Workshop in Virtual Reality Interactions and Physical Simulation VRIPHYS (2008)
F. Faure, M. Teschner (Editors)

A topology-based animation model for the description of 2D models with a dynamic structure

Pierre-François Léon[1], Xavier Skapin[1], Philippe Meseure[1]

[1]CNRS XLIM-SIC laboratory, University of Poitiers, France

Abstract

This paper presents a model that describes the temporal evolution of 2D-topological structures to represent and control dynamic natural phenomena. As input, the user provides the system with a list of actions that gives a high-level description of the evolution in terms of application-specific operations. As output, a complete representation of the evolution is computed. Our model is composed of three parts: A structural model allowing the temporal representation of both topology and geometry; an event model that aims at detecting topological modifications and ensures consistency between topology and geometry; and a semantic model that simultaneously describes the evolution as a sequence of elementary modifications and manages the history of the various entities of the model. We show the efficiency of the model in the geology field, by studying two well-known phenomena, namely sedimentation and erosion.

1. Introduction

Lots of experimental sciences (biology, botany, geology, etc.) face very elaborated natural structures whose evolution laws are complex and often badly understood. A model able to represent and control structural evolutions, define and reconsider the phenomena and provide the history of entities would be a valuable tool to understand the causes, the formation and the possible evolutions of a structure.

In this article, we propose a way to design such a tool, with a general method to animate topological structures from evolution description. Although we currently work on $2D$ scenes, our method is designed to be dimension-independent. More specifically, we propose a model that represents the evolution of a space subdivision by applying a set of topological modifications at given times. Both structure and geometry are updated along time (in order to generate the successive images of the animation). In addition, the own evolution of each entity of the model is represented in a comprehensive way for the user. Moreover, this model ensures consistency between the geometrical model and the topological model, i.e. not only the result should be visually correct, but it should also provide the user with an accurate representation of the underlying structure. For example, if two edges intersect with each other during the animation, the model has to be updated to take the result of the intersection into account (in other words, secant edges are prohibited in the model). This update is automatic and depends on both the application and the local context. The structural modifications are driven by the geometry and some application-dependent rules.

To put our model into practice, we have chosen to study a kind of geological application, the channel creation (Figure 1), for several reasons: First, this type of geological phenomenon has not, to our knowledge, been studied in the topology-based animation field yet. Second, channels can naturally exhibit a wide variety of shapes, so we can test our model in order to represent as many channels as possible. Third, channel creation is the combination of only two well-known phenomena, namely sedimentation and erosion. Therefore, after defining these phenomena, only a small number of parameters are necessary to create channels. Notice that combining sedimentation and erosion may lead to some topological modifications. For example, in figure 1, an erosion (symbolized by an arrow) is "digging" the geological layer $C1$. Later, erosion could either separate $C1$ from $C2$, or divide $C1$ in two non-adjacent blocks. The matching topological modifications are "sliding" and "interface separation" (Figure 2a), and "face split" (Figure 2b).

This paper is organized as follows: Section 2 first explains the limits of current topology-based animation systems, then

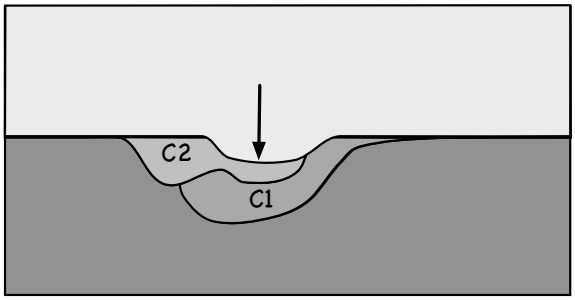

Figure 1: *Example of a channel resulting from successive erosions and sedimentations.*

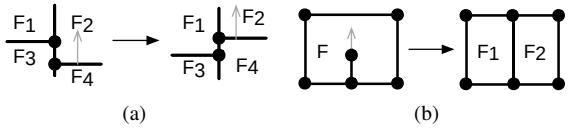

Figure 2: *Examples of topological changes. (a) Vertex sliding over another vertex with topological changes of adjacency relationships between the four faces. (b) The vertex motion leads to split face F into F_1 and F_2.*

presents the general approach of our model. Section 3 describes the different components of our model. Section 4 illustrates its use with 2D channels creation. Finally, we conclude in Section 5 and give some perspectives.

2. Topology-based animation

2.1. Related work

Several systems can represent dynamic structured models. L-Systems represent a formal grammar used to model the growth and the proliferation processes of plants and bacteria [PL90]. This grammar is composed of an alphabet V, a set of constants S, and a set of production rules P describing the system evolution. Variations of the model focus on the selection of rules (using context, conditions, probability, etc.). Map-L-systems apply the principle of L-system to graphs [PL90].
The vertex-vertex systems [Smi06] rely on non-oriented graphs where nodes represent the vertices of the structures, and links represent a permutation of edges around vertices. The structures evolve from a set of modifications applied over time.
Another approach [GTM*05] uses a topological model evolving over time like L-Systems, but works on volumes rather than edges for the internal growth of wood. The growth is driven by the application of a series of rules chosen within the context. Similarly to L-Systems, transformations mainly consist of hierarchical subdivisions of meshes. Those approaches rely on programming language theory, and can represent the evolution of linear structures.
MGS [GM01] is a programming language of structure transformations based on a system of rewriting rules and can be seen as a general framework encompassing most topology-based dynamic models, but does not provide any new dynamic structure.

