Simulating Landslides for Natural Disaster Prevention

Jean-Dominique Gascuel, Marie-Paule Cani-Gascuel,
Mathieu Desbrun[†]
iMAGIS[‡]-GRAVIR UMR C5527 / IMAG
BP 53, F-38041 Grenoble cedex 09, France

Eric Leroi, Carola Mirgon
Bureau de Recherches Géologiques et Minières,
Direction de la Recherche,
117 avenue de Luminy, BP 167, 13276 Marseille, France

Contact: `Jean-Dominique.Gascuel@imag.fr`

Abstract

The simulation of landslide hazards is a key point in the prevention of natural disasters, since it enables to compute risk maps and helps to design protection works. We present a 3D simulator that handles both rock-falls and mud-flows. The terrain model is built from geological and vegetation maps, superimposed on a DEM. Since the exact elevation of the terrain is unknown at the rock's scale, the simulator uses a series of stochastic simulations, where low scale geometry is slightly randomized at each impact, to compute an envelop of risk areas. Computations are optimized using an implicit formulation of surfaces and a space-time adaptive algorithm for animating the particle system that represents the mud flow.

1 Introduction

Computing trajectories of rocky blocks or viscous mud falling or flowing down slope is a preoccupation for many engineering firms that have to establish safety perimeters, or must design protection works. In the area of geophysics, only a few simulation codes are available for such work and even rarer are those that tackle the three dimensional aspect of simulation. On the other hand, Computer Graphics "physically-based animation" systems are now able to simulate a vast range of phenomena, from rigid bodies to viscous liquids interacting with a dynamic environment.

This paper presents a landslide simulator designed in collaboration with geophysicists. It performs dynamic simulations of rock-falls, rock-slides, mud-flows and other geophysic phenomena based upon the geomechanical characteristics of the ground (given by stress-deformation curves and friction coefficients). Computations are made more efficient using local adaptation of the levels of detail for ground model, implicit

[†]Currently at: Department of Computer Science, Caltech, USA

[‡]iMAGIS is a joint project of CNRS, INRIA, Institut National Polytechnique de Grenoble and Université Joseph Fourier.

representation of surfaces, along with an adaptive algorithm for simulating smoothed particles that sample the mud-flow. The simulator has already been used successfully for an industrial application.

1.1 Background

Performing landslides simulations for preventing natural disasters is a difficult task. One of the main reason is the lack of precision on the data: the available resolution for terrain elevation is usually too low compared with the size of the rocks falling down slope, leading to important imprecisions on bouncing directions ; local vegetation should be taken into account since it plays an important part on friction coefficients at each impact; moreover, the exact characteristics of the rocks or the mud that are going to flow down slope are unknown as well. A solution is to perform a large number of slightly different simulations, which stochastic changes of parameters at each impact, and use the envelop of all the computed trajectories to delimit risk areas. However, this approach yields drastic constraints on the efficiency of simulations.

As a result, computer simulations have not been used much in the area of geophysics. Most existing simulators (eg. [8, 16, 12]) are based on 2D profiles of terrains instead of 3D elevation models. They compute trajectories of spherical blocks (i.e., circles since computations are performed in 2D) for which translational and rotational velocities after each bounce are computed by multiplying incident velocities by empirical "collision parameters", that are calibrated using a least square fitting over measured real cases. Computations can be achieved quite efficiently, but the results really suffer from a lack of precision: mechanical properties of materials are only coarsely approximated, and no stress-deformation law is used for instance. Moreover, since computations are only made on a set of 2D profiles, results (ie. envelops of risk areas) may be wrong compared to what would happen in the real 3-dimensional environment, depending on the specific geometry of the terrain.

