International Journal of Sciences: Basic and Applied Research (IJSBAR)

ISSN 2307-4531
(Print & Online)

https://gssrr.org/index.php/JournalOfBasicAndApplied/index

International Journal of Sciences: Basic and Applied Research
ISSN 2307-4531
(Print & Online)
Published by:
GSSRR
Visit: www.gssrr.org

Topological Model with Adaptive Resolution to Simulate an Incision in Surgery

Noubissi Justin-Herve*

Department of Mathematics and Computer Science, ENSAI-University of Ngaoundere, Cameroon
Email: justherven@gmailcom

Abstract

Surgical simulation covers a set of very large fields. Medical physics simulation, which aims to virtually reproduce the behavior of organic tissues, tries to stick, as far as possible, with reality. This last field poses many concrete problems currently very incompletely resolved, and the extreme mechanical complexity of living tissues. The mechanical interaction between organs, the large number of different tissues and topologies, make the physical simulation of a human organism totally utopian at the present time. The best scientific results for the moment manage to simulate this or that type of organ, with, in terms of interaction, certain simple operations, such as primitive cutting of organs. These organs are modeled by meshes whose resolution depends on the cutting zone: the mesh associated with this zone must be defined at several scales to extract all or part of the organ considered. In addition, taking into account any cuts previously made in the vicinity of the zone considered remains an open problem that needs to be resolved. It leads to modifications of the topological model associated with the mesh. Our study therefore focuses on the adaptive resolution subdivision of triangles and tetrahedra in a mesh. Operations must preserve mesh consistency and must be robust. We propose a topology-based model that meets this need. We define and implement adaptive resolution subdivision operations in a triangular and tetrahedral mesh.

Keywords: Surgical simulation; topological model; geometric modeling; G-map; adaptive resolution.

Received: 6/26/2023
Accepted: 7/30/2023
Published: 8/11/2023

----

* Corresponding author.

1. Introduction

Medical imaging is irreplaceable today, for diagnosis as well as for the preparation and follow-up of surgical interventions. The various techniques that already exist and those that continue to appear regularly make it possible to give a very detailed view of the organs, and increasingly precise information on the pathologies. X-ray, scanner, magnetoencephalography (MEG), electroencephalography (EEG), angiography, these technologies each have their advantages and limitations. In addition to these simulation methods helping the surgeon in his diagnosis and during an operation, it has proven necessary to act upstream of the surgeons' decision by offering tools allowing them to more precisely assess a problem, train before surgery, and even contribute to their training. These are the surgical simulators.

Medical imaging is irreplaceable today, for diagnosis as well as for the preparation and follow-up of surgical interventions. The various techniques that already exist and those that continue to appear regularly make it possible to give a very detailed view of the organs, and increasingly precise information on the pathologies. X-ray, scanner, magnetoencephalography (MEG), electroencephalography (EEG), angiography, these technologies each have their advantages and limitations. In addition to these simulation methods helping the surgeon in his diagnosis and during an operation, it has proven necessary to act upstream of the surgeons' decision by offering tools allowing them to more precisely assess a problem, train before surgery, and even contribute to their training. These are the surgical simulators.

In preoccupying field as surgery, interactive manipulation and realistic rendering of the shape and behavior of organs in the surgical simulator proves extremely useful.

However, the current simulation engines are handicapped by the fact that they suffer from inconsistent and not very robust models with regard to the operations applied to them: Topological models meet this need. In laparoscopy, for example, common interventions involve cutting and ablation. In our work, we talk about subdivision and extraction operations.

In an operation involving the extraction of organs, it is important to know at the right time, in the right place, which part of the body to extract.

The model should therefore be able to allow a recursive cutting of organs, up to the zone deemed necessary for the extraction, and above all, taking into account any cuts that would have been made in the vicinity of said zone.

In this case, we are talking about simulation with adaptive resolution which, among other things, makes it possible to manage the best compromise between precision and speed of calculation which is lacking in current simulators.

Our study therefore consists in proposing a model using the adaptive resolution, and based on the topology to manage the recursive cuts of the parts of a meshed organ in triangles (Dimension 2) or in tetrahedrons (Dimension 3) and reconstructed from geometric data.

2. Survey

The simulation of topological modifications such as cutting of deformable bodies has been the subject of several researches for several years. A few modeling approaches have been proposed, most of them are based on finite element models or on mass-spring models [1]. We classify and examine in this part the different approaches proposed. Various approaches have been suggested for solving the problems of simulating object cutouts. In general, the bodies are most of the time modeled in the form of a mesh made up of unique elements such as tetrahedrons or more complex and different elements. We can classify modeling approaches into three categories.

2.1 Approach by deleting volumes in the model

Cotin addresses in his doctoral thesis [2] the problems of cutting and tearing. After describing the simulated body as a set of tetrahedrons whose behavior is well identified in an autonomous way, he proposes to remove tetrahedrons. By removing a large number of tetrahedrons, we obtain an incision, even a cutting of the object.

The destruction approach proposed by Cotin has the advantage of being simple to implement but has the disadvantage of destroying volume, therefore matter, and therefore causing a loss of mass.

2.2 Approach by separation of adjacents volumes

This method preconizes the separation of two adjacent volumes or a set of adjacent volumes, which causes an incision. In practice, this amounts to split the common face of the two neighbors. The initial face was internal, the two final faces are generally on the surface of the body.

This solution, simple to implement, suffers from the fact that the incision must necessarily be operated on an area between two volumes, but not inside the volumes themselves.

This leads to a lack of precision during the cutting simulation. The final incision follows borders of the mesh, with undesirable effects such as stair steps for example, as illustrated at Figure 1.

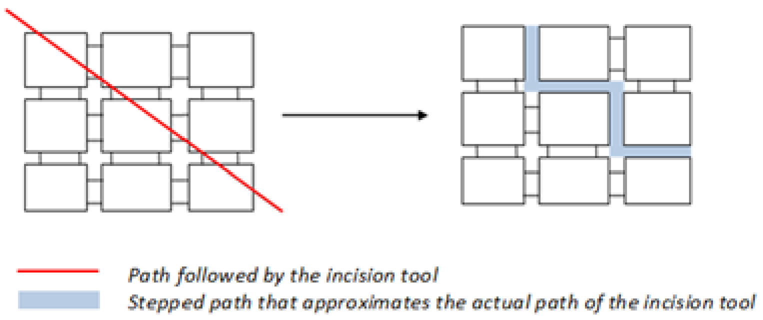

Path followed by the incision tool
Stepped path that approximates the actual path of the incision tool

Figure 1: separation of faces of a mesh object by rectangles and crossed by a tool.

2.3 Approach by cutting elementary volumes

given the shortcomings encountered by previous approaches, some authors seek to make it possible by incising directly through the volumes of the mesh. Bielser [3] worked on the specific problem of cutting a volumic and interactive dynamic model. It uses a tetrahedral mass-spring mesh as a physical model.

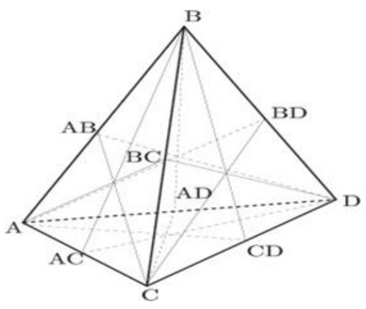


Figure 2: generic "pre-cut" tetrahedron proposed by Bielser [3].

To model cuts, he lists all the possible ways for a tool (a segment for example) to cross a tetrahedron, knowing that the segment can cross the tetrahedron right through or move forward then backward, cutting only one corner of the tetrahedron (case of a half-turn of the cutting maneuver). Inspired by this study, he proposes a data structure based on pre-cut tetrahedron (voir Figure 2). Cette approche assure la conservation de la masse. This approach ensures mass conservation.