All these previous approaches exhibit the same kinds of issues. First they are not intended to generate an animation, but can be rather seen as a modeling process, that is, a succession of construction operations. The result is not a representation of the continuous evolution of the system and is merely a series of evolution steps. Indeed, the transformations are seldom defined over time. On the contrary, our approach is based on an animation system. Second, neither the topological nor the geometric consistency is taken into account. If not controlled, an evolution can lead to major incoherence (for instance, when two or more parts of the structure overlap). The transformation rules are only intended to make the structure more complex and not to deal with geometric or topological problems. In our system, the transformation rules not only control evolutions, but also allows the system to react to collisions or topological changes in an adapted way, by means of an event detection. The events are handled according to geometry, topological structure and application context. Third, the aforementioned systems offer poor control over the result. If a problem occurs (a collision), the rules have to be changed in a way which is not intuitive and usually requires some expertise. Furthermore, if a given structure is expected as a result, this implies specific rules that are often difficult to design. On the contrary, our system relies on *business* evolution primitives, that is high-level transformations which are specific to the application field.

2.2. Our Approach

Our approach consists in animating a space subdivision. It relies on a topological model, the generalized maps, integrated in a temporal structure. More precisely, a first component, the structural model, describes an animation by a sequence of generalized maps, where each map represents a set of simultaneous and (supposedly) instantaneous topological modifications. To choose the time where a new map is needed and, more generally, to ensure that the topological model preserves its consistency when the scene entities are moving, an event manager handles every topological modification.
A semantic model completes our animation model by providing an entity designation system, useful for the final user to describe the scene with a high-level language dedicated to the application. For example, to create a channel, a geologist manipulates some parameters on entities called “subsoil layers” and phenomena called “erosion” and “sedimentation”. All the animation steps defined by the final user are gathered into a list of actions that are translated into basic-level events by the semantic model. The event model interprets

these events and updates the structural model by generating a sequential set of transformations applied to an initial generalized map.
In order to help the user to analyze every step of the generated animation, the designation system is hierarchical, so, by means of its name, we know for each entity the entities where it comes from. Moreover, our model records the whole set of local modifications applied to the entities (it is useful to analyze the outcome of the generated animations step by step).

3. An event model for topology based animation

Our animation model is composed of three main parts, namely a structural model, an event model and a semantic model. The structural model represents the topological and geometrical information with an additional temporal dimension. The event model is able to detect and control the topological modifications and modify the structural model. The semantic model associates a name with each edge to designate the different entities of the subdivision following an hierarchical way (inside a 2D scene, designating edges is enough to manipulate the incident faces and vertices). In this section, we focus on those three parts.

3.1. Structural model

For an accurate neighborhood representation, the structural model must rely on a topological model. Many models are able to represent such structures [Edm60, May67, Wei88, Bri89, Lie94]. We choose the generalized maps (n-g-maps) [Lie94], because the n-g-maps are defined in an homogeneous way in any dimension, that makes the definition of both the model and the operations easier, and will make our animation model extensible to higher dimensions. First, we recall the principles of this structure. Next, we show how we use n-g-maps in a sequential structure described below.

3.1.1. Topological structure

The n-g-maps represent objects by their borders (B-Rep). They model quasi-manifolds, oriented or not, with or without boundary. Geometric objects are subdivided in cells (vertices, edges, faces, etc.) linked together by adjacency / incidence relationships (Figure 3).

An n-dimensional generalized map is a set of abstract elements, called darts, and functions defined on these darts:

Definition 1 Generalized map. Let $n \geq 0$. A n-dimensional generalized map (or n-G-map) $G = (D, \alpha_0, \ldots, \alpha_n)$ is defined by:

- D a finite set of darts;
- $\forall k,\ 0 \leq k \leq n$, α_k an involution on D;
- $\forall k, j,\ 0 \leq k < k+2 \leq j \leq n$, $\alpha_k \alpha_j$ is an involution.

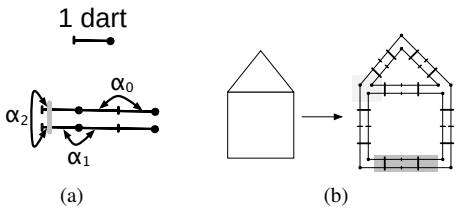

Figure 3: *(a) Graphic convention used to represent darts and links. (b) Example of two faces sticked together and their representation with a closed 2-g-map. In light grey, an example of vertex orbit (0-cell), in dark grey an example of edge orbit (1-cell).*

Let G be an n-G-map, and S be the corresponding subdivision. Intuitively, a dart of G corresponds to an (n+1)-tuple of cells $(c_0, \ldots, c_n)$, where c_i is an i-dimensional cell that belongs to the boundary of c_{i+1}. α_i associates darts corresponding with $(c_0, \ldots, c_n)$ and $(c'_0, \ldots, c'_n)$, where $c_j = c'_j$ for $j \neq i$, and $c_i \neq c'_i$ (α_i swaps the two i-cells that are incident to the same $(i-1)$ and $(i+1)$-cells). When two darts b_1 and b_2 are such that $b_1 \alpha_i = b_2 (0 \leq i \leq n)$, b_1 is said $i-sewn$ with b_2.

Cells are implicitly described as sets of darts through the notion of orbit.

Definition 2 Orbit and i-cell. Let $\{\prod_0, \ldots, \prod_n\}$ be a set of permutations on D. The orbit of an element $d \in D$ related to this set of permutations is $\langle \prod_0, \ldots, \prod_n \rangle$, where $\langle \prod_0, \ldots, \prod_n \rangle$ denotes the group of permutations generated by $\prod_0, \ldots, \prod_n$. Let $d \in D$, $N = \{0, 1, \ldots, n\}$ and let $i \in N$. The i-cell incident to d is the orbit:

$$\langle \rangle_{N-\{i\}}(d) = \langle \alpha_0, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n \rangle (d)$$

A 0-cell is a vertex, an 1-cell is an edge, a 2-cell is a face, and so on.

