Animating Lava Flows

Dan Stora Pierre-Olivier Agliati Marie-Paule Cani Fabrice Neyret Jean-Dominique Gascuel

iMAGIS†-GRAVIR / IMAG
BP 53, F-38041 Grenoble cedex 09, France
Marie-Paule.Cani@imag.fr

Abstract

Animating lava flowing down the slopes of a volcano brings several challenges: modeling the mechanical features of lava and how they evolve over time depending on temperature; computing, in a reasonable time, the interactions inside the flow, and between the flow and a complex terrain data-base; and lastly, rendering the visual aspect of the flow. The methods described rely on smoothed particles governed by a state equation for animating the flow. We adapt this model to the animation of lava by linking viscosity to a temperature field and by simulating heat transfers. We propose particular data-structures that lead to linear computational time with respect to the number of particles. Lastly, we study a model based on a color-and-displacement procedural texture controlled by the flow for the realistic rendering of lava.

Key words: Natural phenomena, animation of fluids, particle systems, procedural textures, implicit surfaces.

1 Introduction

Modeling the richness and complexity of nature is a constant challenge for Computer Graphics. In addition to realistic shapes and rendering, Computer Animation techniques try to generate consistent motion and deformations. A wide range of natural phenomena have been animated over the past few years. They include gaseous phenomena such as fire or smoke [13, 18, 8], liquids – from waves to droplets [9, 12, 16, 7, 10], and the simulation of soft terrains, from footprints to landslides [4, 19, 11].

This paper addresses another complex problem: the animation of lava flows. Our aim is to provide visual realism in both motion and rendering. In order to generate complex trajectories due to the interactions between lava-flows and a complex terrain database, we need to model the main mechanical features of lava and the way they evolve over time. Another difficulty is to compute the simulation at acceptable rates. Lastly, we would like to provide a convincing visualization of lava "skin" appearance, even though we are not attempting to use a simulation approach for this part of the phenomenon.

Recently, several movies featuring volcanos have been launched, showing the potential impact of lava animation in the field of special effects. However, despite the advertising that announced the use of Computer Graphics techniques, 'Dantes Peak' and 'Volcano' mostly use regular special effects, including scale models for lava. As far as we know, Computer Graphics was used for the 2D compositing of these models with fire images, rather than for computing a graphical simulation of the flows.

Geologists have been interested for years in the simulation of volcanic explosions and lava flows. However their concern is not producing realistic images, but rather computing a realistic account of flow and ash coats (e.g. in a vertical section), and measuring the way some global physical values vary over time. Thus, their models (see `http://www-geo.lanl.gov/wohletz/Erupt.html`) do not seem directly usable for a Computer Graphics simulation.

In terms of physically-based simulation, lava can be seen as a liquid whose viscosity increases exponentially when the material cools down. Previous solutions to the animation of liquids range into two main categories: Eulerian approaches, that consist of discretizing space into a fixed 3D mesh and studying how physical fields evolve at the mesh nodes [12, 7]; and Lagrangian approaches that consist of following the motion of punctual mass elements, called particles, that sample the liquid. The second approach seems more convenient in the case of lava-flows since the use of particles enables us to attach a deformable lava skin to the flow without introducing extra marker elements. In addition, particle systems have successfully been used for modeling a wide range of behaviors that includes very viscous substances, and for animating transitions between solid and liquid states, using a temperature parameter [20, 21]. Among particle systems, the smoothed particle model introduced in [15, 6] plays an original part, since inter-particle forces are not tuned by hand, but are rather derived from a state equation that defines the macroscopic behavior of the material. This model has been applied to the simulation of mud

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

slides [11]. There has been no attempt yet, to the authors knowledge, to modify its mechanical features over time.

The approach described in this paper relies on an extension of the smoothed particle model. We control the evolution of the lava dynamics over time by providing each particle with a viscosity parameter, which depends on temperature. During a simulation, heat transfers at the surface of the lava and inside the flow causes the temperature to decrease, enabling the transition from a low-viscosity behavior to a highly viscous one. The visual aspect of the lava changes accordingly, thanks to a coating of the particles that controls local color and granularity of the lava skin as functions of temperature.

Section 2 details the model we have designed for the physical features of lava. Implementation issues for the efficient computation of a lava-flow, including the design of convenient data structures, are discussed in Section 3. Section 4 describes our method for generating a visually realistic rendering of the animation.