Notwithstanding the fact that Bielser's method is quite realistic because it solves the problem of geometric precision, we note however some drawbacks, including the increase in the number of primitives to be simulated, which is a constraint for real-time simulations [4]. Moreover, as tetrahedra are pre-cut, they can only be cut once. We also note the appearance of arbitrarily small volumes leading to mechanical instability after cutting (poorly conditioned equations because the masses involved become lower, the elastic stiffness greater and the cut elements do not have homogeneous dimensions). Moreover, if the mesh is regular, the cutting process does not preserve the nature of the elements (tetrahedra cut into more complex volumes).

[5] proposed a subdivision approach based on point clouds. It preserves the overall geometry of the model, but has the disadvantage of not managing the topological aspect of the models, which is important when we simulate surgical operations.

Although “volume cutting” type approaches have the advantage of providing better geometric precision, they nevertheless have a major problem of instability, and therefore a lack of control. The “removal” type approaches proposed have the advantage of being simple to implement. However, they destroy matter (non-respect of the law of conservation of mass), require the use of fairly fine meshes to remain realistic. “Neighborhood deletion” type approaches lack geometric precision, despite the fact that they are quite simple to implement. Moreover, none of the proposed models offers the possibility of carrying out several modification operations. But if we want a polyvalent model, it must be possible to destroy neighborhood relations to simulate incisions but also to remove elements for the destruction of material. However, these changes must be offered with controllable precision.

Considerations mentioned lead us to explore a new way, and we present in the following section tools that we need to explore that new way.

3. Basic tools of our approach

Our objective is to simulate topological modification operations such as incisions, tears or cuts, while keeping the model we are handling consistent, and therefore, to propose a model with controllable geometric precision. In the following, we present cellular topological models, then the MOKA modeler.

3.1 Topological models

Topological models presented here make possible the representation of subdivided objects, i.s. partitioned into cells of different dimensions : vertices, edges, faces, volumes, etc.

Topological models are classified according to different criteria:

- Cell type :
- cells of regular shape (triangle or rectangle for example);
- cells of any shape.
- Type of cell assembly, which may or may not support multi-incidence. We distinguish in particular the models representing:
- manifolds, orientables or not, with or without boundary;
- Cellular complexes (non-manifold).

- **Glossary :**

We present in this glossary, notions that we regularly manipulate in topological models.

Neighborhood relationship: We use the terms adjacency relation between two cells of the same topological dimension, and incidence relation between two cells of different topological dimensions [6].

Incidence: Two cells C1 and C2 are incident if and only if they are of different dimensions, and $C1 \cap C2 \neq \emptyset$.

Adjacency: Two cells C1 and C2 are adjacent if and only if they have the same dimension i, and if there is a cell C of dimension i - 1 incident to C1 and C2.

Manifold: an n-dimensional manifold can be constructively defined as a set of n-dimensional cells, glued together along the n−1-dimensional cells of their edges, in such a way that an n−1-dimensional cell is incident to at most two n-dimensional cells..

Involution: An involution $\alpha: B \rightarrow B$ is a bijection on a set B which is its own inverse: $\forall b \in B, b\alpha\alpha = b$.

La notation bα signifie que l'on applique l'involution α sur un élément de B. En d'autres termes, bα est identique à α(b).

Embedding: it allows to add informations to various cells..

- **G-map**

Also called generalized maps, they are ordered cellular models based on elements called darts [7]. Their main advantage is that their definition is homogeneous in all dimensions and that they make it possible to represent quasi-manifolds, orientable or not. Cells are implicitly represented by a set of strands linked together by involutions called α. Neighborhood relations and modeling operations are based on darts and α involutions.

Definition 1 (Generalized map):

A generalized map of n-dimension, n≥0 (or n-G-map) is an algebra

$G = (B, \alpha_0, \dots, \alpha_n)$ *défined by:*

- *B is a finite set of abstract objects called darts ;*
- $(\alpha_i), i = 0, \dots, n$ *are involutions on B ;*
- $\forall i, j$ *tels que si* $0 \leq i \leq i + 2 \leq j \leq n, \alpha_i\alpha_j$ *is an involution.*

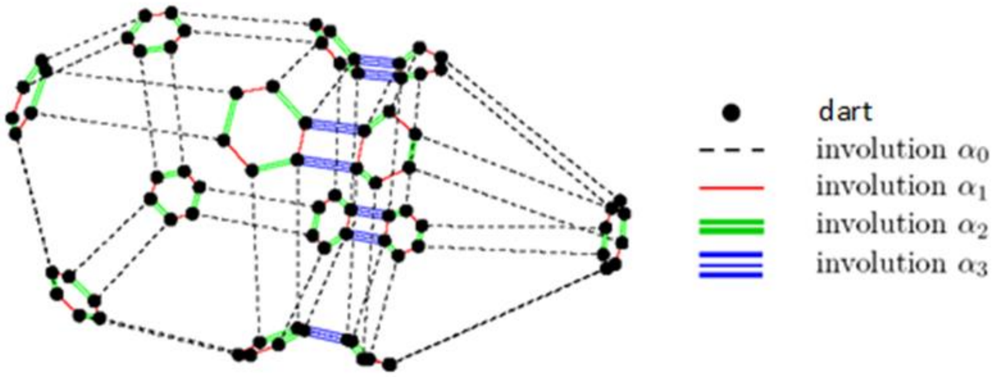


Figure 3: exploded representation of a 3-G-Map.

The notion of cell of dimension *i* can be found in an n-G-carte through the more general notion of *orbit*.

Definition 2 (Orbit):

Let $G = (B, \alpha_0, \dots, \alpha_n)$ *ane n-G-map, b∈B a dart anda sub-set* $\{\alpha 1', \dots, \alpha k'\} \subset \{\alpha 0, \dots, \alpha k\}$.

The orbit $<\alpha_1, \dots, \alpha_k>(b)$ *is the set of darts b' such that there exists any composition c of* $\alpha_1', \dots, \alpha_k'$ *such as bc=b'.* This characterizes the set of reachable darts from b by composition of involutions of $\{\alpha_1, \dots, \alpha_k\}$.

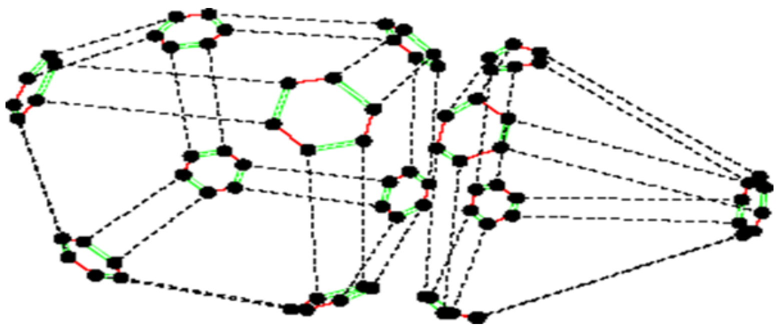


Figure 4: volumes (orbit $<\alpha_0, \alpha_1, \alpha_2>$).

3.2 MOKA : A topologically based geometric modeler

Developed by Frédéric Vidil and Guillaume DAMIAND from XLIM-SIC Laboratory [8], MOKA is a 3D topological based geometric modeler. This modeler is based on the generic 3-G-map kernel. It allows the creation and manipulation of 3D objects using many operations. This kernel is generic because the different applications that can use it all have different specific needs. It therefore allows the modeling of quasi-manifolds, orientable or not, of dimension less than or equal to three. It was developed in C++, in order to obtain an easily modifiable and extensible code.

We briefly explain the overall structure of this kernel. It includes three main classes :

- the *Gmap* class is the base class for declaring a 3-G-map,
- the *Dart class* represents a dart,
- the Attribute class represents an attribute associated with a particular orbit.