Definition 3 Closed n-G-map. Let $n \geq 0$. A n-dimentional generalized map (or n-g-map) $G = (D, \alpha_0, \ldots, \alpha_n)$. G is closed $\Leftrightarrow, \forall i \in \{0, \ldots, n\}, d\alpha_i \neq d$.

3.1.2. The temporal model

Our model is also temporal since it represents the animation of structured objects as a sequence of topological modifications (Figure 4). We consider that modifications are instantaneous and can be simultaneous. Our approach is inspired by the key frame animation method [Par01]. However, whereas the original key frame method aims at providing convenient interpolations to generate in-between images, a key frame is introduced in our model only if the structure is subject to one or several topological modifications. If the topology changes at a given time, a new key frame taking these modifications into account is created. Therefore, no topological

modification appears between two consecutive key frames, only embedding is modified.

More precisely, our model is a succession of connected closed n-g-maps, where each new n-g-map is created from the previous one. At time i_{k+1}, the last n-g-map associated with time i_k is cloned and its time is set to i_{k+1}. Next, the new n-g-map is altered by applying all topological modifications happening at this time. This methodology implies to have the set of topological modifications sorted according to time [LSM06].

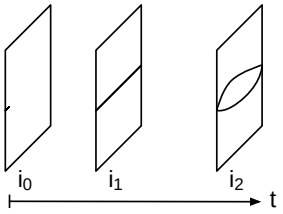

Figure 4: *Temporal model in 2D = sequence of 2-g-maps sorted according to time t.*

The structure detailed above only describes the topological modifications that happen at given times. We need to add some temporal embedding to describe the motion of the entities until the next topological changes. To do so, we use a simple vertex embedding (one may use some higher embedding dimension, such as embedding edges or faces with splines, but collision detection and embedding updates would be more costly in both time and memory). A function $f : t \rightarrow \mathbb{R}^n$ is associated with each 0-cell.

Note that the embedding we have chosen is a temporal function representing positions in space, not in space-time. However, our model is equivalent to a space-time model (i.e. a (nD+t)-dimensional model). Indeed, the former can be obtained from the latter by "temporally slicing" it at times corresponding to each key frame. Between two slices, only the temporal embedding changes, but there is no topological modification. The inverse transformation (from key frames to the space-time model) is done by extruding every n-g-map along the temporal axis, and by joining the resulting volumes (Figure 5).
Although both models are equivalent, our experience shows that it is easier to design transformations and control topological changes using a key frame model than using a continuous space-time structure (where we must deal with many additional but not really useful "temporal" neighborhood relationships). That is the main reason justifying our choice of the key frame approach.

3.2. Event model

The structural model allows us to describe the animation by a sequence of topological and embedding modifications. The

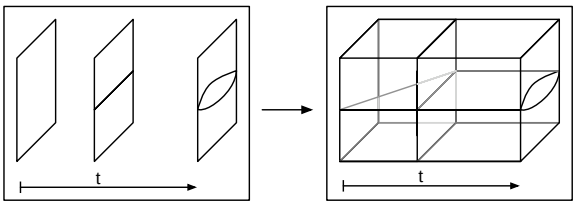

Figure 5: *Transformation from the temporal model to the space-time model by extruding 2-g-maps and joining the resulting volumes using vertex embeddings.*

point is to determine a) the time of each temporal modification, b) which operations are needed to handle the modifications and c) the new embeddings.

To answer question *a)*, a discrete-event system is used. A predictive approach allows us to determine the time of events according to the geometric evolution, i.e. when an entity intersects another one. Events can be generated from various sources. Some of them are generated by the sequence of actions, i.e. the set of animation steps defined at application-level (for instance, when a new phenomenon begins). Other events result from the topological evolution of the system. For example, if a vertex is moving along an edge, an event is triggered when the vertex reaches the edge extremity. Finally, some events result from collisions between initially independent entities. To predict the time at which an event occurs, we use a collision detection approach based on Provot's equations [Pro97]. Every event is completely defined by its date and the set of entities involved.

A discrete-event approach is a well-known paradigm in simulation and has been used in various fields (see [DZ93] for instance). When detected, events are added to a priority queue sorted according to time. The event handling starts by validating the event (i.e. stale events can appear, when, for instance, modifications cancel a previously expected event [DZ93]). If the event is valid, it is handled according to the context of the scene and leads to a sequence of topological, geometric and designation transformations (see the semantic model below). Indeed, the modifications depend on the simulated phenomenon. We use a pattern matching strategy to determine which algorithms must be applied to the structure. The pattern matching search consists in scanning an orbit, trying to get a graph isomorphism (for the structural part) and trying to verify all pattern predicates (that allow to express conditions about geometry or semantics). If no pattern is found, an error is raised. Otherwise, a replacement strategy is used, which appears as an algorithm applied on the identified elements of the pattern. Nevertheless, we are currently trying to define a formal approach that would allow a robust and easier definition of transformations and provide convenient answers to questions *b)* and *c)* (see [GM01] for instance). Note that the transformations are application-dependent and must be supplied by the user.

Those transformations are applied to the current n-g-map if the associated time is the event date, otherwise a new n-g-map is created by cloning the previous n-g-map and associated with the event date.

The embedding transformations include the trajectories of the entities (currently, only 0-cells are considered). When analyzing those motions, new events can be detected and are queued. Therefore, this process is repeated as often as necessary and allows the system to compute the animation.

