithm can be decomposed into three parts. (1) A fracture initiation part, that simulates the deformations of the bodies during contacts, and determines where fracture are initiated (section 4). (2) A fracture propagation part, that computes the propagation of the fracture surfaces into the material (section 5). (3) A fragments generation part, that samples the fracture surfaces geometry, and computes the fragments (section 6).

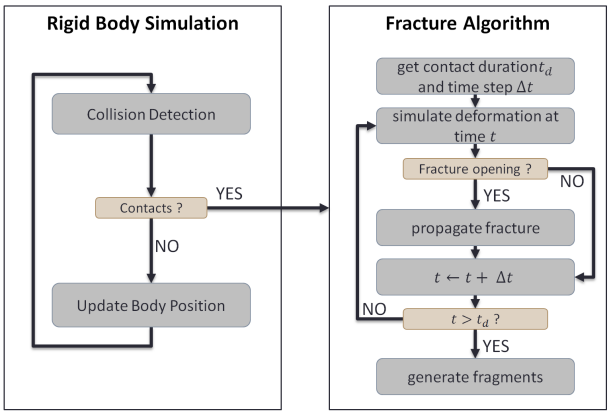


Fig. 2. Overview of our fracture algorithm for one body. At each contact event, the fracture algorithm is processed.

## 4 Fracture Initiation

Our fracture initiation method is based on modal analysis to estimate the contact properties and to deform the impacted bodies. Before the simulation, a modal analysis is performed on each body that can fracture, as presented in appendix (see also *e.g.* [PW89], [JP02], [OSG02], [HSO03] for more details on modal analysis from the computer graphics literature). Because the contact duration influences the fracture simulation (see Figure 3), we present a way to estimate it in section 4.1. We also choose an appropriate time step to capture the main deformations of the body due to the impact, as explained in section 4.2. Section 4.3 details the contact force model that we defined to simulate contact forces between stiff bodies. Finally, section 4.4 sums up the whole process.

### 4.1 Contact Duration

To build our contact duration estimation, we started from a formulation based on the Hertz's model of sphere-sphere contacts [Joh87]:

$$t_d = c. \left( \frac{m^2}{E^2 r} . \frac{1}{v_{rel}} \right)^{1/5} \quad (1)$$

This formulation relates the contact duration $t_d$ and the mass $m$, stiffness $E$, radius $r$ of the spheres, as well as their velocity of approach $v_{rel}$ at the time of impact ($c$ being a constant scalar). The stiffer the body, the shorter the contact duration, while the heavier the body, the longer the contact duration. This property is useful in the context of brittle fracture, since the stiffness and mass of the bodies should modify the fracture simulation outcomes. However,

the formulation in equation (1) is written for sphere-sphere impacts, and does not take into account the geometry of the body, nor the position of the impact on the body. Therefore, we propose an extension of this model based on modal analysis. We substitute the ratio $m^2/E^2r$ of equation (1) by the mass/elasticity ratio of the mode of deformation which is the most excited by the impact. We choose the most excited mode because this the one that will involve the greatest displacements during the impact, as shown in Figure 3. Equation (1) becomes:

$$t_d = c.\left(\left(\frac{2\pi}{Im(\omega_{max})}\right)^2.\frac{1}{v_{rel}}\right)^{1/5} \quad (2)$$

where $Im(\omega_{max})$ ($Im(x)$ is the imaginary part of the complex number $x$) is the natural frequency of the mode $max$, the most excited mode.

The frequencies $Im(\omega_i)$ of each mode depend only on the stiffness (*i.e.* the material elastic properties), the mass, and the geometry of the body, keeping the same properties as the first formulation of contact duration. Note that if the mode $i$ is critically damped, this approximation loses its sense. In practice, we select the most excited mode among those that are not critically damped. Also, since contacts involve at least two bodies, we compute contact duration estimation for each of the bodies involved in the contact, and retain the largest duration. Finally, if the body is colliding at several locations at the same time, we choose the contact that has the smallest duration. The effect of the contact duration on the simulation is highlighted in Figure 4.

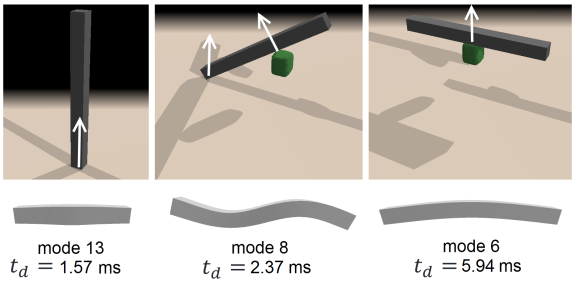


Fig. 3. Contact duration on a rod falling in different configurations. Depending on its initial position of the rod, the most excited mode is different, and so is the estimated contact duration.