Note that the attribute associated with an orbit can be geometric, like the coordinates of a 3d point that we associate with a vertex orbit, or a 2-G-map that we associate with a face embedding, but also other attributes of color, texture. . . This kernel makes it possible to associate any type of attribute with any orbit of the 3-G-map. A specific attribute is associated with a particular orbit. Several different attributes can be associated with the same orbit. For example, we can associate a face with a 2D surface, a color and a texture. These attributes are grouped in the *Embedding* class. Figure 5 shows the schema representing these classes, and how they are related. On this UML diagram, we have represented only the main fields so as not to go into technical details.

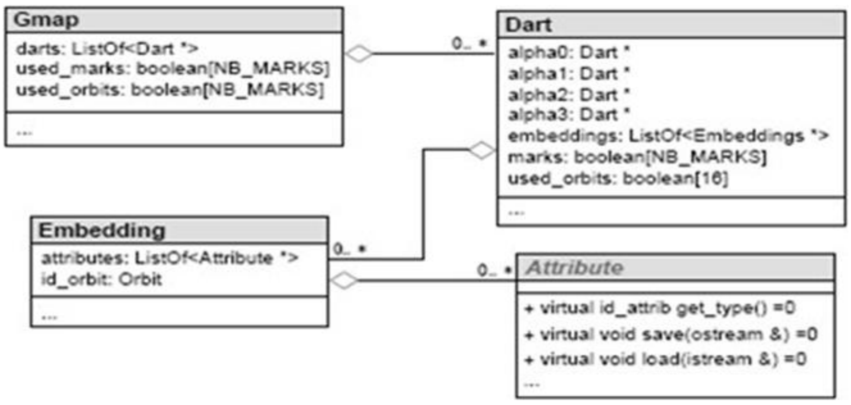

Gmap
darts: ListOf<Dart *>
used_marks: boolean[NB_MARKS]
used_orbits: boolean[NB_MARKS]
...
0..*
Dart
alpha0: Dart *
alpha1: Dart *
alpha2: Dart *
alpha3: Dart *
embeddings: ListOf<Embeddings *>
marks: boolean[NB_MARKS]
used_orbits: boolean[16]
...
Embedding
attributes: ListOf<Attribute *>
id_orbit: Orbit
...
0..*
0..*
Attribute
+ virtual id_attrib get_type() =0
+ virtual void save(ostream &) =0
+ virtual void load(istream &) =0
...

Figure 5: The UML schema of the 3G-map core, partial representation.

- The *Gmap* class is mainly composed of a set of darts, but also has other attributes. The field *used_orbits* is an array of 16 booleans, which allows each orbit to know if it is immersed or not. This makes it possible not to perform a journey in order to look for an embedding, when we know that this orbit is not embedding. Finally, the *used_marks* field gives the boolean marks that are in use. The *Gmap* class has methods for requesting and reserving a free tag, and releasing a tag. Both of these methods will use and update the array. This class has a large number of methods, making it possible to create or delete strands, to sew or unsew these strands, with or without updating embeddings, methods to assign, delete or retrieve an Attribute for a particular orbit. Other methods make it possible to test whether a strand is marked, to mark it or to demarcate it. These are the main methods that the user can call. There are of course several other methods, some of which are private and used internally to, for example, update the embeddings, or test if two strands belong to the same orbit.
- The *Dart* class has four pointers to represent the four α involutions in 3D and a list of *Embedding* carried by this dart. Each *Embedding* corresponds to a particular orbit. The boolean array *used_orbit* lets you know in *O(1)* if this dart carries an Embedding for an orbit. This avoids browsing the Embedding list unnecessarily. Finally the marks array contains the boolean marks of this dart.

- The *Embedding* class contains an *id_orbit* field corresponding to an identifier of the orbit corresponding to this Embedding. It then contains the list of Attributes contained in this Embedding.
- The *Attribute* class is a pure virtual class. The user desiring a particular attribute will create its class deriving from Attribute, and set its "behaviour". It is necessary, among other things, to give a different identifier to each Attribute, to define the Save and *Load* methods. This allows for example to write a generic save method in the *Gmap* class, which will use the *Save* method of the Attribute classes redefined by the user.

We do not detail more about functionalities of this kernel. Its main asset is to be very generic. In addition, it transparently manages the use of any embedding, which allows a non-specialist user to use it without worrying about this aspect. Its two main flaws are its slowness and the memory space occupied. Indeed, by its genericity, it carries out many tests in order to update all the possible embeddings, and the structure in list of lists for the attributes is expensive in memory space. But this kernel can be considered as a prototype. When a specific application requires a particular embedding, it is possible to specialize the kernel to take this embedding into account, and thus remove tests and attribute lists. We then obtain a specialized kernel, for which we can no longer plunge any orbit, but less expensive in terms of execution time and memory space. This kernel change can be made without any modification of the application sources, simply by keeping the same interface between the general version and the specialized version.

4. Our hierarchical model

In surgical simulation, the operation of organ incisions with a scalpel, for example, is a very complex operation to perform. In our case, given that the organ that we wish to incise is represented by a tetrahedral mesh, it is therefore a question for us to model an operation of subdivision of tetrahedra. This operation can be recursive and vary according to the need, it is for us here to be able to subdivide tetrahedra according to the need, and therefore so that it is defined at different scales and according to the desired zone. We propose in this part a model allowing to refine the resolution of a model to desired levels, by subdivision operation while preserving the integrity of the model.

4.1 Definitions

- **Adaptive resolution :** it designates in our paper the fact of considering several parts of a mesh in a “finer” way than others, to approach the shape of the simulated object as closely as possible. The goal is to choose, at the right time, in the right place, which resolution gives the best compromise between precision and speed of calculation.
- **Object resolution level :** In adaptive resolution, any object is created at a certain resolution level. In this work, we associate this level with the object in the form of a natural whole number.

Nous utiliserons quelques fois le diminutif « **niveau** » pour désigner le niveau de résolution.

4.2 Problem analysis

We want to subdivide a tetrahedron. We propose an approach which will allow us to obtain small tetrahedra resulting from the subdivision of a tetrahedron and which are similar to the tetrahedron from which they come from.

A tetrahedron being made up of four faces (which are triangles), that faces are therefore concerned during a tetrahedron subdivision operation. It is therefore important to study the behavior of the triangles representing the faces of the tetrahedron.

Suppose we want to subdivide a triangle into several triangles so that the resulting small triangles are similar to the triangle they come from. We can inspire by Loop diagram [9] as shown in Figure 6.

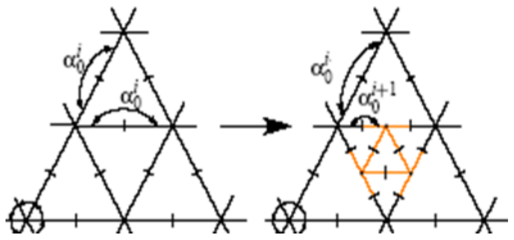


Figure 6: Loop schema.

We get 4 triangles from the initial triangle as shown in Figure 7.

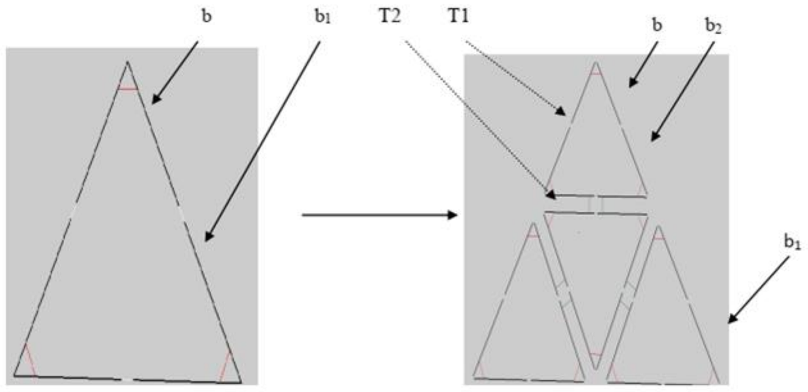


Figure 7: subdivision of a triangle.