In a 2*D* animation, to modify the topological model, we use six topological operations, namely *edge creation*, *edge split* (Figure 6b), *vertex identification* (From Figure 7a to Figure 7b), *vertex unidentification* (From Figure 7b to Figure 7a), *edge removal* (Figure 6c) and *edge contraction* (Figure 6d) [DDGLA05]. The *edge creation* operation creates four darts linked together by α_0 and α_2 to form an isolated edge (following the n-g-map model). The *edge split* cuts an edge by inserting vertices inside. This operation is used to oversample an edge. The *vertex identification* operation merges two vertices. This operation is used to link an edge extremity to a vertex. The *vertex unidentification* unlinks an edge extremity and is often used during some intermediate state. The *edge removal* operation unlinks two vertices linked by an edge. *Edge contraction* operation does the same as *edge removal* but also merges both extremity vertices into one.

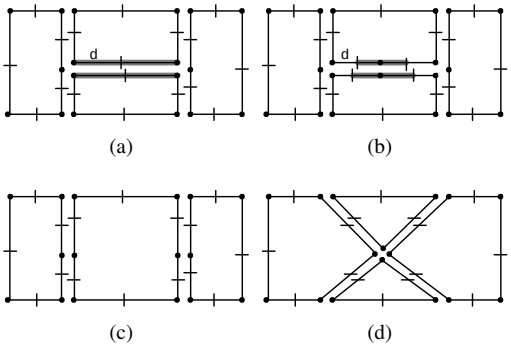

Figure 6: *(a) Initial 2-g-map, topological modification operates on d. (b) Edge spliting, creation of four darts. (c) Edge removing : merging of two faces. (d) Edge contraction.*

3.3. Semantic model

The semantic model has two purposes. First, it must explicitly represent the changes that have been applied to the structure along time. Second, it must represent the history of topological entities. We use a mechanism for entity designation. The designation consists in associating a name with each entity of the scene. Those names are used as parameters for high-level descriptions and for low-level operations. This mechanism relates the high-level operations (used in the sequence of actions), that let the user control the animation, to the low-level (structural) operations upon which the decisions of the user are applied. This designation is hierarchical. For example, inserting n vertices in an edge results in $n+1$ edges whose designations are prefixed by the original edge's designation.

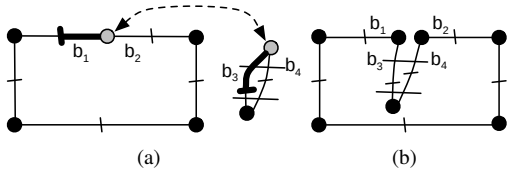

Figure 7: *(a) Initial 2-g-map composed of two connected components : a face and an edge. (b) Identification of the vertex represented by b_1 and the vertex represented by b_3.*

In 2D, only 1-cells (edges) are explicitly designated by an identifier (Figure 8) and we use them to designate vertices and faces. Indeed, since the n-g-map model represents quasi-manifolds, an edge is always incident to two vertices and at most two faces.

Topologically, an edge orbit is made of four darts and its name is assigned to one of them (Figure 8). The designation of 0-cells and 2-cells is done using this dart. Thus, a vertex is designated both by the name of an edge and its position on the edge (*begin*, *notbegin*). A face is designated by the name of an edge from its boundary, and its position with respect to the edge orientation, $left$ or $notleft$.

During the processing of events, the semantic model generates a list of topological operations applied to entities. This list is a sequential list of modifications, i.e. a script that the initial n-g-map undergoes over time. This script outlines the history of the model construction and permits to analyze the outcome of the animation by focusing on the evolution of cells neighborhood relationship through time, for instance.

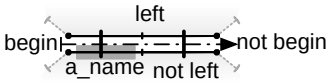

Figure 8: *Designation mechanism of 0,1,2-cells from a dart carrying the designation (here "a_name") of the 1-cell.*

3.4. Animation generation

Figure 9 shows the general animation process. (a-b) The final user (a geologist for instance) provides a list of actions defined in space and time, to describe an application-dependent animation. The sequence of actions is composed of both structural information and operations on designated entities. (b-c) This description is analyzed and converted into a sequence of events. Event processing generates a low-level

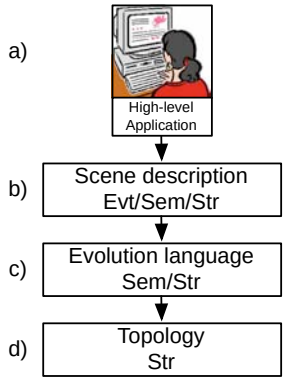

Figure 9: *Use of the three models (event, semantic, structural) in the animation creation process. (Evt: Event; Sem: Semantic; Str: Structural)*

topological evolution script where each topological modification is recorded. This processing uses structural and semantic informations. (c-d) The low-level script (called "evolution language" in the figure, which allows to apply topological operations, independently of the topological model, on designated entities) is interpreted and calls the topological model for processing the low-level operations with the help of the semantic model to bind entities with topological elements. (d) Structural operations are processed.

4. An application in geology

In this section, we show an application of our animation model with a 2*D* case study on two well-known geological phenomena, namely sedimentation and erosion, both involved in the channel construction. After presenting models for sedimentation and erosion, we will describe how they have been integrated into the animation model. For each phenomenon, the general methodology first consists in defining the initial events. These events allow the system to carry out the phenomenon. Other events can appear during the phenomenon and must be defined too. Thereafter, since an event usually implies topology and/or embeddings modifications, these modifications must be defined as well, through patterns and transformation algorithms. Each event requires at least one pattern and the associated transformations. Note that several patterns can be used for the system to handle different cases.