### 4.2 Simulation Time Step

To determine an appropriate time step for the simulation of the deformations during the contact, we use the Nyquist-Shannon sampling theorem. However, instead of taking the highest frequency of all retained modes, we take the highest frequency $Im(\omega_{high})$ of the non-critically-damped mode among the modes that have been excited by the contact impulse. Thus, the time step $\Delta t$ used in our simulation is:

$$\Delta t = \frac{4\pi}{Im(\omega_{high})} \quad (3)$$

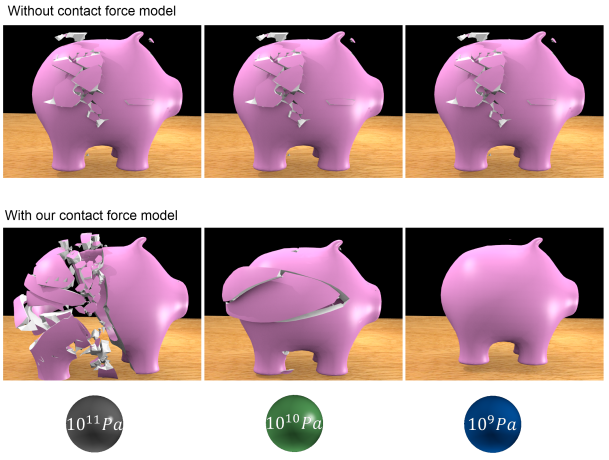


Fig. 4. A piggy bank hit by balls of different stiffness. Top: no contact force model is applied. We use a quasi-static approach to model a single deformation state of the piggy bank from the rigid body impulse. Bottom: the contact duration approximation of our contact force model leads to higher local stresses when the ball has a higher stiffness. With a softer ball, the contact will last longer and the energy of the ball will be damped without generating any fracture.

### 4.3 Contact Force Model

We use a sinusoidal function $f(t)$ to model contact forces:

$$f(t) = a.\sin(u.t) \quad (4)$$

where $a$ is the amplitude, $u$ a frequency, and $f(t)$ is the magnitude of the contact force. The frequency $u$ is computed with the contact duration (2) as $u = \pi/t_d$.

**Force amplitude** The rigid body simulator computes contact impulses that prevent bodies from interpenetrating. A contact impulse $\phi$ can be interpreted as the integral of a lasting contact force over time [Hah88]. In our case, we interpret it as the integral of the contact force of equation (4) over the duration of contact $t_d$:

$$\phi = \int_0^{t_d} f(t).dt = \int_0^{t_d} a.\sin(u.t) = a.\frac{1-\cos(u.t_d)}{u} \quad (5)$$

The amplitude $a$ of the contact force is deduced from this integral as $a = (\phi.u)/(1-\cos(u.t_d))$.

Figure 5 shows an example of the deformations of a slab due to a contact force.

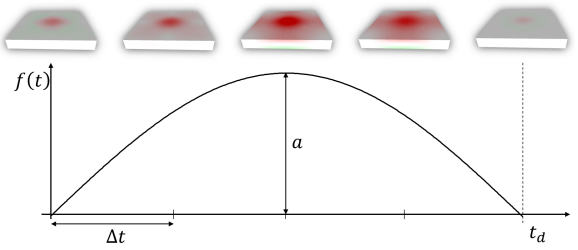


Fig. 5. Our contact force model applied on a stiff slab (color intensity is proportional to the maximum principal tensile stress). In order to model contact forces, a sinusoidal force $f(t)$ is applied at discrete times using a time step $\Delta t$ for a duration of contact $t_d$. The parameters $\Delta t$, $t_d$ and the amplitude $a$ are determined thanks to modal analysis and rigid body simulation.

### 4.4 Fracture criterion

At each single or multiple contact event from the rigid body simulator, we analytically compute the damped deformation wave that occurs due to the contact force, using modal analysis. From the displacement vector computed for a body, we compute the Green-Lagrange strain $\epsilon_e$ tensor for each element $e$, and deduce the stress $\sigma_e$ of the elements using a Hookean model of elasticity: $\sigma_e = C\epsilon_e$ ($C \in \mathbb{R}^{6\times6}$ being the linear constitutive law). If the maximum principal stress of an element exceeds a threshold $R_c$, a fracture is initiated (Rankine hypothesis [GS06]). To model the inhomogeneity of the material, we randomly sample weak elements into the bodies from a density of imperfection. Fractures are initiated at the location of the weak element which is the closest to the element that has the highest principal stress. The whole process of fracture initiation is summed up in Algorithm 1.

**Algorithm 1** Fracture initiation test for one body $\mathcal{B}$

$\mathcal{M} \leftarrow modal\_analysis(\mathcal{B})$ // appendix
$\mathbf{b} \leftarrow$ contact position
$\phi \leftarrow$ contact impulse magnitude
$t \leftarrow 0$
$t_d \leftarrow contact\_duration(\mathbf{b}, \mathcal{M}, \mathcal{B})$ // § 4.1
$\Delta t \leftarrow time\_step(\mathbf{b}, \mathcal{M}, \mathcal{B})$ // § 4.2
**for** $t < t_d$ **do**
  $t \leftarrow t + \Delta t$
  $f(t) \leftarrow$ contact force at $t$ // § 4.3
  $modal\_deformation(f(t), \mathcal{M}, \mathcal{B})$ // § 4.3
  **if** $\exists$ element $e$ with maximum principal tensile stress $> R_c$ **then**
    propagate fracture at $e$ // § 5
  **end if**
**end for**

## 5 Fracture Surface Propagation

Once a fracture has been initiated (section 4), it propagates through the body and eventually separates it. In the case of brittle fracture, the initial direction of the propagation is often deduced from the stress state of the material around the crack opening location. In stiff material, small deviations of the crack tips are observable. These small deviations are due to the local inhomogeneities and geometry of the material. However, there is currently no evident way to validate or measure the validity of the fracture patterns obtained. Therefore, we model the deviations of the crack using a noise function to produce visually good fracture surfaces. We first present in paragraph 5.1 how we model the fracture surfaces. In a second part (paragraph 5.2), we detail our propagation algorithm based on this surface definition, including robust collision handling between crack, and a stopping condition based on energy.