Subdividing a triangle would therefore amount to subdivide each of its edges into two, and inserting new ones. We find that initially, $b\alpha_0=b_1$. After a subdivision of the triangle, $b\alpha_0=b_2$. Dart b would therefore have two images by the involution α_0. This subdivision operation can be recursive, and so one dart could have a different image depending on the resolution involution.

Suppose now that we want to subdivide one of the triangles resulting from the subdivision of our initial triangle as illustrated in Figure 8.

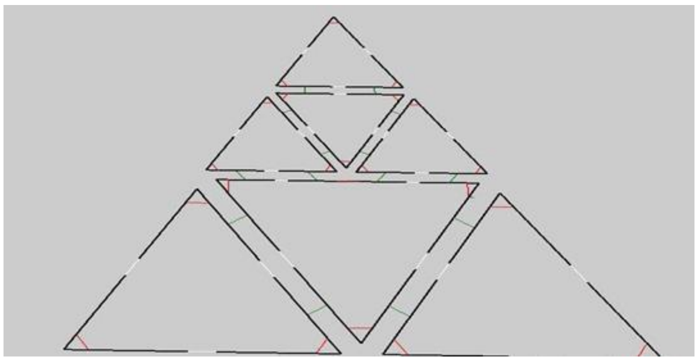


Figure 8: subdivision of the triangle at level 2.

The subdivision of the triangle T1 is not without topological repercussions on the triangle T2. In particular, this subdivision causes the subdivision of one of the edges of the triangle T2. We therefore observe that in a triangle mesh, the subdivision of a triangle has repercussions on the neighboring triangles. Recall that this subdivision operation is recursive. A similar observation is made when we want for example to subdivide a tetrahedron into several tetrahedra.

It is therefore a question for us thereafter of proposing a topological model making it possible to refine the resolution of a model at the desired levels, by operation of subdivision; this subdivision must be applicable several times to the model, in a recursive way. The integrity of the model must be preserved, which implies, in particular, a management of topological neighborhoods in accordance with the resolution.

4.3 Adopted approach

Given that a tetrahedron is made up of four faces (triangles) and that each face is made up of three edges, each edge being made up of two strands, we suggest a strategy of modeling based on the study of the behavior of edges during a subdivision operation, then face, and finally tetrahedron.

4.3.1 Edge adaptive resolution subdivision

Our problem boils down to the subdivision of an edge at a given level of resolution k. We have seen that a strand could have several images by the involution α0. In order to manage this resolution during an edge subdivision, we define the map μa which associates to each strand the level of resolution of its edge by :

$\mu_a : B \rightarrow N$

$b \in B \longmapsto \mu_a(b) = n_a$, where n_a is the resolution level of the edge of dart b.

Remark Let b be a dart with edge resolution level na. Inserting a vertex on the edge corresponding to strand b induces the creation of two new strands b_1 and b_2 and the initial and new strands of the edge take as edge resolution level n_a+1 such that :

$$b\alpha_0^{(n_a+1)} = b_1; \quad b_1\alpha_1 = b_2; \quad b\alpha_0^{n_a}\alpha_0^{(n_a+1)} = b_2$$

Notation $b\alpha_0^{n_a}$ *means the image of b at level* n_a *by theinvolution* α_0.

Suppose Figure 9, where b' is a dart such that $b\alpha_0^{n_a} = b'$.

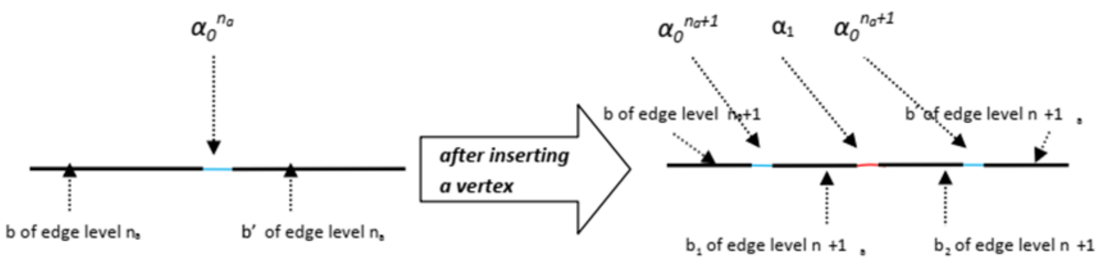


Figure 9: inserting a vertex on an edge.

4.3.2 Face adaptive resolution subdivision

Our problem boils down to the subdivision of a face into several faces at a given level of resolution n_f given. In our case, it is a question of being able to subdivide a triangle into several uniform triangles by taking into account the topological neighborhood relations existing between the triangles.

We subdivide the three edges of the face according to our principle of edge subdivision described previously. By then connecting the vertices thus created by inserting three new edges, we form four new small faces (triangles).

Remark Subdivision of a face induces insertion of new edges to connect the vertices created. In the principle of G-maps, in order to restore vertices, there will indeed be the creation of new edges (which we also call internal edges and their darts are called internal darts, the others being external darts) connected two by two by α_2 involutions, as shown in Figure 10.

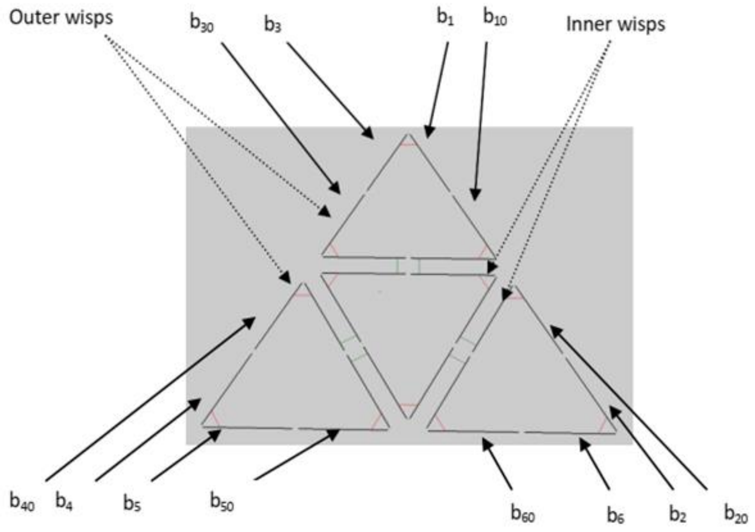


Figure 10: triangle subdivision.

We have seen that the subdivision of a face has influences on the adjacent faces. In this case, the face subdivision is made without really taking into account the adjacent faces and the modifications that these would have undergone over the subdivisions. We must therefore be able to identify a face whose edge had already been subdivided in the case of an adaptation of the adjacent edge.

To overcome this handicap, we then define the map μ_f which, with each strand, associates the level of resolution of its face by :

$\mu_f : B \rightarrow N$

$b \in B \longmapsto \mu_f(b) = n_f$, where n_f is the the resolution level of the face of dart b.

Theorem 1 *Let B be the set of darts of a G-map.*

If μ_a designates the level of resolution of the edge of a dart, and μ_f the level of resolution of the face of this dart, then $\forall b \in B, \mu_f(b) \leq \mu_a(b)$.

In fact, subdivision of a face at level *i* induces the replacement of the face level of its darts by *i*. So for any dart *b* resulting from the subdivision of a face *f*, we have $\mu_f(b)=i$.

However, subdividing a face automatically results in the subdivision of its edges and the insertion of new edges. Subdividing an edge induces the replacement of the edge level of the darts affected by this edge subdivision by *i*. Thus, for any dart b resulting from the subdivision of each edge a, we have $\mu_a(b)=i$. Inserting edges results in the creation of new darts which take as edge and face level i. Thus, for any dart b resulting from the insertion of each edge a, we have $\mu_a(b)=i$.