4.1. Sedimentation model

The sedimentation phenomenon is the set of processes by which particles in suspension lay down. According to the principles of stratigraphy enunciated by Stenon (1669), sediments are deposited in approximatively horizontal layers.

Here, we consider a strictly horizontal model of sedimentation (We made this choice to simplify the description of sedimentation. We could use any non strictly horizontal shape by oversampling sedimentation interfaces as we do for erosion model). This sedimentation begins by filling the lowest areas. Next, the upper level rises. In case of several local minimums, two blocks resulting from the same sedimentation merge when they meet in Figure 10, there are two local minimums A and B. The corresponding sedimentations have merged when meeting at C.

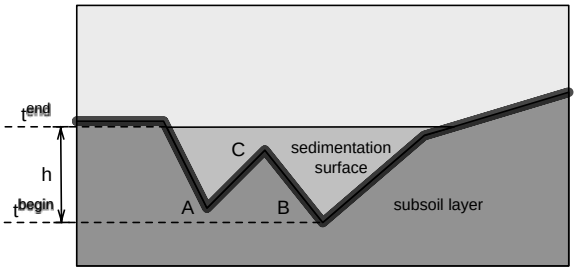

Figure 10: *Parameters of the sedimentation model.*

In the scenario, the sedimentation is defined with five parameters: a prefix to designate the created blocks, a sedimentation area delimited by two edges, two dates *t_begin* and *t_end* for starting and stopping sedimentation, and the sedimentation's height *h*. This simplified model considers that the sedimentation speed is constant over time (i.e. : $v = \frac{h}{t\_end - t\_begin}$).

First, the initializing events must be determined. The system searches for local minimums in the sedimentation area (Figure 11a). Then, the dates at which these minimums start to fill in are computed. *Interface creation* events are then queued for each minimums.

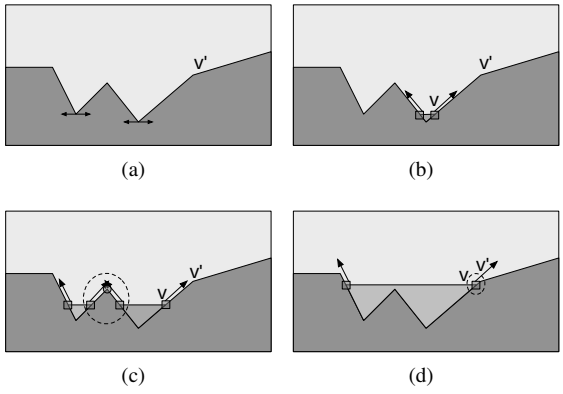

Figure 11: *Sedimentation steps: (a) Searching for local minimums. (b) Creation of the first face and sliding of its extremities. (c) Creation of a new face and sliding of its extremities. Vertex v moves towards a vertex v′ from the subsoil layer. (d) Merging of the two blocks and sliding of the new block extremities. v and v′ collide, then v slides over v′.*

Managing the interface creation event consists in inserting an edge in the sedimentation face at the current minimum area (Figure 11a). Edges extremities are embedded by an interpolation function between the current local minimum coordinates and a target vertex, sliding along the boundary of the subsoil layers with respect to the sedimentation speed.

While the edge extremities slide along an edge, they may reach some vertices. The date of these events, that can be seen as *vertex/vertex collisions*, are computed and stored in the event queue. Those events can be dealt with in two ways depending on the local context. Either both colliding vertices are issued from the same sedimentation phenomenon (Figure 11c) and are merged, implying a face merging (with a process reverse of the one shown in Figure 2b), or one vertex comes from the sedimentation and the other one belongs to a subsoil layer (Figure 11d): In such a case, the first vertex slides over the second one (vertex v) and keeps on sliding along the following edge (as in Figure 2a). The sliding vertex may collide with the other extremity of the edge, so a new vertex/vertex collision event is added to the event model queue. This process is repeated until the end of the sedimentation process.

To distinguish between those cases, the sedimentation process needs two vertex/vertex patterns collisions: one for face merging (Figure 12a) and the other one for vertex sliding (Figure 12b). In both cases, the size s of the edge linking colliding vertices is 0. The first pattern corresponds to the detection of two consecutive null size edges (tagged as "!Sed") incident to two interfaces (tagged as "Sed") coming from the same sedimentation process. The algorithm consists in contracting two null size edges ($n2$, $n3$) and therefore merging two interfaces ($sed01\_0$, $sed01\_1$), leading to the merging of the sedimentation faces. The second pattern corresponds to the detection of one null size edge ($n2$). The algorithm consists in contracting the null size edge and splitting the following edge $n1$ by inserting a new vertex v on $n1$. Next, the sedimentation interface extremity is “unidentified” first then identified with v.

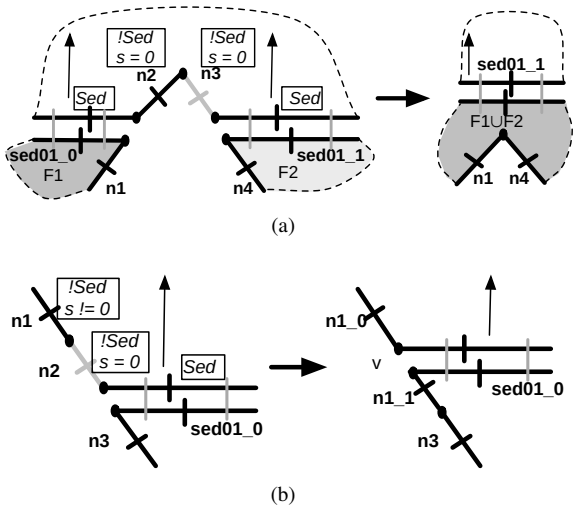

Figure 12: *Collision sedimentation patterns (the current collision is shown in grey, predicates are framed): (a) Merging of two faces $F1$ and $F2$ coming from the same sedimentation process. (b) Sliding of a vertex over another vertex. Dashed lines represent a $(\alpha_1\alpha_0)^*\alpha_1$ link between two darts.*