To the authors knowledge, there has been no work specifically devoted to landslides simulations from the Computer Graphics community. However, models for soft grounds interacting with dynamic objects such as vehicles or human bodies have already been designed [3, 18]. In the first of these works, the large-scale plastic (non-linear) deformations of the ground are modeled by a coarse array of masses that move in 3D, linked together and to the under-ground by visco-plastic interactions. The small-scale linear deformations are modeled by a finer array of masses moving vertically, according to the "pin-screen" paradigm. Colliding objects interact with both grids. This results into very realistic images of footprints left by vehicle wheels, that include small bulges around compressed areas. In [18], the same visual effect is produced, under the hypothesis of constant volume deformations, by transporting to neighboring regions ground matter inter-penetrating with the feet of synthetic runners.

As will be shown below, our choices for the ground model are quite different, since our aim is not to produce nice footprints left by rocks falling down slope, but rather to provide an efficient simulation of rock trajectories from ground stress-deformation laws provided by geophysicists. In our case, only ground compression under collision is computed, since it affects the subsequent motion of the falling rock.

1.2 Overview

This paper presents a landslide simulator that performs 3D dynamic simulations of rock-falls and mud-flows. Based on both geo-mechanical data and physically-based animation algorithms, it handles various models that interact together during simulations such as elasto-plastic material for the ground, rigid solid for the rocks, and viscous liquid for the mud. Moreover, it tackles the problem of computational efficiency :

- Only regions of the ground where impact may take place at the next time step are simulated. A local re-sampling provides a representation of the ground at a smaller scale in these regions of interest.
- Implicit formulations of surfaces are used to accelerate collision detection and to compute ground deformations.
- Mud-flows are computed using an adaptive simulation, where the number and size of particles that sample the mud are adapted over time according to the occurring deformations (many particles are used in areas that deform much, while only a few of them is used in stable regions).

The remainder of this paper develops as follows: Section 2 explains how we process geophysics data for building the 3D model of the terrain. Computation of trajectories for rocks falling or sliding down slopes is detailed in Section 3. Muds-flows simulation is described in Section 4. Section 5 concludes and discusses work in progress.

2 Modeling geological data

This section introduces the geological concepts used throughout this paper. They differ somehow from what is usually found in the Computer Graphics literature, since we must share a common language and understanding with the geophysics community.

2.1 Geological data

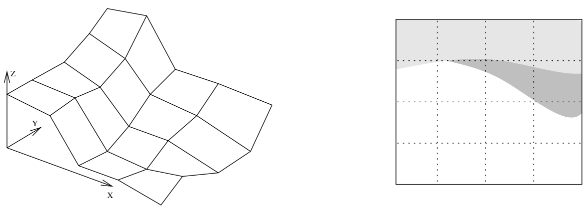


Figure 1: Digital Geology: The elevation model (DEM, on the left), and lithology or vegetation maps (right)

Digital Elevation Models (DEM) (see figure 1) give the altitude of the terrains on a 2D grid. Usually, the resolution is low in comparison with the simulations that will take place: a step of a few tens of meters is common. To that altitude map, we super-impose a lithological (such as granite, swamp, etc) and a vegetation map.

The lithology is used to compute how the ground deforms during a collision. Figure 2 present a typical collision cycle of a rock against a typical elasto-plastic material “with strain”. A collision process takes three steps:

1. Elastic compression: when the rock starts to deform the ground, the ground response force is a linear function of the penetration.

2. Plastic deformation: at a given pressure level, the internal structure of the ground is modified. The response force suddenly changes of slope, since the energy is mainly spent in deformation.

3. Elastic decompression: the response force decreases linearly *from the maximal penetration point*, and with *a higher slope* because of ground stress.

The dashed integral in the figure represents the energy loss during the collision. The difference between the cycle start and the cycle end gives the footprint that the rock will leave onto the ground. Geophysicists usually take parameters of stress cycles from existing tables, or measure them on the field.