### 5.1 Fracture Surface Model

We propose to define the fracture surface using a general implicit surface $\mathcal{S}$:

$$\mathcal{S} = \{\mathbf{p} | s(\mathbf{p}) = 0\} \tag{6}$$

where $\mathbf{p} \in \mathbb{R}^3$. This definition allows a general and flexible way to model the fracture surfaces using mathematical expressions. In this paper, we used the following expression for $s(\mathbf{p})$ in (6):

$$s(\mathbf{p}) = bump\left(R^{-1}(\mathbf{p} - \mathbf{p}_0)\right) \tag{7}$$

where $R \in \mathbb{R}^{3\times3}$ is an orientation matrix that defines to global orientation of the surface, $\mathbf{p}_0$ defines an offset, and $bump(\mathbf{p})$ is defined as follow with a 2-D simplex noise function [Per02]:

$$bump(\mathbf{p}) = p_y - noise(p_x, p_z) \tag{8}$$

### 5.2 Propagation Algorithm

**Initial direction** The initial direction chosen by the crack to separate the material is the direction that maximizes the elastic energy dissipation [Cot65]. The directions of principal stress satisfy this condition. Therefore, we choose to open the fracture in the direction $\mathbf{d}$ of the maximum principal stress of the element $e_0$ on which the fracture starts. The exact location of the beginning of the fracture is the center $\mathbf{c}_0$ of the element $e_0$. Therefore, the parameters of the fracture surface equation (7) are:

$$\mathbf{p}_0 = \mathbf{c}_0 \tag{9}$$

$$R = (\mathbf{d}_{\perp1}\ \mathbf{d}\ \mathbf{d}_{\perp2}) \tag{10}$$

where $\mathbf{d}_{\perp1}$ and $\mathbf{d}_{\perp2}$ are two orthogonal vectors that are also orthogonal to $\mathbf{d}$.

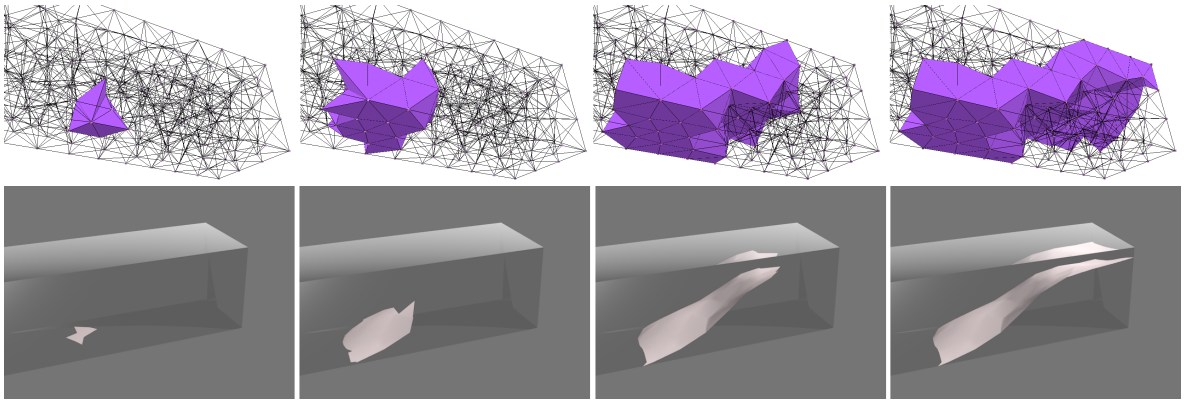

Fig. 6. Propagation of one crack in a coarse mesh. **Top line**: Representation of the current set $\mathcal{E}$ of elements cut by a fracture surface $\mathcal{S}$. Initially (top left), the set of element in composed of the element $e_0$ on which the fractured has been initiated. During the propagation, the neighbors of the set $\mathcal{E}$ that cross the surface $\mathcal{S}$ are added to $\mathcal{E}$ in a breadth-first search manner. **Bottom line**: representation of the actual fracture surface computed as the intersection between $\mathcal{S}$ and the volumes of $\mathcal{E}$. The surface is triangulated with the edges of the mesh elements (see section 6).

**Fracture Propagation** The key idea of our crack propagation algorithm is to use the implicit surface to visit the physical mesh elements. Starting from the initial element $e_0$, we visit the neighbors $neighbors(e_0)$ that are crossed by the implicit surface. An element is considered as crossed by the fracture surface if one of its node position $\mathbf{p}_i$ is on the negative side of the fracture surface ($s(\mathbf{p}_i) < 0$) and one of its node position $\mathbf{p}_j$ is on the positive side of the fracture surface, or on the fracture surface ($s(\mathbf{p}_j) >= 0$). We visit in a breadth-first fashion the neighbors of the crossed elements to form a set $\mathcal{E}$ of elements crossed by the fracture surface (see Figure 6).