2 Physically-based Modeling of Lava

2.1 Modeling Lava

Natural lava-flows present a wide diversity of morphologies and structures. Among other factors, the topography of terrain, and the flow rate out of the crater affect the lava spread. These factors will be considered in Section 3 for the practical computation of lava-flows. Studies of the main mechanical features of lava [3] establish that:

- Viscosity depends on the chemical composition of the lava;
- For a given chemical composition, viscosity increases exponentially when the temperature decreases (see Figure 1, from [3]).
- The mass-density of lava remains quasi-constant until it commutes to the solid state. The value of mass-density for basaltic lavas is around 2500 $kg.m^{-3}$.

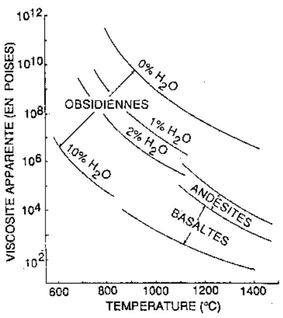

Figure 1: Viscosity as a function of temperature, for lavas of different chemical composition.

Our aim is to simulate the lava flowing process. We are looking for a general model of a viscous fluid, robust enough for enabling high viscosity changes inside the material. These changes will take place in space – as viscosity is not constant along a lava flow at a given time – and in time, since lava is cooling down during the simulation. During the simulation, the flow's mass density should be maintained around 2500 $kg.m^{-3}$.

The next section presents the deformable model that we have used as the background for our simulations. This model uses a particle system for modeling the flow, which is also convenient for rendering lava skin as will be shown in Section 4. Its main feature is that, contrary to standard particle systems, it sets the material to a specified mass-density at rest, thanks to a state-equation that describes the desired macroscopic behavior.

2.2 Smoothed particles maintaining a constant rest-density

Our model is an extension of the smoothed-particles simulation method developed by Mathieu Desbrun[1] [6]. This section reviews this previous work.

Smoothed particles simulate a substance whose physical characteristics are governed by a *state equation*. The following equation states that the pressure[2] field P inside the material is proportional to the difference between the current mass-density ρ and a given rest value ρ_0:

$$P = k(\rho - \rho_0)$$

where the parameter k represents a stiffness coefficient.

During a simulation, the material is sampled by "smoothed particles". Each of them represents a sample of mater of fixed mass m_i which is distributed in space around its current location x_i according to a smoothing kernel W_h (parameter h controls the extend of the kernel's support). The particles sample the values of continuous physical fields such as pressure P and mass-density ρ inside the material (so P_i and ρ_i respectively denote pressure and density at particle i). Interpolation yield the value and gradient of these physical fields anywhere in space:

$$\begin{aligned} f(x) &= \sum_j f_j \frac{m_j}{\rho_j} W_h(x - x_j) \\ \vec{\nabla} f(x) &= \sum_j f_j \frac{m_j}{\rho_j} \vec{\nabla} W_h(x - x_j) \end{aligned}$$

where f_i is the value of field f at particle i.

This can be used for deriving the expression of internal pressure forces exerted on particle i from the expression

[1]The formalism was originally inspired from works in astrophysics [15] on "Smoothed Particles Hydrodynamics" (SPH).

[2]As in [6], the term "pressure" we use is not totally adequate: P is rather a *difference of pressure* with a given reference value, thus it can take either positive or negative values.

of pressure given by the state equation [6]:

$$\vec{F}_{P_i} = \sum_{j \neq i} -m_i m_j \left(\frac{P_i}{\rho_i^2} + \frac{P_j}{\rho_j^2} \right) \vec{\nabla}_i W_h^{ij}$$

where $\vec{\nabla}_i W_h^{ij}$ is the gradient of $W_h(x_i - x_j)$. The density field is computed by integrating pressure changes over time from its initial value ρ_0 (corresponding to the initial particle distribution) using the continuity equation:

$$\rho \; div(V) + \frac{d\rho}{dt} = 0$$

The component of the interaction force $\vec{F}_{P_i}$ which is exerted between particle i and particle j can be either positive or negative according to the sign of the over-pressures P_i and P_j. It has been shown in [6] that these forces are similar to Lennard-Jones attraction-repulsion forces (ie. long-range attraction and short-range repulsion) that are commonly used in physically-based particle systems [14]. One of the advantages of SPH is that no local user-specified parameter is needed, since they are directly derived from the global state equation.