We deduce that if B is the set of darts of the face, $\mu_f(b)= \mu_a(b)=i$.

Taking into account the neighborhood relations of the faces stated a little earlier in the paper, consider two neighboring faces F1 and F2 of common resolution level i and both having edges of resolution level i. The subdivision of F1 at level i+1 would result in the subdivision of an edge of F2 without F2 being subdivided, and the subdivision of this edge of F2 would result in the replacement of the edge level of the darts affected by the subdivision of this edge of F2 by i+1. The neighboring face F2 would have in this case certain darts of level i+1, whereas its own level is i. We will then have for any dart b resulting from the subdivision of this edge of F2, $\mu_f(b)=i$ *et* $\mu_a(b)=i+1$, we have $\mu_f(b) < \mu_a(b)$.

We deduce that if B is the set of darts of our G-map, $\forall b \in B, \mu_f \leq \mu_a(b)$.

Remarks Let *b* be a dart of an edge A, and *i* a given level of resolution. Edge *A* is not subdivided at level *i* if $\mu_a(b) < i$.

- Any edge insertion or subdivision induces the modification of the darts affected by this insertion or subdivision. Indeed, during the insertion or the subdivision of an edge at a level *i*, new strands are created, and certain strands of levels *i-1* are affected. It is therefore necessary to modify the vectors of image strands of all the strands concerned by the insertion or the subdivision of the edge by adding to it at position i, the strand image by α_0 of each of these strands. Moreover, taking into account the face adjacency relations stated above, the darts of the adjacent faces by α_2 et α_3 undergo the same modifications if they had not yet been subdivided at level *i*.

- When a face is subdivided at level *i*, all inner darts take as face and edge resolution level *i*. The outer darts of the resulting level *i* faces take as face resolution level *i* if they had not yet been subdivided at level *i* (i.e. if their current face level is less than *i*), and as edge resolution level *i* if their current edge level is less than *i*. Taking into account the face adjacency relations stated above, the darts of faces adjacent by α_2 take as edge resolution level *i* if their current edge level is less than *i*. Edges of faces adjacent by α_3 take edge resolution level *i* if their current edge level is less than *i*, and face resolution level *i* if their current face resolution level is less than *i*.

4.3.3 Tetrahedron adaptive resolution subdivision

The problem boils down to the recursive subdivision of a tetrahedron into several uniform tetrahedra, the subdivision operation being able to be repeated several times. It is therefore for us to perform the subdivision of a tetrahedral mesh.

To achieve this tetrahedron subdivision, we start by subdividing the four faces of our tetrahedron by applying our face subdivision method stated above as shown in Figure 11.

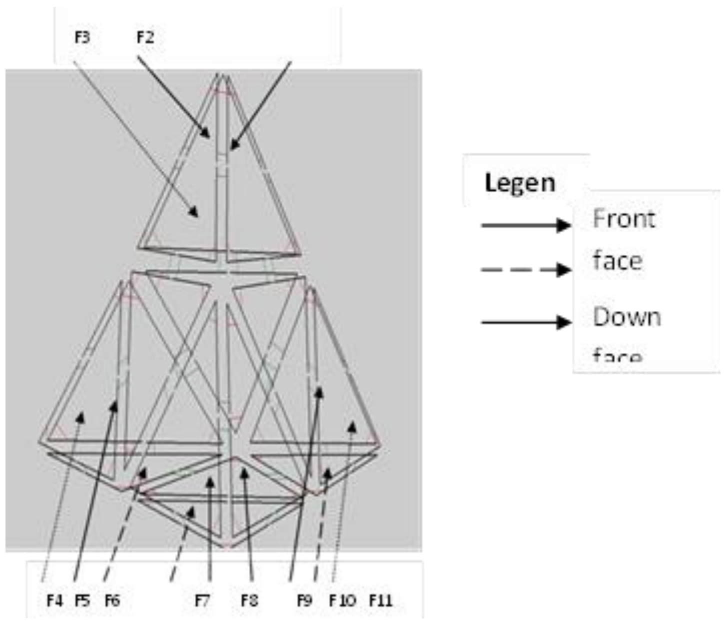


Figure 11: Subdivision des quatre faces d'un tétraèdre.

Once the four faces of the tetrahedron have been subdivided, the new problem is how to obtain several uniform tetrahedra. For this, we are inspired by the *Sierpinsky tetrahedron*, as illustrated in Figure 12.

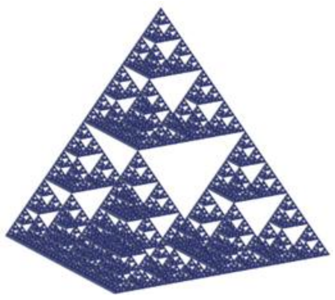


Figure 12: sierpinsky tetrahedron according to [10].

Although the considered approach seems interesting, it has however a drawback for our field which is surgery. Indeed, to proceed as Sierpinsky would lead us to a mesh in tetrahedra, but leaving at each subdivision, a hole on each face. We propose to modify Sierpinsky's approach. A first approach consists in "filling up" the holes and obtain four tetrahedra and an octahedron as illustrated in Figure 13.

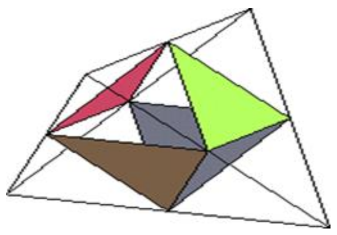


Figure 13: Tetrahedron subdivided into four tetrahedra and one octahedron.

However, it is first necessary to unsew and sew faces in order to obtain the four tetrahedrons. Then, you have to insert faces in order to "plug" holes and obtain the octahedron. It is then necessary to subdivide the octahedron into tetrahedrons, by inserting faces. We propose a faster method by avoiding constructing and then decomposing an octahedron. After subdividing the four faces of the tetrahedron such, we decide to isolate from the initial tetrahedron four volumes (F1, F2, F3), (F4, F5, F6), (F7, F8, F9), (F10, F11, F12), and four faces by uncut faces. We obtain the configuration illustrated in Figure 14.

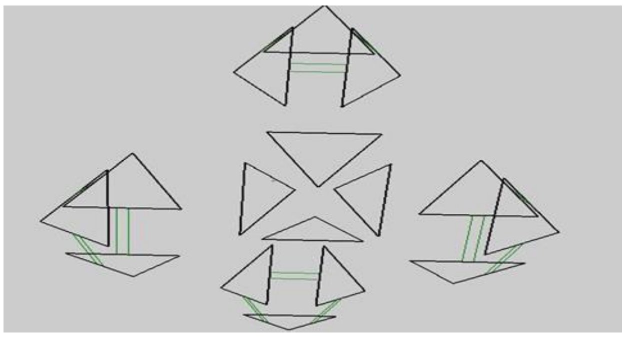


Figure 14: insulation of faces and volumes.

We can then perform face insertions in order to close the four volumes obtained. Inserting faces will allow us to obtain four small tetrahedra and automatically create four other uniform faces with the small faces already created. We can therefore use these four new faces in addition to the four other isolated faces to create four new tetrahedra uniform to the other existing tetrahedrons by using the face stitching operations. We therefore obtain eight tetrahedra resulting from the initial tetrahedron.

However, let us not forget that we must take into account the notion of resolution. We must therefore be able to identify the level of resolution of a tetrahedron.

For this, we define an application μ_t which associates to each strand the level of subdivision of its tetrahedron by :

$\mu_t : B \rightarrow N$

$b \in B \longmapsto \mu_t(b) = n_t$, où n_t est le niveau de résolution du tétraèdre du brin *b*.

A similar comparison to the one stated for the proof of Theorem 1 allows us to state the following corollary:

Corollary Let *B* be the set of darts of a G-map. $\forall b \in B, \mu_t(b) \leq \mu_f(b) \leq \mu_a(b)$.