**Fracture Stopping Criterion** We adopt a macroscopic understanding of the brittle fracture propagation, and propose an energy-based criterion to stop the crack propagation. We express the fracture energy $E_f$ (*i.e.* the amount of energy that has been used to propagate the crack) with a piecewise linear approximation:

$$E_f = \sum_{e \in \mathcal{E}} A_e.G_c \tag{11}$$

where $A_e$ is the area of the fracture surface that crosses element $e$. $G_c$ is the fracture toughness of the material, which expresses its resistance to fracture propagation. Similarly, we define $E_s$ as the strain energy available from the non-fractured state of the body using:

$$E_s = \sum_{e \in \mathcal{E}} A_e.\gamma.\eta_e \tag{12}$$

where $\eta_e$ is the strain energy density of element $e$, and $\gamma$ is a constant factor that links $E_f$ and $E_s$. The crack can propagate only if the available amount of energy permits it. When a new element $e_i$ is visited, it can be added to the set $\mathcal{E}$ of fractured elements only if $E_f + A_{e_i}.G_c < E_s + A_{e_i}.\gamma.\eta_{e_i}$, *i.e.* if the new sum of fracture energy needed is under the sum of energy available. Figure 7 shows the influence of the fracture toughness $G_c$ on the propagation of the fracture.

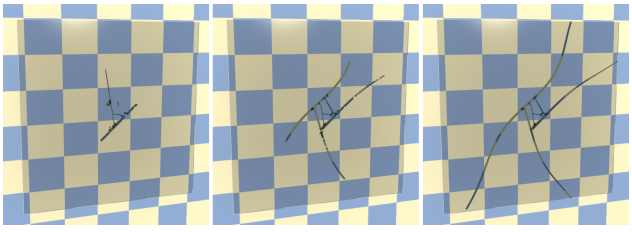

Fig. 7. A thin glass slab is hit by a ball at its center, with different fracture toughness $G_c$ (the fracture paths have been highlighted during the rendering). The left slab has a toughness $G_c$ of $150\,J.m^{-2}$, the middle slab has a toughness of $90\,J.m^{-2}$, while the right slab has a toughness of $60\,J.m^{-2}$.

**Collisions between the cracks** A newly visited element can not be added into the set $\mathcal{E}$ of cut elements if it is already marked as fractured. This simple rule allows to handle the collision between the arbitrarily complex implicit surfaces in an approximate manner using the physical mesh. Algorithm 2 sums up our fracture propagation algorithm.

## 6 Fragments Generation

This section details how we define and generate fragments in a first part (paragraphs 6.1 and 6.2). In a second part, we present how the triangular surface of each fragment is generated (paragraph 6.3). The recursive fracture of the generated fragments can not be handled in a straight-forward way. In the discussion Section 7.3, we propose three ways (two of them have

**Algorithm 2** Propagation of one fracture surface

1: $\mathcal{S}$ = surface defined by equation (7)
2: $\mathcal{E} = \{e_0\}$
3: $E_f = 0$
4: $E_s = 0$
5: **for all** $e \in \mathcal{E}$ **do**
6: $E_f = E_f + A_e.G_c$
7: $E_s = E_s + A_e.\gamma.\eta_e$
8: mark $e$ and its nodes as being crossed by current fracture
9: $\mathcal{E} = \mathcal{E} - \{e\}$
10: **for all** $n \in neighbors(e)$ **do**
11: **if** $n$ crossed by $surf(\mathcal{S})$ **and** $E_f < E_s$ **and** $!marked(n)$ **then**
12: $\mathcal{E} = \mathcal{E} \cup \{n\}$
13: **end if**
14: **end for**
15: **end for**

been tested) to approximate the deformations of the fragments and to allow their recursive fracture.

### 6.1 Fragment Model

Defining how the fragments are represented is a determinant choice in a fracture algorithm. Indeed, it has consequences on how they are generated, how they can be fractured again, and how they can be visually displayed. Since the fracture surfaces computed pass through the elements of the physical mesh, we consider the material around nodes, and define the fragment as a subset of nodes of the initial mesh [MBF05]. Each node stores a unique region identifier (region ID) representing the fragment to which it belongs. Initially, all the nodes have the same region ID which is the ID of the non fractured body region. The region ID of the nodes are modified during the fragments generation process, and each group of node that has the same region ID represents a single fragment. We leverage the node marking of the fracture propagation process to generate fragments (line 8 of algorithm 2).

### 6.2 Separation of the Initial Mesh

From the fracture propagation step, we obtain sets of fractured element $\mathcal{E}$. We use these sets of elements to compute different regions of the materials that represent the generated fragments. During the fracture propagation, the visited elements have their nodes marked using the following convention. Each node that is on the positive side of the implicit fracture surface is marked $(h, 1)$, and $(h, 0)$ if it is on the negative side ($h$ being a fracture surface identifier, see Figure 8). Two nodes belong to the same fragment if they have the same mark, or if at least one of them is not marked (but belongs to the same element). Therefore, we define the fragments by flood filling: we start from one node, and recursively add its neighbors if they satisfy the previous condition (see Figure 9.