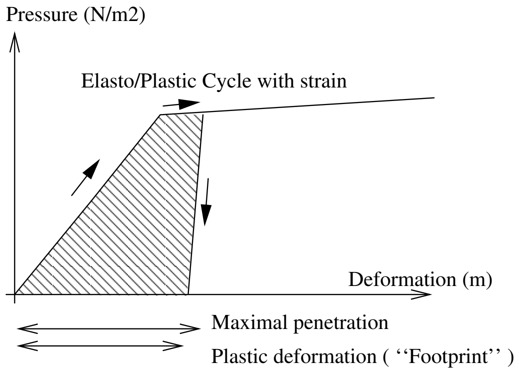


Figure 2: A typical stress cycle. The dashed area represents the energy loss during a collision.

Usually, natural terrains are not bare. Their vegetal cover plays an important part in the loss of energy during a collision. We take this into account by defining an “energy deperdition” coefficient weighing a friction force which is computed as a quadratic function of the rock’s speed at impact.

2.2 Ground model

In order to have a continuous description of the terrain, we must interpolate the DEM data. We use a bilinear interpolation (see figure 3a) of the various data known at each node of the DEM grid (altitude, elasto/plastic coefficients, vegetation friction, etc.) to sample the ground at a smaller scale. The refined ground model can be seen as a bidimensional array of “pistons” modeling the deformations of cylindrical samples of the soil in the normal direction to the local ground surface (see figure 3b).

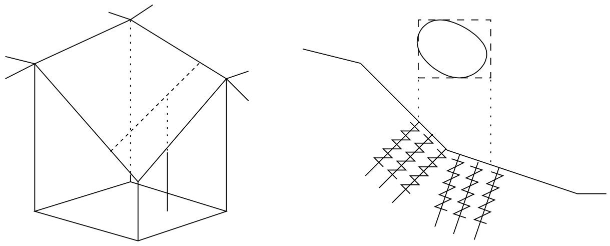


Figure 3: Digital Elevation Model (DEM): (a) bilinear interpolation of elevations; (b) and the model for pistons carpet below the rock.

During computations, each piston will store the local stress history needed to compute plastic deformations and response according to the given stress cycle. Geophysicists use to neglect transversal transport of matter, which is difficult to quantify. Moreover, we are only interested here into modeling the compression of the soil under the falling rock, which will affect the dynamics of the rock after the impact: modeling local bulges around compressed areas as was done in previous Computer Graphics works on footprints [3, 18] is not needed for our application. In consequence, each piston deforms and responds to collisions independently from its neighbors. It applies to colliding objects axial response forces, normal to the deformed ground surface.

Storing and simulating at each time step the whole ground at the smallest scale would consume too much memory and computational time. We rather perform a local adaptation of the ground's *level of detail* during computations: a fine grid is computed by caching the ground data only over a small "region of interest", defined as the vertical projection of the rock's axis-parallel bounding box. We call this region the "piston's carpet", as pistons are really computed only in this area. When the rock moves, rows and columns of pistons are removed or added to the adequate side of the carpet, thus taking benefits of temporal coherence. By this means, we focus computations only where needed.

As a result, footprints left by the rocks onto the ground, which are computed during impacts since they are essential for finding the ground's response, are not updated anymore when the rock moves down-slope to another region of interest. This greatly accelerates computations, since the number of pistons to process is proportional to the size of the falling rock, rather than to the size of the ground data-base.

3 Trajectory simulation for rocky blocks

3.1 Model for the rocks

In our simulations, rocks are rigid solids defined by an implicit surface [2] whose sample points are stored in a local frame. Each rock is provided with a mass and an inertia tensor, and is animated with usual dynamic laws of motion.

In some cases, modeling large rocky blocks fracturing into smaller ones may be useful. Our approach is to model these large blocks as a set of rigid parts linked together by geometric constraints [10]. When internal constraint forces exceed a given threshold, the constraint is suppressed, so the block breaks into pieces (see Figure 4).