Remark At each subdivision of a tetrahedron from a dart b of tetrahedron level *i-1*, the level of the strands of the tetrahedra created by i must be modified.

At each insertion of faces at a level i, it is necessary to browse all the inserted faces and replace the edge and face levels of their strands by i.

If an edge has just been subdivided at level i, and its darts have images by α_3 and for each dart b of the edge, if $\mu_a(b\alpha_3)<i$ (and then $\mu_f(b\alpha_3)<i$ by the Theorem 1),

- $b\alpha_3\alpha_0^i = b\alpha_3\alpha_0$, $\mu_a(b\alpha_3)=i$, $\mu_f(b\alpha_3)=i$,
- $b\alpha_3\alpha_0 = b\alpha_3$, $\mu_a(b\alpha_3\alpha_0)=i$, $\mu_f(b\alpha_3\alpha_0)=i$

4.3.4 *Our topological model with adaptive resolution*

We propose an extension to the formalism of G-maps by introducing a notion of hierarchy at the dart level. A dart *b* is therefore also identifiable by a dart level, an edge level, a face level, a tetrahedron level, a list of the different $b\alpha_0^i$ resulting from the subdivision of an edge at a given level i. This extension of resolution consists in keeping all the intermediate meshes of the process of subdivision.

By introducing the hierarchy at the level of darts, it is possible to identify darts according to the level where they were introduced in the map. The level 0 darts are those introduced in the initial g-map and before any subdivision. Level 1 darts are introduced after a subdivision, and so on. More precisely :

If we have B_0 at level 0, we will have B_1 at level 1 such that $B_0 \subset B_1$.

More generally, if *k* is the maximum level of resolution, we will therefore have :

$$B_0 \subset B_1 \subset B_2 \subset B_1 \subset \cdots \subset B_k$$

During a subdivision step, we perform vertex insertions (edge sections), edge insertions or face insertions between the new vertices, so the valence of vertices in the initial mesh is not modified. Thus, an α_1 no dart being inserted in an α1 permutation between two levels.

Formally,

$\forall x \in B_i \backslash B_{i-1}, \alpha_1^j(x) = \alpha_1^i(x)\ pour\ tout\ i < j \leq k,$ *k* is the maximum level of resolution.

Unlike bonds α_i, $i \geq 1$ which do not change after a subdivision step, bonds are modified (not necessarily but only if the edge has been subdivided) and therefore require special consideration. In the example of Figure 14, for darts of B^i, the relations α^{i+1} are not equal to the relations α^i.

We can therefore model the topology of a generalized map extended to adaptive resolution by :

$$G = (Bi, \alpha 0i\}\ i \in [0, k], \alpha 1, \alpha 2, \ldots, \alpha n)$$

Each level of resolution is represented by :

$G_i = (B_i, \alpha_0^i, \alpha_1|G_i, \alpha_2|G_i, \ldots, \alpha_n|G_i)$ *où* $\alpha_j|G_i, j \geq 1$, *is the restriction of* α_j *to the elements of* G_i.

5. Results

Subdivision operations that we implement must respect the standards of the topology. For this purpose, we used the topologically based geometric modeler MOKA. MOKA has routines for inserting a vertex on an edge, inserting faces and edges. We also have a routine for closing a volume. Only, these routines do not take account level of resolution of the objects.

We present in this section, some algorithms that were necessary during the implementation of our subdivision operations. We also present a view of our adaptive resolution subdivision operations..

5.1 Adaptive resolution subdivision of triangles

- **Edge subdivision algorithm.**

Input : A dart $b \in B, i \in N^*$ the current subdivision level.

Output : Dart b_{10} such as $b\alpha_0^i = b_{10.}$

(/*Begin*/)

```
$b_{10} \leftarrow insert\_vertex(b)$ ;

$edge\_level(b) \leftarrow i$ ; $edge\_level(b\alpha_0^{i-1}) \leftarrow i$ ;

$edge\_level(b_{10}) \leftarrow i$ ; $edge\_level(b_{10}\alpha_1) \leftarrow i$ ;

$b\alpha_0^{i} \leftarrow b_{10}$ ;        $b\alpha_0^{i-1} \leftarrow b_{10}\alpha_1$;

$b_{10}\alpha_0^{i} \leftarrow b$;        $b_{10}\alpha_1\alpha_0^{i} \leftarrow b\alpha_0^{i-1}$;

Return($b_{10}$) ;

(/*End*/)
```

We will refer to this function by calling ***subdividedge(b,i)*** ;

We present in Figure 15 an example of subdivision with adaptive edge resolution.

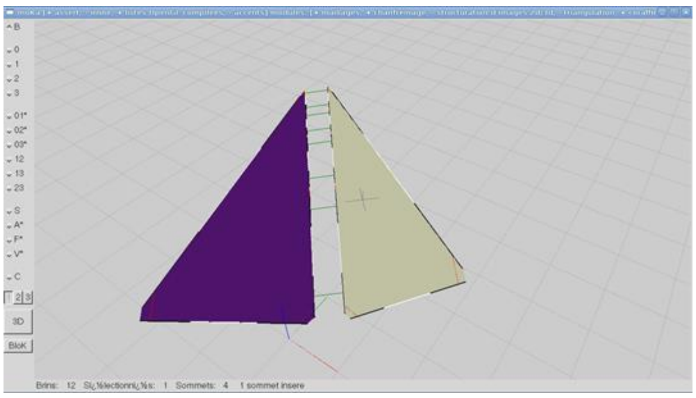

Brins: 12 Sï¿½lectionnï¿½s: 1 Sommets: 4 1 sommet insere

Figure 15: edge subdivision at Level 3.

- **Traingle subdivision algorithm**

Input : A dart $b \in B, i \in N^*$ the current subdivision level.

```
(/*Begin*/)

if first_subdivision(b) then

    $b_2 \leftarrow b\alpha_0$ ; $b3 \leftarrow b\alpha_1$ ; $b4 \leftarrow b_3\alpha_0$ ; $b_5 \leftarrow b_4\alpha_1$ ; $b_6 \leftarrow b_2\alpha_1$ ;

    $b\alpha_0 \leftarrow b_2$ ; $b_2\alpha_0 \leftarrow b$ ;

    $b_3\alpha_0 \leftarrow b_4$ ; $b_4\alpha_0 \leftarrow b_3$ ; $b_5\alpha_0 \leftarrow b_6$ ; $b_6\alpha_0 \leftarrow b_5$ ;
```

```
else
$b_2 \leftarrow b\alpha_0^{i-1}$ ; $b_3 \leftarrow b\alpha_i$ ;   $b_4 \leftarrow b_3\alpha_0^{i-1}$ ;
$b_6 \leftarrow b_2\alpha_1$ ;      $b_5 \leftarrow b_4\alpha_1$;
if not subdivide($b_5$,i) then
$b_{50} \leftarrow$ subdividedge ($b_5$,i);
else
    $b_{50} \leftarrow b_5\alpha_0^i$ ;
if not subdivide($b_3$,i) then
$b_{30} \leftarrow$ subdividedge($b_3$,i);
else
$b_{30} \leftarrow b_3\alpha_0^i$ ;
if not subdivide(b,i) then
$b_{10} \leftarrow$ subdividedge(b,i);
else
$b_{10} \leftarrow b\alpha_0^i$ ;
$b_{60} \leftarrow b_6\alpha_0^i$ ; $b_{40} \leftarrow b\alpha_0^i$ ;   $b_{20} \leftarrow b_2\alpha_0^i$ ;
$bi_1 \leftarrow$ insert_edge($b_{10}$,$b_{30}$);
$bi_2 \leftarrow$ insert_edge ($b_{20}$,$b_{60}$);
$bi_3 \leftarrow$ insert_edge($b_{40}$,$b_{50}$);
$bi_1\alpha_0^i \leftarrow bi_1\alpha_0$;   $bi_1\alpha_0\alpha_0^i \leftarrow bi_1$ ;
$bi_2\alpha_0^i \leftarrow bi_2\alpha_0$;   $bi_2\alpha_0\alpha_0^i$ □ $bi_2$ ;
$i_3\alpha_0^i \leftarrow bi_3\alpha_0$;     $bi_3\alpha_0\alpha_0^i \leftarrow bi_3$ ;
```

```
$bi_1\alpha_2\alpha_0{}^i \leftarrow bi_1\alpha_2\alpha_0$;        $bi_1\alpha_2\alpha_0\alpha_0{}^i \leftarrow bi_1\alpha_2$;

$bi_2\alpha_2\alpha_0{}^i \leftarrow bi_2\alpha_2\alpha_0$;        $bi_2\alpha_2\alpha_0\alpha_0{}^i \leftarrow bi_2\alpha_2$;

$bi_3\alpha_2\alpha_0{}^i \leftarrow bi_3\alpha_2\alpha_0$;        $bi_3\alpha_2\alpha_0\alpha_0 \leftarrow bi_3\alpha_2$;

Addlist(list_darts, b, $b_2$, $b_3$, $b_4$, $b_5$, $b_6$, $bi_1$, $bi_2$, $bi_3$) ;

while non vide(liste_darts)

ChangeLevel_bA2(dart);

Read(next_dart) ;

(/*End*/)
```