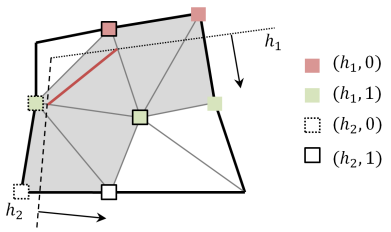


Fig. 8. Node marking process on a body crossed by two fracture surfaces $h_1$ and $h_2$. Each node of each fractured element (gray) is marked using a positive/negative test with respect to the fracture surfaces. Note that even if the top left element is crossed by the implicit surface, it is not detected as such. The red line represents the shortcut that will be taken by the triangulation algorithm.

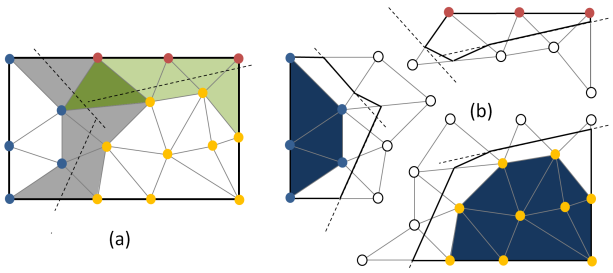


Fig. 9. Generation of fragments using marked elements. The block has been opened by two fractures (gray and green) as in Figure 6.d. The dark green element is marked with both fracture surfaces. **(a)**: The fragments are defined by flood filling and using nodes ID. **(b)**: Each region of nodes with the same ID forms a fragment (depicted by colored points). White points are nodes that do not represent material for a fragment. The thick dark contours represent the fracture surfaces, sampled on edges of the white elements.

### 6.3 Triangulation of the Fragment Surfaces

We sample the fracture surfaces with the edges of the physical mesh, as proposed in [MBF05]. Each edge stores the position of one cut point, that represents the intersection between the edge and the fracture surface (see Figure 10).

To generate the surface triangles, we visit all the elements marked as crossed by a fracture surface during the fracture propagation process. For each element, we store in a list $\mathcal{L}$ the region ID of each of its node. Then, for each distinct value $r \in \mathcal{L}$, we generate the appropriate surface triangles as depicted in Figure 11. For tetrahedral elements, different cases may arise. Either 1, 2 or 3 nodes of the element have a region ID equal to $r$. Repeating the process shown

in Figure 11 for each region $r \in \mathcal{L}$ leads to generate all the faces of the fracture surface.

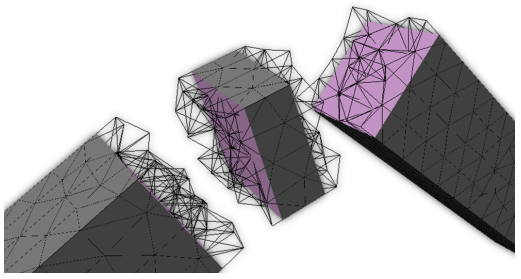

Fig. 10. Surface mesh and physical mesh. The colored section represent a fracture that cut the body straight through the physical mesh. Although the visual mesh represents with fidelity the fracture surfaces computed, the underlying physical mesh (in wire frame) is not updated.

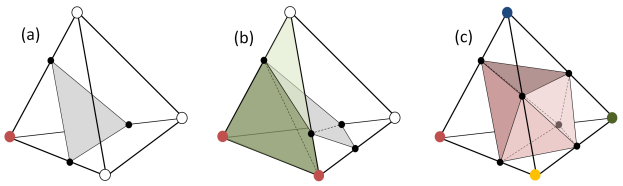


Fig. 11. Surface triangulation cases. The red nodes are nodes with region ID $r$, while white nodes have a different region ID. Black dots represent the cut position of the fracture surfaces on the edges. **(a)**: Only one node has a region ID equal to $r$, a single triangle is generated (gray). **(b)**: Two nodes have a region ID equal to $r$, a quad is generated. If the green face is a surface triangle, it is cut and here gives a quad represented by the dark green part. **(c)**: If a tetrahedron has all its edges cut, the center part of the tetrahedron (red) is not associated to any fragment. In that case, the central shape generates a small fragment by itself.

## 7 Results and Discussion

### 7.1 Computation time performances

**Configuration** Our fracture simulation method has been implemented in C++ on a laptop with 4 GB of RAM, and an Intel®Core™2 Extreme (2.3 GHz, we only use one thread). Our GPU implementation has been tested on an NVidia Quadro FX 3700M. We tested our method with Havok Physics [Hav] and NVidia PhysX [Phy] for the rigid body simulation [GMD10].

**GPU Implementation** Thanks to modal analysis, the fracture initiation step can be fully parallelized [CWL06]. We implemented a parallel version of this step, using GPU hardware to accelerate the computations. All modes of deformations are transferred once onto the GPU global memory, avoiding expensive GPU/CPU memory transfers. In a first sub-step, a list of deformation states (the successive deformations during time of contact) is generated using one thread per degree of freedom of the initial mesh. The deformation states are stored on the GPU global memory, and never transferred to CPU memory. Then, these states are used in a second sub-step to compute a list of the maximum principal stresses of each element using one thread per element. The lists of principal stresses are finally transferred back to CPU memory, and exploited to apply fracture criterion and initiate fractures. Our parallel implementation of the fracture initiation step gives up to $14$ times speed up on our configuration.