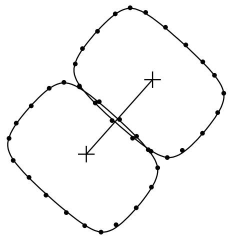


Figure 4: Rock fracture can be modeled using pre-fractured solids linked by breakable constraints.

3.2 Air friction during rock-fall

In addition to gravity, we model air-friction during rock flight phases. The equation we use is:

$$\overrightarrow{airFriction} = \frac{1}{2}\rho C_x \int ||V||\overrightarrow{V} \cdot \overrightarrow{n}\, dS$$

where $\frac{\overrightarrow{V}}{||V||} \cdot \overrightarrow{n}\, dS$ is a surface element of the rock, seen from the air flow, ρ the air density, and $0 < C_x < 1$ the drag coefficient (equals to 0.1 for a heavy sphere).

3.3 Collisions with the ground

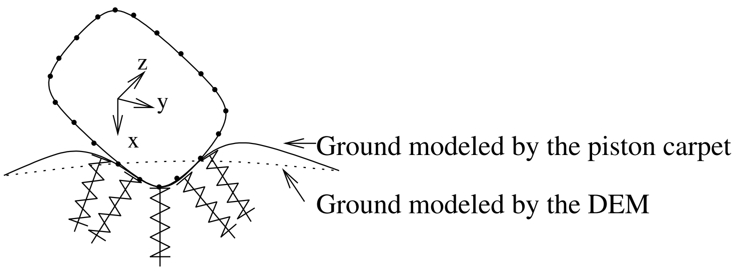


Figure 5: A rock interacting with pistons of the refined ground model

We compute collision detection and response within three steps, as in [11]:

1. We take benefits of the implicit formulation for the rock to optimize collision detection by testing each piston's surface point against the rock implicit function.

2. When a collision is detected, we model the rock's footprint on the ground by compressing along its axis each piston which penetrates the rock. Using the implicit formulation of the rock, the exact new locations of these pistons are computed through binary search.

3. Each piston computes a response force in the direction of the normal to the deformed ground according to its new length and to its current state (elastic

compression, plastic deformation, or decompression states). It also stores the new state to be used at the next time step.

4. Normal response forces and viscous friction forces weighted by the vegetation coefficient are added to the set of external actions to be applied to the rock up at the next time step.

Fast Bounce mode

Although the simulation process is quite fast (almost real-time on a low end workstation), it is sometime useful to simulate a hundred of trajectories in a few seconds, just to have an idea of a parameter change, for instance. Hence, a fast simulation mode was added to the simulator. The basic idea is to use a rough collision model, based on impulse and a single-point collision detection. The collision processing algorithm then becomes:

1. Test if the center point of the bottom face of the rock's bounding box is below the ground,

2. Move the rock up so that no more collision occurs, and generate an impulse at the collision point. The impulse is such that the speed after collision is reflected by the ground surface, and takes into account an arbitrary energy loss coefficient.

This energy loss coefficient could be chosen by geophysicists in relation with simpler simulation models, or tuned to match simulation results from the full rock-ground interaction model. In the first case, they could check if the new detailed simulation accurately match old well known but limited models. In the later case, they could fine tune the accelerated mode on a simple and well known configuration, then use it as a coarse model for complex configurations.

3.4 Stochastic Simulation and Results

As said before, the detailed geometry of the ground is not perfectly known. A randomization of various simulation parameters enables to have different trajectories for a block, and to assert hazard zones by stochastic simulations. But since the occurring contacts are not punctual (contact forces are integrated over a whole contact area), randomization of the terrain should have some spatial coherence (see figure 6). If they don't, summing over the rock's surface will average out the randomness added to the surface normals.

