The Visual Computer

Hierarchical generalized triangle strips

Luiz Velho[1],
Luiz Henrique de Figueiredo[2],
Jonas Gomes[1]

[1] IMPA – Instituto de Matemática Pura e Aplicada,
Estrada Dona Castorina, 110,
22460-320 Rio de Janeiro, RJ, Brazil
[2] LNCC – Laboratório Nacional de Computação
Científica, Rua Lauro Müller 455,
22290-160 Rio de Janeiro, RJ, Brazil
E-mail: {lvelho/jonas}@visgraf.impa.br,lhf@lncc.br

This paper introduces a new refinement method for computing the triangle sequences of a mesh. We apply the method to construct a single generalized triangle strip that completely covers a parametric or implicit surface. A remarkable feature of this application is that our method generates the triangulation and the triangle strip simultaneously, using a mesh refinement scheme. As a consequence, we are able to produce a hierarchy of triangle strips defined at each refinement level. This data structure has many applications in geometry compression and rendering.

Key words: Geometric modeling – Mesh representation – Sequential triangulations – Hamiltonian paths – Triangle strips – Accelerated rendering – Geometry compression

Correspondence to: L. Velho

1 Introduction

Triangle meshes, or *triangulations*, are one of the most widely used representations for geometric models. A triangulation is a 2D simplicial complex, a simple structure with nice combinatorial properties. Moreover, surfaces of arbitrary topology can be tessellated into a mesh of triangular patches. Such triangles can be rendered very efficiently in both software and hardware. For this reason, they are the basic geometric primitive of the graphics pipeline.

One of the main disadvantages of triangle meshes is that, in general, they do not provide a compact surface representation because a large number of triangles is required to faithfully describe the geometry of a complex surface.

This problem motivated the search for encoding schemes that could be used to represent triangle meshes in a more compact and efficient way. One such scheme is the *triangle strip* (and its generalizations). The mesh encoding using triangle strips exploits the spatial coherence of the simplicial complex structure. It enumerates mesh elements in a sequence of adjacent triangles to avoid repeating the vertex coordinates of shared edges.

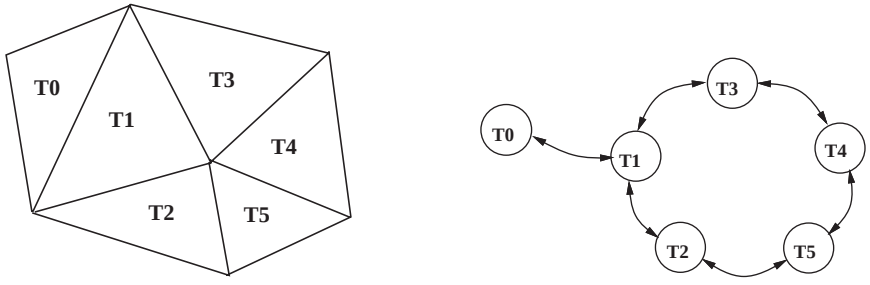
In the traditional setting, the triangle strip encoding leads to the problem of converting a given triangle mesh into the minimal set of triangle strips covering the mesh. This problem is (NP) complete [6] and, thus, existing algorithms must rely on heuristics to find suboptimal solutions.

Triangle meshes are often used to approximate smooth surfaces or to interpolate data points. In such cases, the mesh is generated, respectively, from a surface description (in parametric or implicit form) or from sparse samples. Considering this observation, we noticed that triangle sequences can be constructed during the process of generating the mesh. This reduces the complexity of the problem and leads to a better solution.

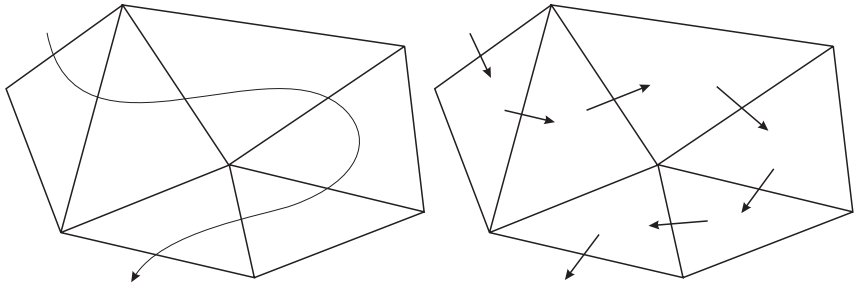
In this paper we describe a method for constructing a hierarchy of generalized triangle strips. The method can be integrated into the mesh creation process. In fact, with a small computational effort, we are able to build triangle sequences that have many good properties.