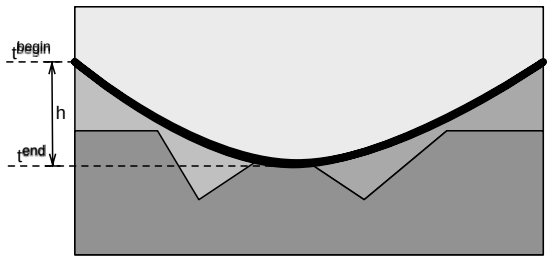

Figure 13: *Parameters of the erosion model.*

4.2. Erosion Model

The erosion is the set of degeneration processes and relief transformations.

To make the description of the phenomenon parameters easier, the erosion is shaped as a curve (in fact, the shape has no incidence on the management of erosion in our model). Erosion begins with deforming the erosion surface. If the erosion surface meets another surface, then the latter is modified following the shape of the erosion surface (Figure 13).

The erosion process is defined by four parameters: the erosion area delimited by two edges, two dates t_begin and t_end for starting and stopping erosion, and height h. To initiate the phenomenon, we oversample the erosion surface (Figure 14a). New vertices are inserted inside the original edge and their geometric embeddings are updated. Their trajectories are computed in order to deform the interface towards a given curve. The initiating events therefore consist in several edge splittings (if needed) and embedding modifications.

During the motion, collisions can appear between the eroded surface and the subsoil layer. When applying the trajectories, those *collision events* are predicted and queued. Handling collision events (Figure 14c) depends on the context. Either the erosion surface has priority over subsoil layers and therefore erodes those layers, or it is split into several pieces after the collision. This notion of priority is explicitly described in [Per98] and adapted to the modeling of geological structures in [BPRS01]. While the vertices of the erosion surface are moving, they can meet other entities such as vertices and edges. Figures 15 and 16 describe the vertex-edge possible collisions and the handling following the nature of

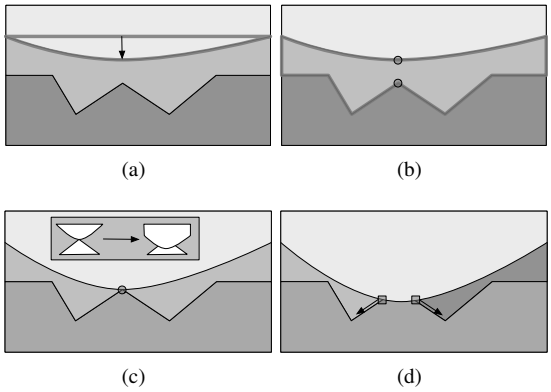

Figure 14: *Erosion steps: (a) Erosion surface oversampling and vertices embedding update. (b) Research for the nearest collision. (c) Processing of the event according to the context, when a collision occurs (here, we assume that the erosion surface has some priority over the subsoil layer). (d) The erosion surface has eroded the layer: A new face is created and the contact vertices slide along the layer below.*

the entities engaged (the erosion surface is tagged as "Ero", the other surface tagged as "!Ero"). Vertex-vertex collisions, where vertices are not linked by an edge, are handled as a particular case of those shown in Figure 16. The embedding of each new vertex is computed to make it slide along initial edges.

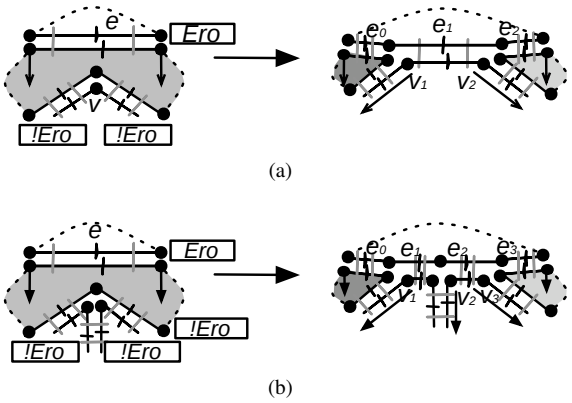

Figure 15: *Collision processing between an erosion edge and one vertex of the subsoil layer seen at topological, geometrical and semantic levels. The arrows show the motion of vertices. Dashed lines represent a $(\alpha_1\alpha_0)^*\alpha_1$ link between two darts. a) A simple case: degree of $v = 2$. b) A more complex case: degree of $v = 3$.*

Figure 15 illustrates the collision processing between an edge belonging to the erosion surface e and a vertex v of the model. If the degree of v is equal to 2 (Figure 15a), two new

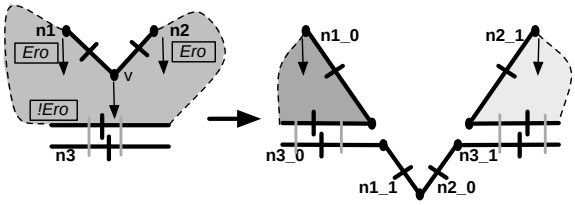

Figure 16: *Collision processing between an erosion vertex and one edge of a subsoil layer. The arrows show the motion of vertices. Dashed lines represent a $(\alpha_1\alpha_0)^*\alpha_1$ link between two darts.*

vertices v_1, v_2 are inserted in e, edges adjacent to v vertex are unlinked, then linked to previously inserted vertices. The embedding of new vertices is updated to make them slide over e. Otherwise, if the degree of v is $n > 2$, we insert n new vertices and we unlink all edges incident to v to link them to the new vertices according to vertex orbit order.