The procedure ***ChangeLevel_bA2 (b)*** allows to replace $\mu_a(b)$ et $\mu_a(b\alpha^i)$ by *i* if $\mu_a(b)<i$ and $\mu_f(b)$ and $\mu_f(b\alpha^i)$ by *i* if $\mu_f(b)<i$. If $b\alpha_2$ exists, this procedure also allows to replace $\mu_a(b\alpha_2)$ and $\mu_a(b\alpha_2\alpha_0{}^i)$ by *i* if $\mu_a(b\alpha_2)<i$ and $\mu_f(b\alpha_2)$ and $\mu_f(b\alpha_2\alpha_0{}^i)$ by *i* if $\mu_f(b\alpha_2)<i$.

The ***insert_edge(b$_1$, b$_2$)*** *fonction* creates a new edge from darts b_1 and b_2 and returns a dart of the inserted edge.

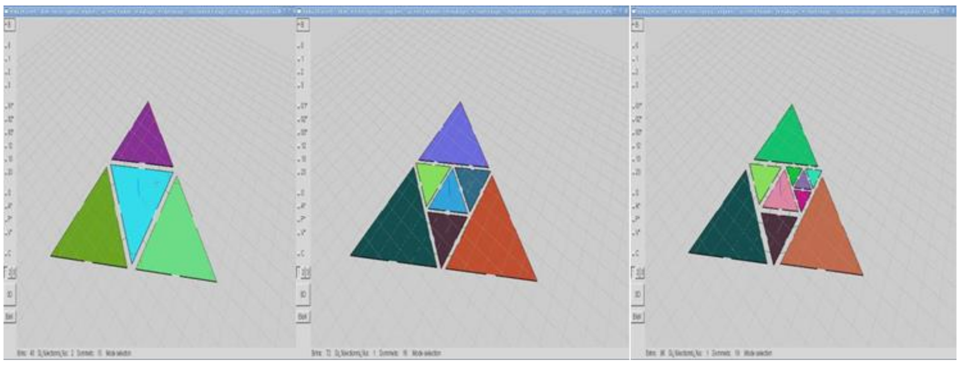


Figure 16: visualization of successive subdivision<s of triangles.

5.2 Adaptive resolution subdivision of tetrahedra

The adaptive resolution subdivision of tetrahedra algorithm is based on the adaptive resolution subdivision of triangles algorithm as we described earlier.

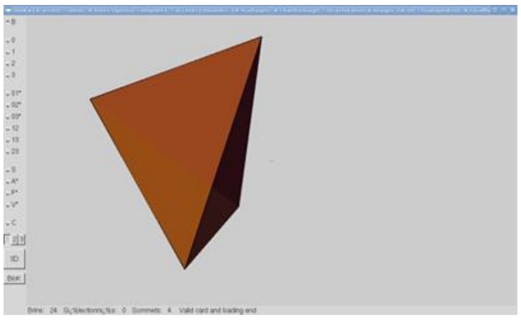


Figure 17: initial tetrahedron.

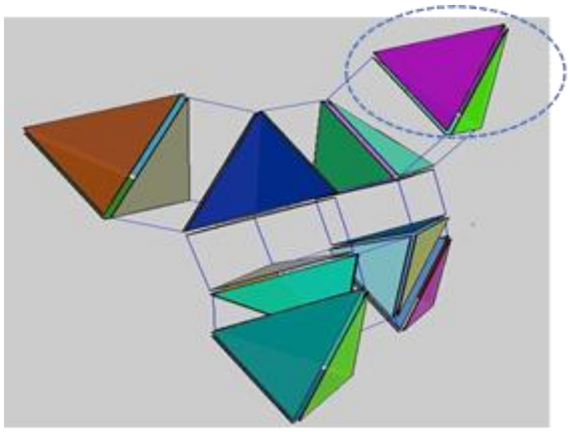


Figure 18: subdivision at step 1, 8 tetraedra.

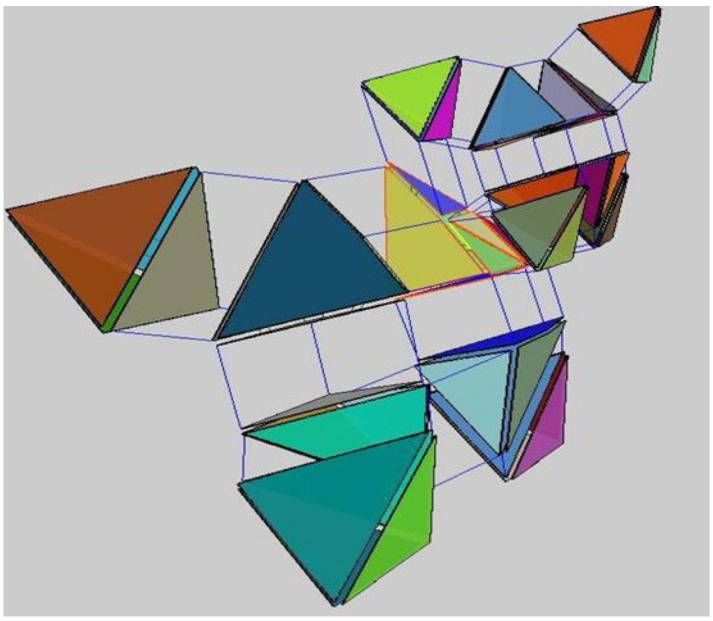


Figure 19: subdivision at step 2 of the tetrahedrom bordered by blue color in Figure 1.

We note in Figure 19 that one face (the face offered by the red color) of a neighboring tetrahedron has been subdivided. Now suppose we want to subdivide the tetrahedron whose face has already undergone a subdivision. Our model will have to take into account the neighborhood relations between the tetrahedrons. Indeed, the model will have to take into account the fact that the edges and faces of certain tetrahedra have been subdivided at level 1 and 2, although the level of the tetrahedron may be 1. Figure 20 illustrates this neighborhood management. We subdivide a current level 1 tetrahedron to level 2.