As with any particle system, the animation is computed by integrating, for each particle, applied forces over time. Non-conservative forces that model viscosity inside the material, as well as external forces such as gravity and collision forces with obstacles, are added to pressure forces. The smoothed particles representation provides an expression for the maximal time step at which each particle should be simulated, computed from Courant's condition [6]. The use of an adaptive time-step based on this minimal rate ensures stability of computations.

The main benefits of this model is that the user defines the macroscopic behavior of the material through the state equation. As a consequence, the same substance may be simulated with variable accuracy, by simply tuning the number of particles that sample it (this is not the case for particle systems based on Lennard-Jones interaction forces, such as those used in [20, 21], since the material behavior then depends on both forces parameters and the scale at which these specific forces are used). Another advantage of this model is that we are able to control the rest mass-density, whose value is well known for lava. In practice, we model lava by taking the same smoothing kernel that in [6] and by setting the parameters of the state equation to[3]:

$$\rho_0 = 2500 \; kg.m^{-3} \qquad k = 90000$$

The above model is not sufficient for modeling all the physical features of lava listed in Section 2.1. In the next two sections we extend the model to include variable viscosity and to simulate heat transfers over time.

[3] the higher k is, the more incompressible the material will be, but the required time step is smaller.

2.3 Modeling viscosity inside a lava-flow

Viscosity plays a very important part in the dynamics of lava flows. The model for a viscosity force must be chosen with care. The expression of viscosity force used in [15, 6], originally designed for cosmologic fluids, was:

$$\vec{F}_{v_i} \;=\; -m_i \sum_{j \neq i} m_j \Pi_{ij} \vec{\nabla} W_h^{ij}$$

where Π_{ij} is a function of the relative speeds of particles i and j, and of the distance between them:

$$\Pi_{ij} = \begin{cases} \frac{hc(v_{ij} \cdot x_{ij})}{\rho_{ij}(x_{ij}^2 + h^2/100)} & \text{if } v_{ij} \cdot x_{ij} < 0 \\ 0 & \text{if } v_{ij} \cdot x_{ij} \geq 0 \end{cases}$$

where c is the speed of sound[4] inside the material and $v_{ij} = v_j - v_i$, $x_{ij} = x_j - x_i$, $\rho_{ij} = (\rho_i + \rho_j)/2$.

This expression is not convenient for us, even for modeling viscosity at a given temperature. Firstly, the force contribution from a given neighbor particle j becomes zero when the distance between the two particles increases. Secondly, even when the speed vectors are not oriented along the segment between the particles, this force contribution is always applied along this segment. Lastly, the parameters are not easy to calibrate due to the complex formulation of Π_{ij}.

We rather use a more intuitive and simpler expression for viscosity forces, aimed at modeling the variations of the velocity field at the neighborhood of a particle. The value of the force is proportional to the difference between the particle's speed and the mean speed around it:

$$\vec{F}_{v_i} \;=\; \alpha_v \frac{m_i}{\rho_i} \sum_{j \neq i} \frac{m_j}{\rho_j} v_{ij} W_h^{ij}$$

Since the viscosity of lava increases exponentially when temperature decreases, we set the parameter α_v to:

$$\alpha_v \;=\; b \exp(-aT_i), \; a > 0, \; b > 0$$

where T_i is a new parameter representing temperature, which is associated with each particle (we use $a = 220$ and $b = 0.5$ in our simulations). Next we have to design a simulation method for computing the local variations of the temperature field during a simulation.

2.4 Simulation of heat transfers

Two kinds of heat transfers should be taken into account during a simulation: those that are internal to the lava flow, and those that are external (i.e. transfers with the air and with the ground).

[4] i.e. the speed of the fastest propagation wave.