We use a Perlin[17] noise model, with a frequency of the order of the rock's size.

$$\begin{aligned}
A &= Tab[Hash(E(\lambda x), E(\lambda y))] \\
B &= Tab[Hash(E(\lambda x) + 1, E(\lambda y))] \\
C &= Tab[Hash(E(\lambda x), E(\lambda y) + 1)] \\
D &= Tab[Hash(E(\lambda x) + 1, E(\lambda y) + 1)] \\
Perlin_0(x, y) &= BilinearInterpol(A, B, C, D, frac(\lambda x), frac(\lambda y)) \\
Perlin(x, y) &= Perlin_0(x, y) + \frac{1}{2} Perlin_0(2 * x, 2 * y) \\
&+ \frac{1}{4} Perlin_0(4 * x, 4 * y) + \cdots
\end{aligned}$$

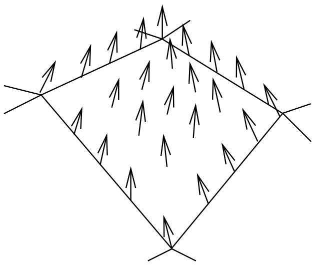


Figure 6: Because the force is integrated over a large contact area, the geometric randomness added for the stochastic simulation should have spatial coherence.

Where $E(x)$ is the integral part, and $frac(x) = x - E(X)$ the fractional part of a number x.

The basic idea is to initialize once for each simulation a random array $Tab[]$, and to have an hash function that maps the noise parameters (here the x and y space coordinates) to an index in that array. Four values are taken from that table, and bilinearly interpolated to have the base $Perlin_0$ noise. The base frequency is given by the parameter λ, and higher frequency noise are usually added by summing scaled $Perlin_0$ noise functions.

Gathering simulation data

To gather simulation data, a "*statistical map*" can be attached to any DEM, representing various parameters: blocks velocity, altitude, height over the ground, energy loss by friction, energy loss by ground deformation, etc. These data are saved for later analysis, or used to display statistical information on the ground, using arbitrary color scales on a vertically projected texture map.

4 Simulating mud-flows

4.1 Modeling mud with smoothed particles

The second typical phenomena in landslides is mud flowing down slope. Viscous fluids such as mud are difficult to simulate since conventional simulation methods as finite elements hardly cope with large deformations and changes of topology. Eulerian approaches, that consist in discretizing space into voxels and then computing what flows in and out each voxel [9] would not be convenient here, since interactions with obstacles which are not necessarily aligned with voxels need to be computed. Physically-based particle systems, such as those used in [15, 19, 20, 14, 7] seem a better approach for our case.

Our approach relies on the "smoothed particles" model introduced in [4]. The main feature of smoothed particles is to model materials governed by a macroscopic *state equation*. It defines the global behavior of the material independently of the sampling rate. Since real mud-flows may consist into large set of heterogeneous materials (including small rocks, trees, etc), no precise mechanical model could model such a