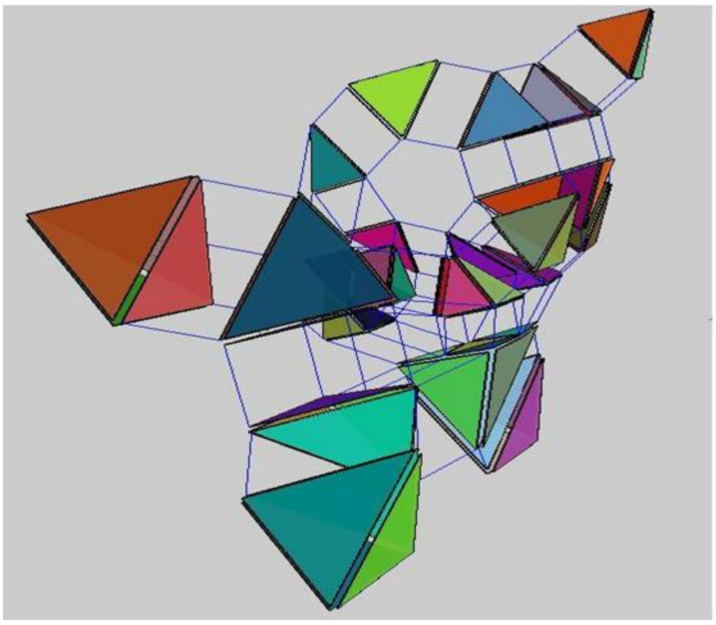
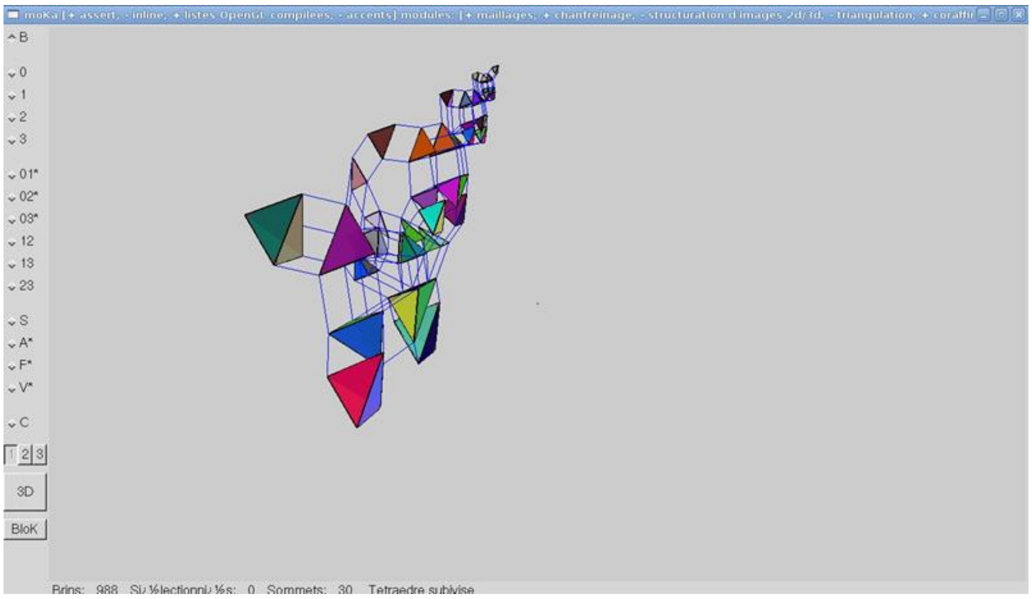


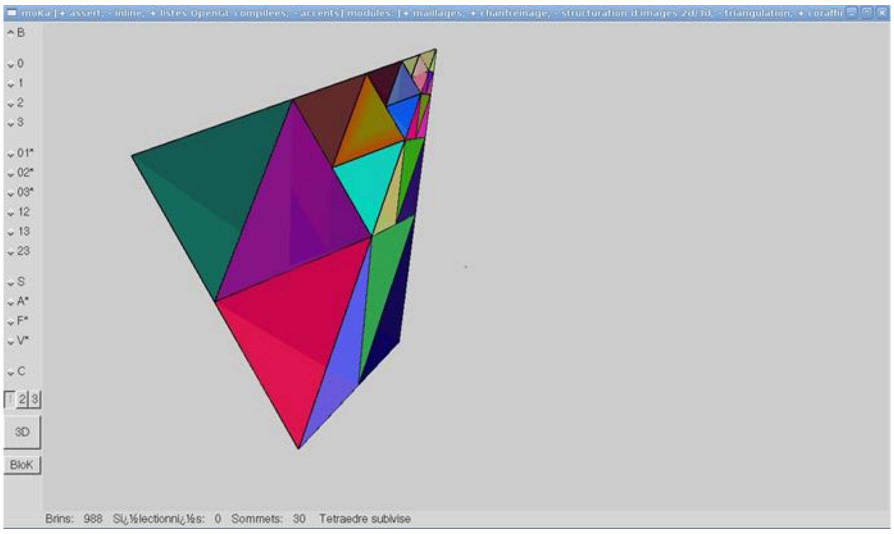
Figure 20: new step 2 tetrahedron subdivision illustrating neighborhood management between tetrahedra.


moKa [+ assert, - inline, + listes OpenGL compilees, - accents] modules: [+ maillages, + chanfreinage, - structuration d images 2d/3d, - triangulation, + coraffir
B
0
1
2
3
01*
02*
03*
12
13
23
S
A*
F*
V*
C
1 2 3
3D
BloK

Brins: 988 Si¿½lectionni¿½s: 0 Sommets: 30 Tetraedre subivise

Figure 21: after three subdivision steps.

The visualization of Figure 22 shows the consistency of our model.



```
Brins:  988  Sï¿½lectionnï¿½s:  0  Sommets:  30  Tetraedre subdivise
```

Figure 22: compact view showing the preservation of the integrity of the tetrahedron after several subdivisions.

6. Conclusion

We have proposed in this paper, a topological model with adaptive resolution for surgical simulation. Our model makes it possible to simulate the operation of recursive cutting of the parts of a meshed organ into triangles or tetrahedra to form triangular or tetrahedral meshes and according to the desired area. This operation, applied to such a mesh, must be robust and ensure that the topological consistency of the model is preserved. Our approach is based on G-maps, which allow to detect topological anomalies in real time. It would be interesting to effectively couple our topological model to mechanical objects.

7. Conflict of Interest

There is no conflict of interest as far as we are concerned.

8. Funding statement

None.

Acknowledgments

Author would like to thank the Xlim-Sic Laboratory.

References

[1] Traite IC2 Information – Commande- Communication. Informatique graphique, modélisation géométrique et animation, 2007.

[2] Cotin. “Modèles anatomiques déformables en temps-réel." Thèse de Doctorat, INRIA Sophia Antipolis-Université de Nice, Sophia Antipolis, 1997.

[3] Bielser, Maiwald et Gross. “Interactive cuts through 3-dimentional soft tissue." In: EUROGRAPHICS’99, pp. C31 – C38. Milan, Italie, septembre 1999.

[4] Francois BOUX DE CASSON. “Simulation dynamique de corps biologiques et changements de topologie interactifs.” Thèse de Doctorat, Université de SAVOIE, 2000.

[5] Van Nguyen, S., Tran, H. M., & Maleszka, M. “Geometric modeling: background for processing the 3d objects." Applied Intelligence, 51, 6182-6201, 2021

[6] M. Mäntylä. “An introduction to Solid Modeling." Computer Science Press, 1988.

[7] P. Lienhardt. “N-dimensional generalized combinatorial maps and cellular QuasiManifolds.” Research Report R 93-04, Université Louis Pasteur, département d’informatique, Strasbourg, France, mars 1993.

[8] Vidil, F., Damiand, G., Dexet-Guiard, M., Guiard, N., Ledoux, F., Fousse, A., ... & Bertrand, Y. “Moka", 2003.

[9] REFIG, “Extension multirésolution des cartes combinatoires : Application à la représentation des cartes de subdivision multirésolution”, 2007.

[10] Sakai, S., Nakamura, M., Anemura, N., Onishi, M., Iizawa, I., Nakata, J., . . . Tamotsu, K. “Sierpinski’s forest: New technology of cool roof with fractal shapes. Energy andBuildings", 2012.