As in [20, 21], we model heat transfers inside the material by integrating the general heat equation:

$$\frac{dT}{dt} = k\triangle T$$

However, the way we do this is different from previous works, since we have to express it is the smoothed particles formalism. Since SPH cannot be used for computing second derivatives due to the smoothing effect of the kernel W_h, directly computing $\triangle T$ (the Laplacian of temperature) is difficult. In practice, we compute $\triangle T$ for each particle using:

$$\vec{\nabla} T_i = \sum_{j \neq i} m_j \frac{T_j - T_i}{\rho_j} \vec{\nabla} W_h^{ij}$$
$$\triangle T_i = \sum_{j \neq i} m_j \frac{\vec{\nabla} T_j}{\rho_j} \vec{\nabla} W_h^{ij}$$

Thanks to temporal coherence and very slow speed of the heat transfers inside the material, we can use the value of $\vec{\nabla} T_i$ stored at a given time step for computing $\triangle T_i$ at the subsequent time step, in a single list traversal.

Now we need to model heat transfers that take place at the surface of the lava flow, including both lava-air and lava-ground interfaces. The term representing the time variation of temperature for a particle located at the surface of the flow should[5] be proportional to the difference between its temperature and the temperature of exterior medium, proportional to the size of the contact surface, and inversely proportional to the local mass density:

$$(\frac{dT}{dt})_{ext_i} = k_{ext}(T_i - T_{ext})\frac{r_i^2}{\rho_i}$$

where r_i, the approximate radius of the small sample of lava the particle models, is computed from:

$$\frac{4}{3}\pi r_i^3 = \frac{m_i}{\rho_0}$$

In our simulations, we set the temperature of basaltic lava to 1200 $^\circ C$ when it spreads out of the crater. It goes down to about 900 $^\circ C$ during the simulation.

3 Computing animations of lava flows

3.1 Data structures

The bottleneck of any simulation of a particle system is the computation, at each time step, of the interactions between neighboring particles. In the smoothed particles formalism, interactions (through pressure and viscosity forces) take place as soon as the distance between two particles is smaller than $2h$, where h is the radius of their smoothing kernel. Since the model maintains the material around a given rest density, the number of neighbors always stays around a given small value (between 20 and 30 in our simulations). However, in a brute force implementation, the neighbor search would be quadratic in the number of particles, which would lead to unacceptably low performances. A well-known solution consists of using a 3D grid that stores the connected list of the particles that lie inside each voxel. In the specific case of lava spreading out of a volcano, this solution cannot be used directly, since the 3D grid would be too large.

Our solution exploits the fact that while lava spreads far away from the crater, the local vertical range of the flow always remains small compared to the size of the terrain (see Figure 2 (a)). Thus, we use a predefined grid of a large size in the horizontal plane, but with only a few elements per vertical column. The vertical scope of each column is adapted in order to cover the local vertical range of the flow over this precise point of the horizontal plane (see figure 2 (b)). During neighbor search, only particles lying in a grid voxel that is closer than $2h$ from the current particle are considered. Since lava approximately maintains its rest density everywhere, calibrating the grid in order to have an almost constant number of particles per occupated voxel is easy. This strategy provides a simulation time which is close to linear with respect to the number of particles.

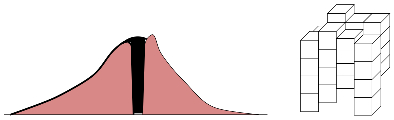

Figure 2: (a) Lava spreading out of a volcano (cross section). (b) Data structure for neighbors search.

The terrain is defined by a digital elevation model (DEM), i.e. a 2D grid where altitudes are stored. The use of a DEM makes easier the detection of collision between a particle and the terrain, since it only requires the computation of the terrain's altitude (through bilinear interpolation) at the particle's projection onto the horizontal plane. In practice, collision detection is performed only for particles that are at the surface of the flow. The computation of this subset of particles is also necessary to apply the heat transfer equations. The next section proposes a criteria for performing this computation.

3.2 Characterization of particles at the flow surface

An original feature of highly deformable materials compared with solid deformable bodies, is that the set of small matter elements that lie near the surface may change over time. This is particularly obvious when a

[5]This is just a simple model convenient for a Computer Graphics applications. We do not pretend to simulate the real physics of lava cooling such as degazing and radiative transfers.

flow separates into several disconnected components, or when some components merge. In our simulations, the set of particles lying near the surface has to be recomputed at each time step. We need a robust criteria for characterizing these particles.

The gradient of mass-density at a particle location indicates the direction where the highest neighbor mass is found. So the border should be searched for in the opposite direction. Our criteria states that a particle :

is on the surface if the quotient between the mass of its neighbors located in the half space oriented by $-\vec{\nabla}\rho_i$ *and the total mass of all neighbors is under a threshold.*

Since this criteria will not work when the lava has completely spread out (the gradient of mass-density is in the particle's plane when all the particles are in contact with a flat part of the terrain) we add a second criteria:

a particle is at the border if that particle and all its neighbors are approximately located in the same plane.