The remainder of the paper is organized as follows. Section 2 gives definitions and some background of triangle sequences, Sect. 3 discusses previous work, Sect. 4 presents the basic method for gener-

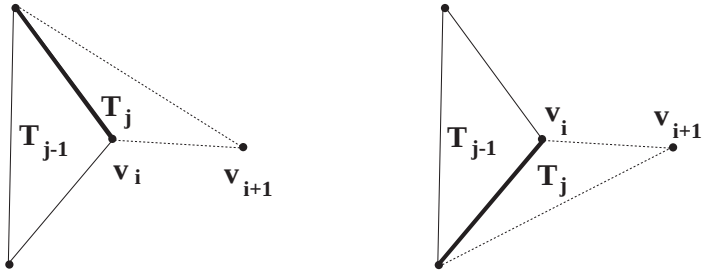
The Visual Computer (1999) 15:21–35



1



2a 2b



3

Fig. 1. Triangulation and its dual graph

Fig. 2a, b. Geometric representations of a triangle sequence

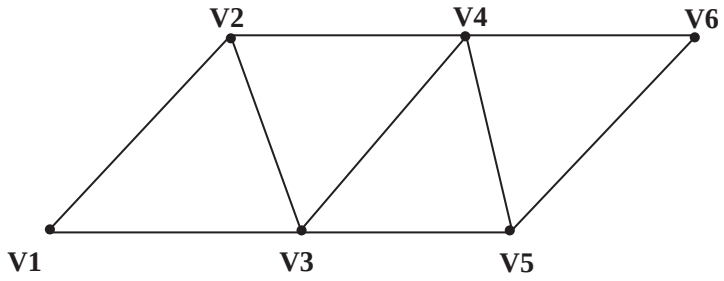
Fig. 3. Two options for continuing a triangle sequence

ating hierarchical triangle sequences using refinement, Sect. 5 analyzes uniformly refinable sequential triangulations, Sect. 6 investigates the case of adaptive refinement, Sect. 7 shows an application of the method to obtain triangular strips that completely cover implicit and parametric surfaces, and Sect. 8 concludes with final remarks and directions for future work.

2 Definitions and background

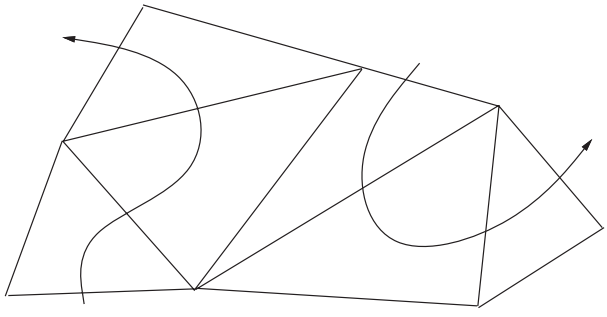
In order to investigate the various ways of representing triangle meshes, we need to study their topological structures, in terms of connectivity and adjacency relations between the elements of the meshes. For that purpose, it is convenient to look at the *dual graph* of the mesh, which explicitly gives the adjacency relations between the triangles of the mesh. In the dual graph, nodes correspond to triangles, and two nodes are connected when their associated triangles have a common edge. Figure 1 shows a triangulation and its dual graph.

A *generalized sequential triangulation*, or *Hamiltonian triangulation*, is a triangulation for which there is an ordering $T_1, \ldots, T_N$ of all its triangles such that two consecutive triangles T_j and T_{j+1} share an edge. Such an ordering exists if and only if the dual graph of the triangulation contains a Hamiltonian path. (A path in a graph is said to be Hamiltonian when it visits all nodes in the graph exactly once.)

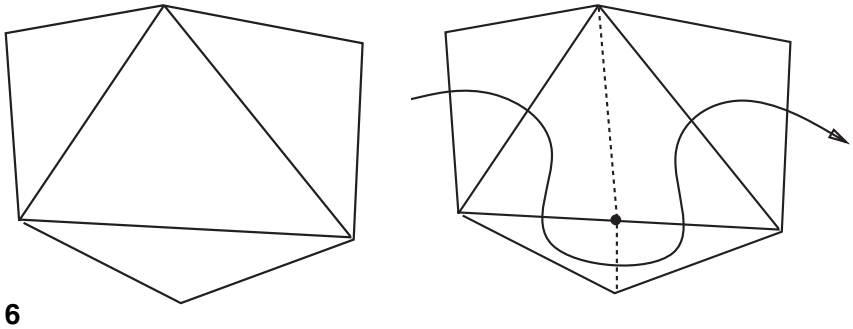


$(V_1,V_2,V_3,V_4,V_5,V_6)$

4



5



6

Fig. 4. A sequential triangulation and its triangle strip encoding

Fig. 5. Partial ordering with two triangle sequences

Fig. 6. Hamiltonian triangulation obtained by insertion of a Steiner point

Notice that each triangle in a Hamiltonian triangulation has an entry edge and an exit edge with respect to the Hamiltonian ordering. The knowledge of these edges completely characterizes the sequence. Geometrically, a generalized triangle sequence can be represented by drawing an oriented path on the mesh domain that visits each triangle crossing its entry and exit edges (Fig. 2a). This path is called a *sequential path*. Another representation indicates the entry and exit edges with arrows (Fig. 2b).