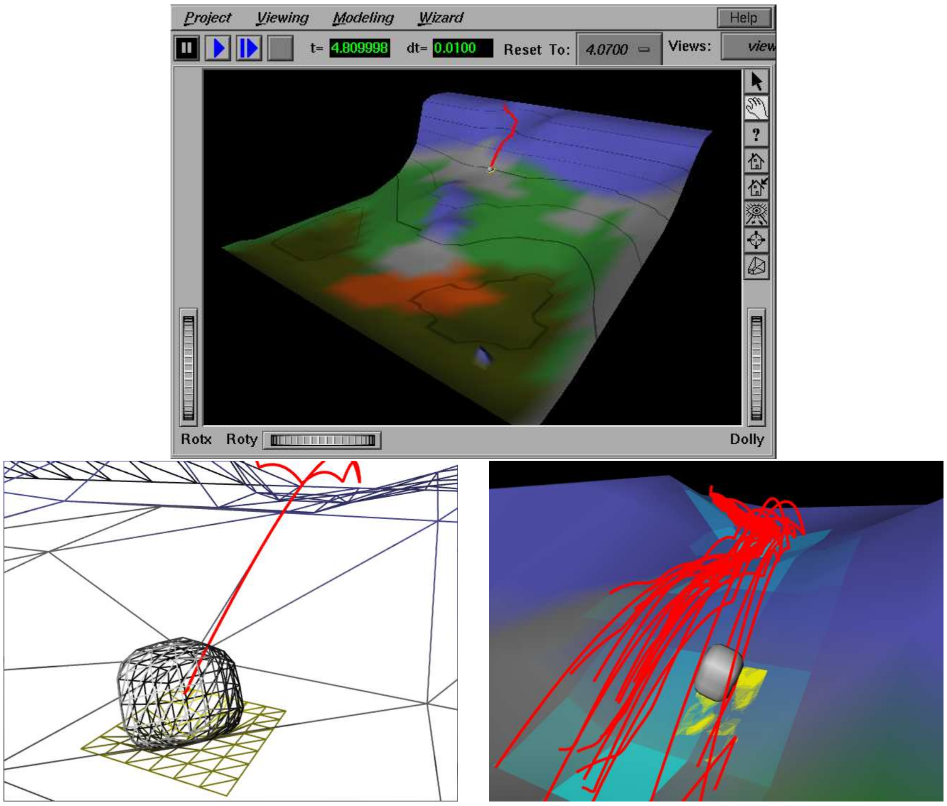


Figure 7: Top: Snapshot of the simulator. Left: a closup to the trajectory of a bloc (in red), with the *piston's carpet* mesh in yellow. Note the plastic deformation of the ground. Right: a set of trajectories. The ground is colored in cyan by the energy loss due to ground collision.

behavior accurately. Defining a macroscopic behavior, designed or fitted on data, is then convenient. The state equation governs the evolution of a *pressure* field P inside the substance. Conservative internal forces, called "pressure forces", are proportional to the gradient of pressure [1]. These forces are combined with dissipative forces that model internal friction inside the deformable body.

Our macroscopic model for mud flows is a viscous substance that comes back to a constant volume when no external force is applied. To simulate this behavior, we set the state equation to [4]:

$$P \;=\; k(\rho - \rho_0) \tag{1}$$

This generates pressure forces that tend to restore a rest density ρ_0 when the current density ρ has become too high or too low. The parameter k in equation (1) controls the strength of density recovering, and is thus analogous to a stiffness parameter.

This equation is a simplified version of the state equation for fluids flows in isothermic media:

$$\rho = \rho_0 e^{-c(P_0 - P)}$$

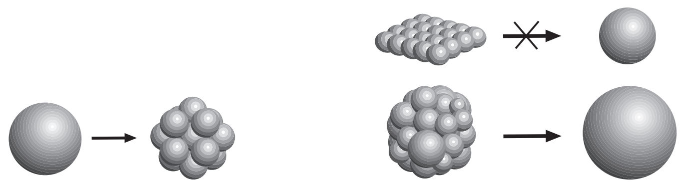
where variations of the compressibility $c = \frac{1}{\rho}\frac{\partial \rho}{\partial P}$ depending of pressure have been neglected (i.e. $c = c_0$, the compressibility at reference pressure P_0), and $c_0|P_0 - P|$

supposed sufficiently small.

During a simulation, the matter is sampled by particles, or sample points, representing a small mass distribution around them. Attraction/repulsion forces between particles are derived from the state equation as being proportional to the gradient of pressure, thus yielding different approximations of the same behavior whatever the sampling resolution. Then, motion is obtained through integration of forces over time.

Modeling mud-flows with changing mass as in [12] is easy with our model: we just have to position some extra material on the terrain. During the simulation, the flow may entrain some of this ground material while some of the flow particles may be stopped by friction.

4.2 Adaptive simulation



(a) (b)

Figure 8: Adaptive animation scheme: (a) Particles are divided into smaller ones where high deformations are taking place. (b) Particles merge in stable areas.