Figure 16 illustrates the collision processing between a vertex v from the erosion surface and an edge e from a subsoil layer. The handling begins with the split of $n3$ by the insertion of a vertex (in the case of vertex-vertex collisions where vertices are not part of the same edge, we simply do not insert a new vertex on $n3$, and the following process is the same). Next, we insert one vertex inside the two edges incident to the vertex v and to the common face between v and $n3$. Then we "*unidentify*" the two new edges of $n3$ and we identify them again with the new vertices inserted in $n1$ and $n2$.

Finally, we must note that after managing collision, contact vertices of the erosion surface can slide along the eroded surface (Figure 14d). This evolution is similar to the slide of vertices along edges described in the case of sedimentation. Therefore, the vertex/vertex collision event can be generated in the erosion case. Figure 17 shows the patterns matching those collisions and their handling.

Pattern 17(a) is similar to the pattern of vertex sliding used about the sedimentation process for the topological modification but slightly different for the embedding computation. Pattern 17(b) represents the disappearance of the edge $n2$ by its contraction to form a new vertex. A new embedding is affected to this vertex, corresponding to the intersection between lines $n3$ an $n1 - n4$.

4.3. Results

In this section we present the results computed by our animation system (see videos at: http://www.sic.sp2mi.univ-poitiers.fr/topanim/). Images of Figure 18 are extracted from our animation player. This software is used to visualize the animation and navigate forward and backward in time. The left part of the window represents

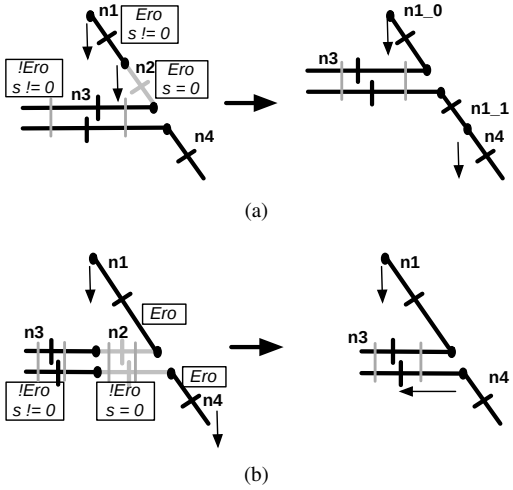

Figure 17: *Collision erosion patterns (in grey the active collision, predicates are framed): (a) Vertex sliding. (b) Edge contraction.*

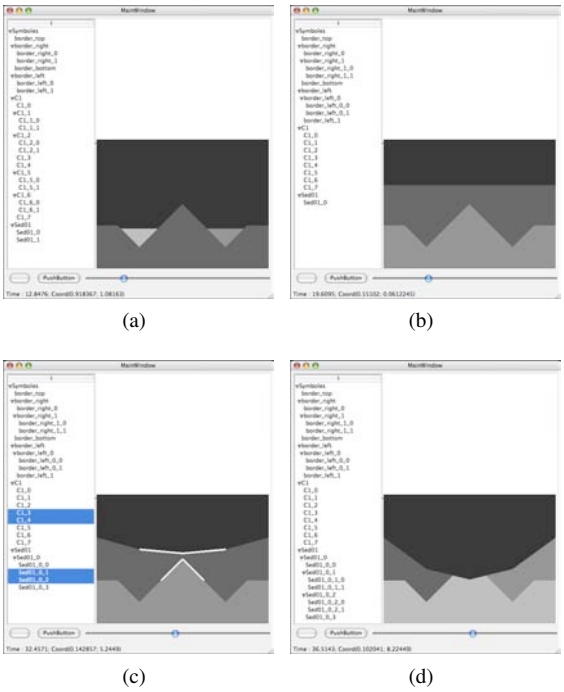

Figure 18: *(a) Channel creation from a predefined shape. The sedimentation begins and creates two separate blocks. (b) Block merging. (c) Beginning of the erosion and detection of the next intersection. (d) Result after processing collision.*

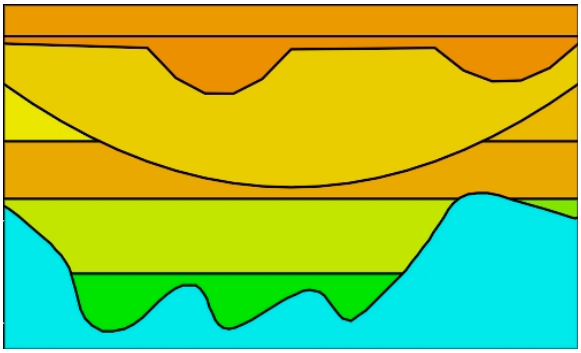

Figure 19: *Result of a more complex script composed of six sedimentations and three erosions.*

the entity designation tree. This tree is updated for each keyframe change. Moreover, at each step of the animation, we can select an entity by its name and see it highlighted in the animation window (as shown in Figure 18c). This selection and the temporal navigation both give the possibility to analyze entity evolutions. Those images also show a channel creation from a predefined shape: A sedimentation begins and creates two separate blocks which merge when they reach the center point. Next, an erosion process begins on the top interface and "grinds" the central point.

The figure 19 shows the end of a more complex animation (48,256 darts divided among 104 frames). The animation begins with four sedimentations followed by an erosion which splits a layer. Then another sedimentation process begins, followed by two simultaneous erosions and a sedimentation. We currently use Maxima (`http://maxima.sourceforge.net/`) as an extern program to solve the non linear equations implied by collision detection. We do not resort to any optimization or detection speed-up. The overall computation time is thus 260s (22s if the solutions of the non linear equations are cached) but this can be improved easily (by using bounding boxes or other similar techniques). This process shows that by using any combination of sedimentations and erosions with controlled parameters, our model lets us create a large range of channels.