3.3 Simulation process for lava flows

At each integration time step:

- Generate some more particles inside the crater, according to the desired flow rate.
- For each particle:
  1. Compute the list of neighbors.
  2. Use it to compute ρ_i, $\vec{\nabla}T_i$, and $\triangle T_i$.
  3. Compute pressure and viscosity forces.
  4. Integrate the equation of motion according to the set of external and internal forces.
  5. If the particle is on the surface of the flow:
     - test for collisions with the ground,
     - compute the heat transfer with the exterior.
  6. Compute the heat transfers with neighboring particles.
- If necessary, modify the value of the integrate step in order to maintain Courant's condition.

We have seen that in order to maintain lava in a quasi-incompressible state, our simulations are computed with a high value for k. Courant condition then leads to about 64 integration steps between two frames, using integral integration schemes for increasing stability. Even in this case, computational time is reasonable: On a SGI O^2 workstation, an image is generated every 20 seconds for a simulation of 1000 particles, and every two minutes for a simulation of 6000 particles. If we decrease k to a less realistic value such as $k = 100$, less integration steps are needed between two images, so we obtain one image per second for 1000 particles.

Figure 4 shows some steps of a lava flow for our standard high value for k. Particles are displayed as spheres of radius r_i. Their color varies with temperature. Displaying particles as spheres is adequate for carefully following each particle's motion, but does not, however, yield the visually realistic rendering we need.

4 Visually realistic rendering

4.1 Trying to obtain the aspect of basaltic lava

Figure 7 depicts a real basaltic lava flow of type "aa", whose particularity is the formation of clinkers during the slow transition between fluid and solid states. These clinkers are made of the lava foam that has cooled down quicker that the rest of the flow, and merged into regularly spaced clusters. In this paper, we are trying to render this specific kind of lava, whose roughness, size of clinkers, and color highly depend on temperature.

At first sight, obtaining a similar visual aspect for our synthetic lava flows looks like a challenge. The most current solution for defining a surface around a set of particles consists of computing an iso-surface of a sum of scalar fields generated by the particles [1, 20, 21, 5]. This solution easily deals with topology changes (separations and fusions) but is not convenient for our application since implicit surfaces generate very smooth shapes [2].

Motion and deformations of our lava flows are simulated at a relatively large scale, then it seems obvious that more geometric complexity should be generated. Performing a micro-simulation of the lava skin mechanics would be difficult, computationally intensive, and is not really useful since our only aim is visual realism. Thus we must deal with this smaller scale by providing a geometric dressing of the larger one. Perturbating the surface with a stochastic continuous noise, such as Perlin's noise [17], seems a good approach. A first idea would be to define a 3D noise depending on temperature in a local frame linked to each particle, and to use it to deform the implicit primitive defined by the particle. The surface of the flow would then be computed as an iso-surface for the sum of perturbated fields. However, this approach would be extremely time consuming due to the resolution needed for polygonizing the surface. Moreover, when two neighboring particles move at different speeds, the addition of noise contributions would probably produce aliasing artifacts (such as interferences).

Our approach consists in generating a procedural displacement map, together with a color texture, on the smooth implicit-surface generated by a standard isotropic fields around the particles. Both color and roughness of the texture are function of the local temperature of the flow. One of the challenges is to maintain temporal coherence when an animation sequence is rendered: each surface detail may continuously deform and change its color, but must consistently follow the flow.

4.2 A lava skin that follows the flow

Texture is going to be generated on triangles, so the first point is to tessellate the implicit surface. This cannot be done by using a standard surface tiling algorithm [2] at each time step since, because of our need of temporal coherence, each triangle should *deform and move with the underlying flow*.

Our solution is the following:

1. We associate a sample point on the implicit surface to each particle that verifies the flow surface criteria of Section 3.2. This is done by searching for the iso-surface in the direction of the field's gradient on the particle.
2. We compute the Voronoï diagram of these sample points: for each of these points, we sort sample points that correspond to neighboring particles in counterclockwise order with respect to the surface's normal, and select those that are contributing to the Voronoï region.
3. We tessellate each Voronoï region into triangles that turn around the sample point defining the region.