**Computation Time Results** Table 1 summarizes test cases parameters and results. The fracture test initiation scales with the mesh complexity thanks to the introduction of the density of weak points. Two meshes of different resolution but with the same volume and the same density of weak points will have similar computation time (see last line of Table 1). The bottleneck of our method is the fracture initiation phase (about 75% of the computation time), where the deformations are computed to initiate fracture. The complexity of our fracture propagation algorithm is linear in the number of elements of the physical mesh that are crossed by the fracture surface. The complexity of the fragment generation algorithm is linear in the number of nodes of the mesh to be separated. The linear complexity of our global system enables us to fracture bodies composed of more than $175$K elements in about $50$ milliseconds.

### 7.2 Tests and Scenarios

We demonstrate the main features of our approach through simple scenarios.

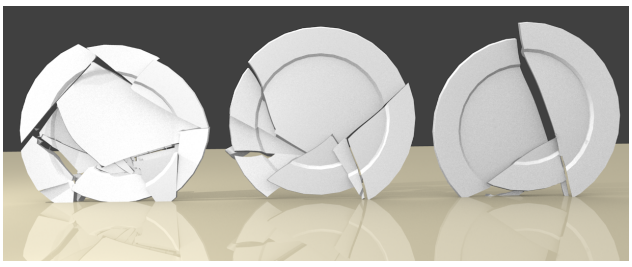

Fig. 12. Effect of damping on the fracture simulation. Our contact force model with modal analysis enables to model the inertia and damping of the plate. The left plate has the lowest damping value $\alpha = 0$, letting high frequency modes to propagate and generate many small fragments. The right plate has the highest damping value $\alpha = 10^{-6}$, generating less fragments.

Compared to previous real-time approaches, we simulate damping and contact properties through our contact force model and modal analysis. These phenomena have consequences on brittle fracture, as

| Body | Figure | Complexity | | Physical parameters | | | | | Timings | | | (ms) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | nodes | tets | $\alpha$ | $\beta$ | $E(Pa)$ | $G_c(J/m^2)$ | $R_c$ | initiation | propagation | fragments | total |
| glass | 1 | 7912 | 24410 | $1.10^{-9}$ | 5 | $200.10^9$ | 150 | $4.10^9$ | 34 / 3.5 | 5.1 | 1.3 | 9.9 |
| piggy bank | 1 | 5992 | 20777 | $1.10^{-8}$ | 3 | $200.10^9$ | 500 | $5.10^8$ | 66 / 8.3 | 6.6 | 6 | 20.9 |
| piggy bank | 4 | 5992 | 20777 | $1.10^{-8}$ | 3 | $200.10^9$ | 500 | $5.10^8$ | 150 / 15 | 12.3 | 6.4 | 33.7 |
| slab low | - | 644 | 1740 | $1.10^{-10}$ | 5 | $100.10^9$ | 100 | $10^9$ | 12 / 1.7 | 0.8 | 0.6 | 3.1 |
| slab moderate | - | 2761 | 8190 | $1.10^{-10}$ | 5 | $100.10^9$ | 100 | $10^9$ | 17 / 3.9 | 2.4 | 0.37 | 6.67 |
| slab high | 7 | 12017 | 39901 | $1.10^{-10}$ | 5 | $100.10^9$ | $60-150$ | $10^9$ | 25 / 7 | 6 | 17 | 30 |
| slab v. high | - | 50476 | 175095 | $1.10^{-10}$ | 10 | $100.10^9$ | 100 | $10^9$ | 26 / 7 | 25 | 21 | 53 |
| bunny | 13 | 5089 | 18767 | $2.10^{-9}$ | 5 | $1.10^9$ | 100 | $5.10^8$ | 42 / 3.2 | 6 | 7 | 16.2 |

TABLE 1

Parameters and timings used for our tests. The values after the slash in the initiation timings column are the timings obtained with GPU accelerations. Total times are computed considering GPU timings. The values $\alpha$ and $\beta$ are the Rayleigh damping coefficients, and the number of retained modes is one hundred (see appendix).

illustrated in Figure 4 (the stiffness of the projectile changes the contact properties) and in Figure 12 (different damping values lead to different fracture paths).

As shown in Figure 7, the fractures can propagate in any direction, and the fracture patterns are not guided by the elements boundaries of any mesh. Moreover, our method can model physically-based partial internal fractures as shown in Figures 1 and 7. Finally, our algorithms are valid for convex, concave, thin or thick bodies. Figure 13 shows the propagation of the fracture into a filled bunny broken at interactive rate.

Fig. 13. Propagation of the fracture into a thick filled body. Front, back and side view of a filled chocolate Stanford bunny thrown on a wall.

### 7.3 Discussion

**Limitations and perspectives for modal analysis** An hypothesis of our work is that we keep the initial modal analysis to simulate the deformation of the body, even if one or more fractures have started to open. We believe that this hypothesis is reasonable for several reasons. First, if one is interested only on a fracture test (*i.e.* checking whether the material will break or not), this hypothesis has no importance. Also, the deformations propagate during cracking, but it is not clear how. The solution we propose is a good trade-off between the realism and the computation cost of the simulation.

An other issue of our model is that modal analysis is not well suited to simulate local deformations on big structures that have many branching parts. A solution would be to use domain decomposition [BZ11] to separate the initial body into smaller domains, and keep the advantages of speed and parallelization. Moreover, modal analysis can be extended in several ways (using *e.g.* modal derivatives [Bar07] or [HTZ+10]) to handle larger deformations. We plan to include this work in our framework to extend our method to ductile fracture, or fracture of softer bodies.