5. Conclusion

We propose an animation model based on the dynamic evolution of topological structures. Though our current application is 2*D*, this model is intended to describe the evolution of *nD* structures. It is composed of a sequence of n-g-maps corresponding to a set of instantaneous topological modifications. Each n-g-map is associated with a designation and a temporal embedding. The description of an animation is carried out by user-defined actions that lead to a set of topological events. The collision prediction engine determines which topological events may occur. Each event is processed following its local context and the user-level application. For the time being, only 2*D* components have been developed.

Their use in *nD* is straightforward except for the designation and the required topological and geometric operations.

To illustrate our model, we have taken examples from geological field and defined the channel creation operation through the description of two phenomena, namely sedimentation and erosion. Those phenomena are decomposed into local events of entity creations and collisions. They are currently shaped by means of simple evolution rules without mechanics simulation. In order to generate more realistic animations, we could use a simulation engine to compute geological layer deformations first, and then to provide our animation model with the results of those computations. Since these simulations usually compute the evolution of each block separately, the resulting structures usually exhibit voids between the blocks and/or block overlaps. Our system addresses this issue by ensuring topological consistency between blocks (neighborhood relations are tracked and preserved during the motion).

We aim at improving our sedimentation and erosion models for the building of channels that can be totally controlled by the geologist with additional parameters. We are also working on other geological phenomena such as fault evolution and block sliding, as shown at the aforementioned URI. Next, following the same methodology, we will study other natural phenomena. On the one hand to improve our application, and on the other hand to complete our animation model by taking into account possible events that we have not met with sedimentation and erosion yet. Moreover, we wish to formalize event processing using pattern matching and rewriting system [GM01]. This approach could simplify the selection of actions to process an event according to the local context, reinforcing the independence of our animation model from a specific application.

Acknowledgements

We want to thank M. Perrin (École des Mines de Paris) and J.-F. Rainaud (Petroleum French Institute) for their assessment in geography and their help in describing geological phenomena. The theorical part of this work has benefited of a grant, within the VORTISS project, by the Agence Nationale de la Recherche (ANR-06-MDCA-015-03).

References

[BPRS01] BRANDEL S., PERRIN M., RAINAUD J.-F., SCHNEIDER S.: Geological interpretation makes earth models easier to build. In *EAGE 63rd Conference, Extended Abstacts* (June 2001).

[Bri89] BRISSON E.: Representing geometric structures in d dimensions: topology and order. In *SCG '89: Proceedings of the fifth annual symposium on Computational geometry* (New York, NY, USA, 1989), ACM, pp. 218–227.

[DDGLA05] DAMIAND G., DEXET-GUIARD M., LIENHARDT P., ANDRES E.: Removal and contraction operations to define combinatorial pyramids: application to the design of a spatial modeler. *Image and Vision Computing 23*, 2 (February 2005), 259–269.

[DZ93] DWORKIN P., ZELTER D.: A new model for efficient dynamic simulation. In *EUROGRAPHICS Workshop on Computer Animation and Simulation (EGCAS)* (Barcelone, Sept. 1993), pp. 135–147.

[Edm60] EDMONDS J.: A combinatorial representation for polyhedral surfaces. In *Notices*, vol. 7. Amer. Math. Soc., 1960.

[GM01] GIAVITTO J.-L., MICHEL O.: Mgs: a rule-based programming language for complex objects and collections. *Electr. Notes Theor. Comput. Sci. 59*, 4 (2001).

[GTM*05] GUIMBERTEAU G., TERRAZ O., MÉRILLOU S., GHAZANFARPOUR D., PLEMENOS D.: *Internal wood growth simulation based on subdivised 3D objects*. Tech. rep., Laboratoire MSI, 2005.

[Lie94] LIENHARDT P.: n-dimensional generalised combinatorial maps and cellular quasimanifolds. *International Journal of Computational Geometry and Applications* (1994).

[LSM06] LÉON P., SKAPIN X., MESEURE P.: Topologically-based animation for describing geological evolution. In *International Conference on Computer Vision and Graphics* (September 2006).

[May67] MAY J.-P.: *Simplicial Objects in Algebric Topology*, première ed., vol. 11 of *Van Nostrand Mathematical Studies*. Van Nostrand, 1967.

[Par01] PARENT R.: *Computer Animation Algorithms and Techniques*. Morgan Kaufmann Publishers, 2001.

[Per98] PERRIN M.: Geological consistency: an opportunity for safe surface assembly and quick model exploration. *3D Modeling of Natural Objects, A Challenge for the 2000's 3* (june 1998), 4–5.

[PL90] PRUSINKIEWICZ P., LINDENMAYER A.: *The Algorithmic Beauty of Plants (The Virtual Laboratory)*. Springer, October 1990.

[Pro97] PROVOT X.: Collision and self-collision handling in cloth model dedicated to design garments. *Graphics Interface* (1997), 177–189.

[Smi06] SMITH C.: *On Vertex-Vertex Systems and Their Use in Geometric and Biological Modelling*. PhD thesis, University of Calgary, April 2006.

[Wei88] WEILER K.: The radial edge structure: A topological representation for non-manifold geometric boundary modeling. In *Geometric Modeling for CAD Applications: Selected and Expanded Papers from the Ifip Wg 5.2 Working Conference* (1988), Wozny M.-J., McLaughlin H.-W., Encarnacão J.-L., (Eds.), Elsevier Science, pp. 3–36.