To increase efficiency while maintaining a given accuracy, the number and size of particles sampling the mud-flow are adaptively modified during computations. Local fission and fusions of particles are automatically generated in order to optimize computation while keeping a desired accuracy. As a result, fewer and larger particles are used in stable areas while refinements occur where the substance undergoes deformation.

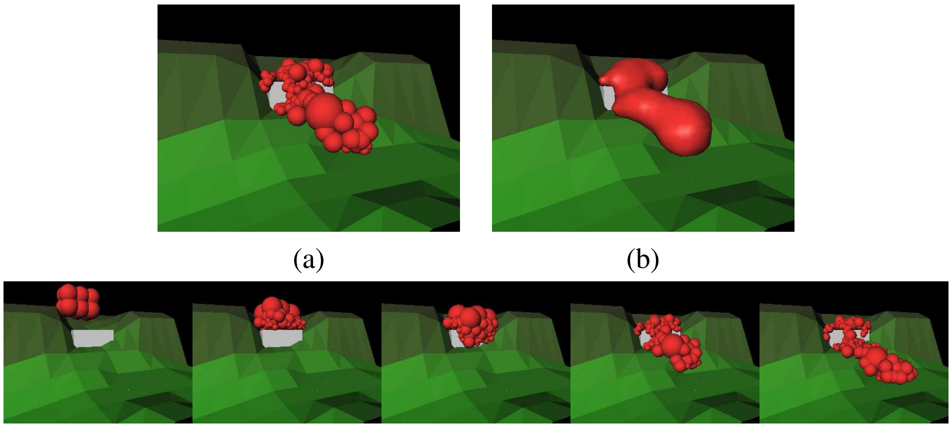
More precisely, during an animation:

- A particle subdivides if the local difference of pressure with its neighbors compared to the particle size exceeds a given threshold.
- A group of particles merge into one particle when the volume they sample is almost spherical and their difference of pressure is under another threshold (see Figure 8) .

As our deformation model relies on a pressure proportional to the local mass density variation, these criteria ensure a better sampling where motion is about to occur, while it cuts down computation where it is stable. Particles are also simulated with individual time steps that are computed from their size and from stability criteria such as Courant condition, further improving efficiency while ensuring stability [6].

4.3 Results

A frame from an adaptive simulation of a mud-like substance is shown in Figure 9 (a). The adaptive simulation is about 4 times faster than the non-adaptive one computed with the finest particle's scale. Displaying particles as spheres gives a good idea of the subdivision/fusion process. In practice, we often coat the adaptive particle system with an *active implicit surface* [5], to get a smooth rendering. This active surface filters the changes of granularity of the internal particle system as it tracks a mass density isovalue while simulating a surface tension. It also yields an efficient polygonization, since it is computed as an iso-surface of a discrete field stored in a grid.



(a) (b)

(c)

Figure 9: Mud-like substance flowing over a dam: (a) Adaptive particles displayed as spheres. (b) Particles coated with an active implicit surface. (c) The whole animation sequence.

5 Conclusion

The presented rock-fall simulator is developed in C++ (≈10.000 lines), and interfaced with the *Tcl* scripting language. It has already been used for an expertise. The *Motif* interface is being adapted to be usable by geophysicists, and we are currently calibrating our simulations using video-recorded rock falls in a stone quarry. The mud-flow simulator is still in progress: the base simulation model is fully functional, and integration into the geophysic framework of the first simulator is currently on the way.

Future works includes the modeling of snow and lava, which requires to add temperature in our state equation.

Acknowledgments: The authors would like to thank Frederic Pontarollo and Jocelyne Leroy for their contributions to this project.

References

[1] G.K Batchelor. *An Introduction to Fluid Dynamics*. Cambridge University Press, 1973.

[2] Jules Bloomenthal, editor. *Introduction to Implicit Surfaces*. Morgan Kaufmann, July 1997.