**Recursive fracture for the fragments** An interesting property of our method is that the fracture propagation and the fragment generation algorithms can recursively be applied on fragments. However, the fracture initiation step cannot performed since the modal deformation basis cannot be pre-computed for the fragments. One possibility to extend this method is to use modal proxies (or modal impostures) as suggested in [PW89] and exploited in [ZJ10] for sound generation. The main principle is to find a shape (proxy) in a database (for which modal analysis has already been performed) that fits well with a fragment, scale it and its modes, and link the nodes of the fragment to the elements of the proxy. Results combining this approach and our fracturing method can be found in [GLMD11]. Another possibility is to use the same deformation modes to generate the deformation on the fragments. When an impulse (computed with the mass of the fragment) is detected on a fragment, the modal DOFs are set as if the whole body were impacted, but only the elements composing the fragment are checked for fracture. This solution has no physical justification, but is acceptable if the visual effect only is desired (please see the accompanying video for a result exploiting this technique). Finally, another solution would be to do a classical FEM simulation on the fragments. We already have the stiffness matrices for each element, and a quasi-static equilibrium can be computed as in [MMDJ01] to initiate the fractures, while the propagation and fragment generation step can be applied recursively.

**Treating resting contacts** In our approach, we are also able to treat resting contacts cases with a quasi-static approach solved with modal analysis. Indeed,

after the modal analysis treatment, we compute a basis that diagonalizes the stiffness matrix. Therefore, computing the reduced deformation that solves the quasi-static equilibrium is computationally cheap. This provides an efficient and plausible test for resting contact cases.

**A Framework for Brittle Fracture Simulation** We developed a complementary framework for fracture brittle simulation, in order to evaluate and compare brittle fracture simulation methods. It embeds features for quick set-up of various scenarios. It also provides tools for real-time visualizations of the influences of the different material parameters on the deformations of the bodies and their potential fractures. Finally, the computation time performances of our framework allow a haptic display of the brittle fracture.

## 8 Conclusion

In this paper, we presented a physically-based method for simulating brittle fracture in real-time. The first contribution of our method is the modeling of the contacts properties through a contact force model. This contact force model combined with modal analysis enables to efficiently simulate the deformations that lead to fracture initiation. The second main contribution concerns the modeling of the fracture propagation thanks to a new energy-driven algorithm. Finally, the third contribution handles fragments generation, and geometric fracture surface sampling.

Our method is the first real-time brittle fracture simulation that uses time step related to the vibrational frequencies of the bodies. It is also the first real-time approach that simulates the damped deformation wave of the bodies during contacts, and that produces a visual geometry of the fracture surfaces that do not rely on mesh elements boundaries. Our results demonstrate the robustness of our approach, but also its physical plausibility through the influences of the fracture parameters and the elastic properties on the fractured bodies. Finally, our method opens exciting perspectives for a new range of interactive applications that need realistic and physically-based brittle fracture simulation.

## References


[And95] T. L. Anderson. *Fracture Mechanics: Fundamentals and Applications*. CRC Press, 1995.

[AS88] S. Arbabi and M. Sahimi. Elastic properties of three-dimensional percolation networks with stretching and bond-bending forces. *Physical Review*, 38(10):7173–7176, 1988.

[Bar07] J. Barbič. *Real-time reduced large-deformation models and distributed contact for computer graphics and haptics*. PhD thesis, 2007. AAI3279452.

[BHTF07] Z. Bao, J. M. Hong, J. Teran, and R. Fedkiw. Fracturing rigid materials. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):370–378, 2007.

[BZ11] J. Barbič and Y. Zhao. Real-time large-deformation substructuring. *ACM Transactions on Graphics*, 30(4):91:1–91:7, 2011.

[Cot65] B. Cotterell. On brittle fracture paths. *International Journal of Fracture*, 1(2):96–103, 1965.

[CWL06] Y. Che, J. Wang, and X. Liang. Real-time deformation using modal analysis on graphics hardware. In *Proceedings of GRAPHITE*, GRAPHITE '06, pages 173–176. ACM, 2006.

[DGA05] B. Desbenoit, E. Galin, and S. Akkouche. Modeling cracks and fractures. *The Visual Computer*, 21(8-10):717–726, 2005.

[DM95] F. Donzé and S.A. Magnier. Formulation of a 3-d numerical model of brittle behaviour. *Geophysical Journal International*, 122(3):790–802, 1995.

[EGGP02] M. Elices, GV Guinea, J. Gomez, and J. Planas. The cohesive zone model: advantages, limitations and challenges. *Engineering Fracture Mechanics*, 69(2):137–163, 2002.

[GLMD11] L. Glondu, B. Legouis, M. Marchal, and G. Dumont. Precomputed shape database for real-time physically-based simulation. In *Proceedings of VRIPHYS*, 2011.

[GMD10] L. Glondu, M. Marchal, and G. Dumont. Evaluation of physical simulation libraries for haptic rendering of contacts between rigid bodies. In *Proceedings of ASME World Conference on Innovative Virtual Reality*, 2010.