This results into triangles that both tessellate the flow surface, define the neighborhood of a particle on the surface, and consistently follow the flow (see Figure 3).

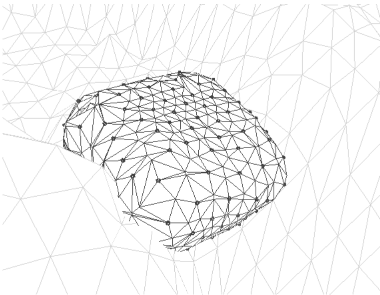

Figure 3: Tesselated Voronoï regions on a flow surface.

4.3 Generating a procedural texture

The idea is to generate a lava skin clinker into each Voronoï region. Since the sample point at the center of the region follows a given flow particle, the clinkers will be adequately carried by the flow.

Clinkers and lava skin between them are modeled by a texture that includes both local color and also a displacement map controlling surface altitude and roughness. We compute the displacement map as the sum of two terms (see Figure 6):

- a given 3D profile defining the shape of a clinker,
- a stochastic altitude perturbation modeling the local surface's roughness.

Continuity conditions at the border of triangles must be maintained carefully. This includes both color and displacement G^1 continuity (i.e. geometric continuity of derivatives).

Modeling specific profiles whose shape and altitude depend on temperature, and whose support is included into each Voronoï region is straightforward. Color consistency and continuity can easily be obtained by associating a given color to each altitude, with a color map depending on the local temperature of the flow. The next paragraph concentrates on the most difficult part of the process, ie, defining a G^1 stochastic noise which gives the desired visual effect.

4.4 Continuous Perlin's noise on a triangular mesh

Perlin's noise [17] is a stochastic auto-similar G^1 noise that has become a standard for generating objects that look like wood, marble, fog, or rock surface. One of its main features is to maintain continuity of both noise and noise derivatives. The idea is to use it for defining a 2D displacement texture that will serve as a perturbation of altitude over each lava skin triangle.

However, we need to generate the noise on a 2D triangular mesh, since it is easier to guarantee continuity through two adjacent texture patches if the noise control points fall exactly on their boundary. Perlin's standard model being defined on a quadrangular grid, we modify the algorithm in the following way:

1. we generate a pseudo-periodic noise function on a regular grid of equilateral triangles by:
   - randomly associating a plane to each grid node, defined by its elevation above the node and by its normal;
   - defining the noise at any point inside the mesh as the barycentric interpolation of the distances to the three planes associated to the vertices of the triangle where the point lies.
2. we define the final noise giving the altitude of a mesh point as the sum of instances of the pseudo-periodic noise defined above, applied at different scales thanks to recursive subdivisions of the triangular mesh. This gives a fractal aspect to the noise.

The color and displacement texture we obtain is depicted at the top right of Figure 6. Since the noise function has been generated on a regular grid made of equilateral triangles, applying it on the non-equilateral triangles resulting from the tessellation of Voronoï regions may locally alter the isotropy of the noise. In practice, we reduce this problem by suppressing the very thin triangles (by merging their edges) that may appear around a sample point.

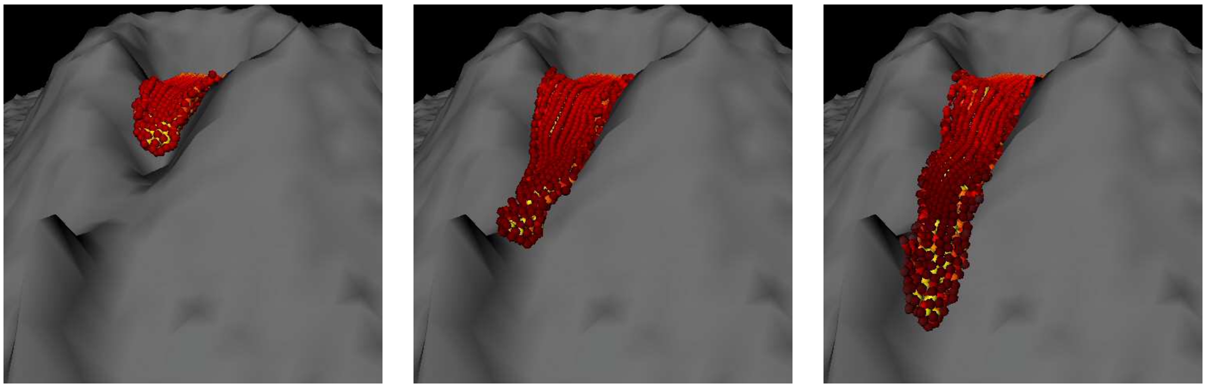

Figure 4: An example of lava-flow where particles are displayed as spheres. About 3000 particles have spread out of the volcano in the last image.