[3] B. Chanclou, A. Luciani, and A. Habibi. Physical models of loose soils dynamically marked by a moving object. In *Computer Animation Conference 1996*, pages –, June 1996.

[4] Mathieu Desbrun and Marie-Paule Cani-Gascuel. Smoothed particles: A new approach for animating highly deformable bodies. In Springer Computer Science, editor, *7th Eurographics Workshop on Animation and Simulation*, pages 61–76, Poitiers, France, September 1996.

[5] Mathieu Desbrun and Marie-Paule Cani-Gascuel. Active implicit surface for computer animation. In *Graphics Interface (GI'98) Proceedings*, Vancouver, Canada, June 1998.

[6] Mathieu Desbrun and Marie-Paule Cani-Gascuel. Space-time adaptive simulation of highly deformable substances. *Submitted for publication*, pages –, 1998.

[7] Mathieu Desbrun and Marie-Paule Gascuel. Animating soft substances with implicit surfaces. In *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 287–290. ACM SIGGRAPH, Addison Wesley, August 1995. Los Angeles, CA.

[8] S. G. Evans and O. Hungr. The assessment of rockfall hazard at the base of talus slopes. *Can. Geotech. J.*, 30:620–636, 1993.

[9] Nick Foster and Dimitri Metaxas. Realistic animation of liquids. *Graphical Models and Image Processing*, 58(5):471–483, 1996.

[10] Jean-Dominique Gascuel and Marie-Paule Gascuel. Displacement constraints for interactive modeling and animation of articulated structures. *The Visual Computer*, 10(4):191–204, March 1994. An early version of this paper appeared in the *Third Eurographics Workshop on Animation and Simulation*, Cambridge, UK, Sept 92.

[11] Marie-Paule Gascuel. An implicit formulation for precise contact modeling between flexible solids. *Computer Graphics*, 27:313–320, August 1993. Proceedings of SIGGRAPH'93 (Anaheim, CA).

[12] O. Hungr and S. G. Evans. A dynamic model for landslides with changing mass. In Marinos, Koukis, Tsiambaos, and Stoumgras, editors, *Engineering Geology and the Environment*, 1997.

[13] E. Leroi, F. Pontarollo, J-D. Gascuel, M-P Gascuel, and M. Bour. Development of a 3d model for rock-fall trajectories, based on synthetic imagery and stress-deformation laws. In *7th International Symposium on Landslides*, Trondheim, Norway, June 1996.

[14] Annie Luciani, Stéphane Jimenez, Olivier Raoult, Claude Cadoz, and Jean-Loup Florens. A unified view of multitude behaviour, flexibility, plasticity, and fractures: balls, bubbles and agglomerates. In *IFIP WG 5.10 Working Conference*, Tokyo, Japan, April 1991.

[15] Gavin Miller and Andrew Pearce. Globular dynamics: A connected particle system for animating viscous fluids. *Computers and Graphics*, 13(3):305–309, 89. This paper also appeared in SIGGRAPH'89 Course notes number 30.

[16] Hungr O. A model for runout analysis of rapid flow slides, debris flows and avalanches. *Can. Geotech. J.*, 32:610–623, 1995.

[17] Ken Perlin. An image synthesizer. In B. A. Barsky, editor, *Computer Graphics (SIGGRAPH '85 Proceedings)*, volume 19, pages 287–296, July 1985.

[18] Robert Sumner, James O'Brien, and Jessica Hodgins. Animating sand, mud, and snow. In *Graphics Interface*, pages –, June 1998.

[19] Demetri Terzopoulos, John Platt, and Kurt Fleisher. Heating and melting deformable models (from goop to glop). In *Graphics Interface'89*, pages 219–226, London, Ontario, June 1989.

[20] David Tonnesen. Modeling liquids and solids using thermal particles. In *Graphics Interface'91*, pages 255–262, Calgary, AL, June 1991.