[Gri24] A. A. Griffith. The theory of rupture. *First International Congress of Applied Mechanics*, 1924.

[GS06] D. Gross and T. Seelig. *Fracture mechanics: with an introduction to micromechanics*. Springer Verlag, 2006.

[Hah88] J. K. Hahn. Realistic animation of rigid bodies. In *Proceedings of SIGGRAPH*, pages 299–308, 1988.

[Hav] Havok. www.havok.com.

[HSO03] K. K. Hauser, C. Shen, and J. F. O'Brien. Interactive deformation using modal analysis with constraints. In *Proceedings of Graphics Interface*, pages 247–256. CIPS, Canadian Human-Computer Commnication Society, June 2003.

[HTK98] K. Hirota, Y. Tanoue, and T. Kaneko. Generation of crack patterns with a physical model. *The Visual Computer*, 14:126–137, 1998.

[HTZ+10] J. Huang, Y. Tong, K. Zhou, Hu. Bao, and M. Desbrun. Interactive shape interpolation through controllable dynamic deformation. *IEEE Transactions on Visualization and Computer Graphics*, 99(PrePrints):1–8, 2010.

[Joh87] K.L. Johnson. *Contact mechanics*. Cambridge Univ Pr, 1987.

[JP02] D. L. James and D. K. Pai. Dyrt: dynamic response textures for real time deformation simulation with graphics hardware. *ACM Trans. on Graphics*, 21:582–585, July 2002.

[MBF05] N. Molino, Z. Bao, and R. Fedkiw. A virtual node algorithm for changing mesh topology during simulation. In *Proceedings of SIGGRAPH*, page 4, 2005.

[MG04] M. Müller and M. Gross. Interactive virtual materials. In *Proceedings of Graphics Interface*, pages 239–246, 2004.

[MMDJ01] M. Müller, L. McMillan, J. Dorsey, and R. Jagnow. Real-time simulation of deformation and fracture of stiff materials. In *Proceedings of the Eurographic workshop on Computer animation and simulation*, pages 113–124, 2001.

[NF99] M. Neff and E. Fiume. A visual model for blast waves and fracture. In *Proceedings of Graphics interface*, pages 193–202. Morgan Kaufmann Publishers Inc., 1999.

[NTB+91] A. Norton, G. Turk, B. Bacon, J. Gerth, and P. Sweeney. Animation of fracture by physical modeling. *Visual Computer*, 7(4):21–219, 1991.

[OBH02] J. F. O'Brien, A. W. Bargteil, and J. K. Hodgins. Graphical modeling and animation of ductile fracture. *ACM Trans. on Graphics*, 21(3):291–294, 2002.

[OH99] J. F. O'Brien and J. K. Hodgins. Graphical modeling and animation of brittle fracture. In *Proceedings of SIGGRGAPH*, pages 137–146, 1999.

[OSG02] J. F. O'Brien, C. Shen, and C. M. Gatchalian. Synthesizing sounds from rigid-body simulations. In *Proceedings of SIGGRAPH*, pages 175–181. ACM Press, July 2002.

[Per02] K. Perlin. Improving noise. *ACM Transactions on Graphics*, 21:681–682, July 2002.

[Phy] PhysX. www.nvidia.com/object/physx_new.html.

[PKA+05] M. Pauly, R. Keiserand, B. Adams, P. Dutré, M. Gross, and L. J. Guibas. Meshless animation of fracturing solids. *ACM Trans. on Graphics*, 24:957–964, 2005.

[PO09] E. G. Parker and J. F. O'Brien. Real-time deformation and fracture in a game environment. In *Proceedings of SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 156–166, 2009.

[PW89] A. Pentland and J. Williams. Good vibrations: modal dynamics for graphics and animation. *ACM SIGGRAPH Computer Graphics*, 23(3):207–214, 1989.

[SWB01] J. Smith, A. Witkin, and D. Baraff. Fast and controllable simulation of the shattering of brittle objects. *Computer Graphics Forum*, 20:81–91, 2001.

[TF88] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscolelasticity, plasticity, fracture. *SIGGRAPH 88 Conference Proceedings*, 22:287–296, 1988.

[ZJ10] C. Zheng and D. L. James. Rigid-body fracture sound with precomputed soundbanks. In *Proceedings of SIGGRAPH*, volume 29, July 2010.

**Loeiz Glondu** is a PhD student at the École Normale Supérieure de Cachan high school in Rennes (France). His main research interests include physically-based simulation, haptic rendering, and virtual reality.

**Maud Marchal** is an Associate Professor in the Computer Science Department at INSA (Engineer school) in Rennes, France. She received her PhD in Computer Science from the University Joseph Fourier in Grenoble, France in 2006. Her main research interests include physical simulation, biomechanics, haptic interfaces, 3D interaction and virtual reality.

**Georges Dumont** is an Associate Professor in Mechanical Sciences at Brittany Antenna of cole Normale Suprieure de Cachan in Rennes since 1994. He is the head of VR4i, Virtual reality for improved innovative immersive interaction, research Team common to IRISA and INRIA and Head of "Media and Interaction" department at IRISA. He received his PhD in computer science from Rennes 1 University in 1990 and his habilitation in mechanical science in 2005. His main research interests include physical simulation, mechanics, biomechanics, haptic rendering, interactive collaboration and virtual reality.