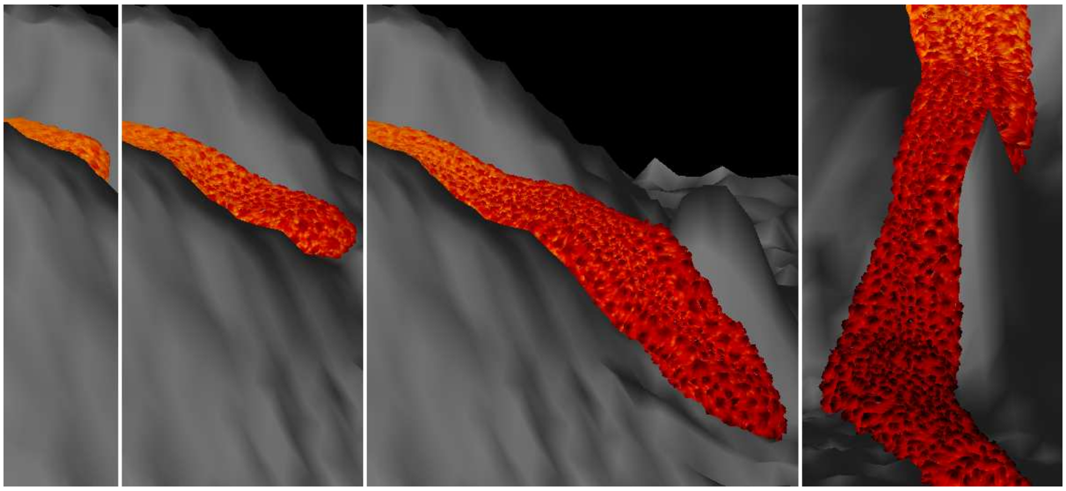

Figure 5: A textured lava flow.

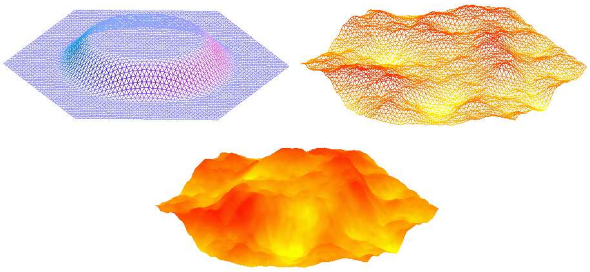

Figure 6: Top left: a clinker's profile; top right: stochastic perturbation modeling the surface's roughness; bottom: the resulting texture.

Figure 7: A real lava flow for basaltic lava of type "aa".

Computational time has been measured on a test scene containing one clinker made of 6 triangles (see figure 6, bottom) covering a video-sized screen and subdivided 32 times (so that 6144 very small triangles are drawn). Computing the texture takes about 0.15 seconds per frame on a SGI O^2 workstation. As small triangles that sample the texture should always be of the same size (about a few pixels), the subdivision depth decreases when the number of polygons increases, so the rendering time remains of the same order of magnitude for any scene (less than a second per frame).

5 Conclusion

This paper has focussed on the animation and rendering of lava flows. Figure 5 depicts some steps of an animation where lava flows down the slopes of a volcano. Animation sequences are available from `http//www-imagis.imag.fr/LAVA/`.

Lava flows represent a challenge for physically-based animation, since the mechanical features of lava change over time. This change is governed by a temperature field, that needs to be animated. The first contribution of this paper is to show that the smoothed particles formalism is convenient for performing this kind of simulation. In order to model lava, we extend the equations, and we propose an efficient way to simulate them. Internal forces inside the flow, and interactions between the flow and a complex terrain data-base are computed in a reasonable time, by the implementation of adequate data-structures. Performing an animation is made easier by the use of a macroscopic model for the lava flow, which only includes a few intuitive parameters (mass, density, stiffness, initial viscosity and temperature of lava, air and ground temperatures).

In the context of searching for visual realism, we also addressed the problem of rendering lava. Our solution, designed for a specific type of basaltic lava flow, consists of rendering a moving texture controlled by the flow and generated on the fly on an implicit surface that surrounds the particles. The texture combines color and displacement information. The displacement map is procedurally computed as the sum of a large scale shape and of a Perlin's noise component. Texture moves and deforms with the underlying flow according to the local variations of the flow speed and temperature. In particular, roughness of the texture increases when the lava cools down, while its color changes. Our methods for defining the displacement map allows us to control the visual aspect of the lava skin, and ensures spatial and temporal continuity during an animation.

Future work includes the modeling of other categories of lavas, such as block lavas for which an extra animation layer could be used for animating large lava-crust blocks carried by the flow, or obsidian flows whose skin behaves as a smooth deformable surface that folds and rolls during motion. The detailed modeling of a lava front, with for example liquid parts spreading out of the rigid crust, is also a challenging work to be done.

Acknowledgments:

We wish to thank Mathieu Desbrun, whose research work on the animation of highly deformable substances served as a basis for this work. Thanks to Pierre-Olivier Noirey for integrating Mathieu's work on our animation platform *FABULE*, and to Eugenia Montiel for re-reading this paper. Many thanks to Eric Ferley for whose practical help was determinant for making the whole thing work.

6 References

[1] J. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, July 1982.

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

[3] Jean-Louis Bourdier. *Le Volcanisme*. Editions BRGM (manuels et méthodes).

[4] B. Chanclou, A. Luciani, and A. Habibi. Physical models of loose soils dynamically marked by a moving object. In *Computer Animation Conference*, pages 27–35, June 1996.

[5] M. Desbrun and M.P. Gascuel. Animating soft substances with implicit surfaces. In *SIGGRAPH'95 Conference Proceedings*, pages 287–290, August 1995. Los Angeles, CA.

[6] M. Desbrun and M.P. Gascuel. Smoothed particles: A new approach for animating highly deformable bodies. In *7th Eurographics Workshop on Animation and Simulation*, pages 61–76, Poitiers, France, September 1996.

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

[8] N. Foster and D. Metaxas. Modeling the motion of a hot, turbulent gas. In *SIGGRAPH '97 Conference Proceedings*, pages 181–189, August 1997.

[9] A. Fournier and W.T. Reeves. A simple model of ocean waves. In *SIGGRAPH '86 Conference Proceedings*, pages 75–84, August 1986.

[10] Patrick Fournier, Arash Habibi, and Pierre Poulin. Simulating the flow of liquid droplets. In *Graphics Interface'98*, pages 133–142, May 1998.

[11] J.D. Gascuel, M.P. Cani, M. Desbrun, E. Leroy, and C. Mirgon. Simulating landslides for natural disaster prevention. In *9th Eurographics Workshop on Computer Animation and Simulation (EGCAS'98)*, September 1998.

[12] M. Kass and G. Miller. Rapid, stable fluid dynamics for computer graphics. In *SIGGRAPH '90 Conference Proceedings*, pages 49–57, August 1990.

[13] A. Luciani, A. Habibi, A. Vapillon, and Y. Duroc. A physical model of turbulent fluids. In *6th Eurographics Workshop on Animation and Simulation*, Maastricht, Netherlands, September 1995.

[14] Gavin Miller and Andrew Pearce. Globular dynamics: a connected particle system for animating viscous fluids. *Computers and Graphics*, 13(3):305–309, 89.

[15] J. J. Monaghan. Smoothed Particle Hydrodynamics. *Annu. Rev. Astron. Astrophys.*, 30:543, 1992.

[16] J.F. O'Brien and J.K. Hodging. Dynamic simulation of splashing fluids. In *Computer Animation'95*, pages 198–205, April 1995.

[17] Ken Perlin. An image synthesizer. In *SIGGRAPH '85 Conference Proceedings*, volume 19, pages 287–296, July 1985.

[18] J. Stam and E. Fiume. Depicting fire and other gaseous phenomena using diffusion processes. In *SIGGRAPH 95 Conference Proceedings*, pages 129–136, August 1995.

[19] R. Sumner, J. O'Brien, and J. Hodgins. Animating sand, mud, and snow. In *Graphics Interface*, pages 125–132, June 1998.

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

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