In a Hamiltonian triangulation, suppose that T_{j-1} is a triangle with a vertex v_i that is shared with the next triangle T_j (Fig. 3). Then, triangle T_j is completely determined by specifying:

- A new vertex v_{i+1} (i.e., a vertex distinct from the vertices of triangle T_{j-1})
- The edge shared by T_{j-1} and T_j, which can be the edge to the left (counterclockwise order) or to the right (clockwise order) of the current vertex v_i

It follows that a generalized sequential triangulation with N triangles can be encoded by a sequence of N+2 vertices, defining the geometric information, and a sequence of N bits, defining the connectivity information (i.e., left or right edge turn). This encoding is called a *generalized triangle strip*.

A *sequential triangulation* is a particular kind of generalized sequential triangulation in which the shared edges follow an alternating left/right turn order. Because of this implied restriction, the connectivity does not need to be encoded explicitly, and the representation is given only by the sequence of N+2 vertex coordinates or ids. This encoding is called a triangle strip. Figure 4 shows an example of a sequential triangulation and its representation as a triangle strip.

The triangle strip, as well as the generalized triangle strip, are standard representations of triangle meshes and are supported by most graphics systems, including the OpenGL graphics library [3, 8].

In this paper we shall use the term *triangle sequence* to designate a sequential triangulation, as well as its generalizations.

A triangle sequence defines a total order relation on the set of triangles of a mesh. It is always possible to partition a triangle mesh $\mathcal{T}$ into a collection of subtriangulations $\mathcal{T}_1, \ldots \mathcal{T}_M$, such that each $\mathcal{T}_k$ is a triangle sequence. (A trivial partitioning can be obtained by using one triangle for each sequence $\mathcal{T}_k$.) The partition defines a partial order on the triangulation $\mathcal{T}$. Triangles in the same subtriangulation are related according to their order in the sequence; triangles in different subtriangulations are not related. Figure 5 shows a partial order on a mesh consisting of two triangle sequences.

A natural problem is how to obtain the optimal partition of a mesh into triangle sequences $\mathcal{T}_1, \ldots \mathcal{T}_M$; that is, a partition that minimizes M. Clearly, minimizing M is related to maximizing the length of each triangle sequence $\mathcal{T}_k$. Depending on the application, it may also be desirable to optimize the sequences of left/right edge turns within a sequence.

An optimal partition is the most compact representation of a triangle mesh among all possible sets of triangle sequence encodings. Ideally, we should have a total order among all of the triangles in a mesh; that is, a single triangle strip covering the mesh completely. However, this is not possible in general because the dual graph is not always Hamiltonian. For example, the triangle mesh of Fig. 5 is not Hamiltonian.

However, the problem of finding the best collection of triangle sequences of a mesh is NP complete [6]. Nonetheless, by adding new vertices (called Steiner points), it is always possible to refine a triangle mesh into a Hamiltonian triangulation [2]. Figure 6 shows an example of the insertion of a Steiner point to produce a Hamiltonian triangulation.

Thus, it seems reasonable that triangulation methods based on the insertion of vertices, such as mesh refinement schemes, could be used to generate Hamiltonian triangulations. This paper focus on this problem. In the next section, we review some of the strategies that have been used to compute good triangle sequences. In Sect. 4 we present a new method that uses refinement to construct Hamiltonian triangulations.

3 Previous work and applications

Previous work related to triangle strips falls into three major categories: algorithms to generate triangle strips for accelerated rendering, algorithms to compute sequential triangulations for geometry compression, and theoretical investigation of paths on triangle meshes.

Triangle strips are important for accelerated rendering because they can significantly increase the throughput of the visualization pipeline. First, the data rate is increased, since only $N+2$ vertex coordinates have to be sent to the graphics engine for a sequence of N triangles. Second, viewing operations, such as matrix transforms and clipping, need to be applied only once to the elements of the data stream, further increasing the rendering performance.

Akeley et al. [1] develop a program to convert triangle meshes into strips using a greedy algorithm. They build a sequence by always choosing the next triangle as the one adjacent to the least number of neighbors. Speckmann and Snoeyink [10] build triangle strips based on the dual graph of a triangulation. Their algorithm computes a minimum spanning tree of the adjacency graph and segments it with heuristics that tend to produce long strips. Evans et al. [7] use a technique called patchfication that identifies rectangular regions of a mesh consisting of quadrilaterals. Such regions can triv-

ially be encoded as triangle strips. This last strategy is similar to the one employed in our method, in the sense that it builds the triangle strip while creating a triangulation.

The properties of sequential triangulations can be exploited in various ways for the compression of geometric models. On the one hand, the sequential structure reduces the connectivity information to one bit, or even eliminates its explicit encoding. On the other hand, the locality of reference inherent in a triangle sequence makes it possible to encode geometry with fewer bits using prediction schemes.

Deering [4] introduces the concept of geometry compression based on generalized triangle sequences. His algorithm employs lossy compression for the quantization of coordinate values, as well as a vertex cache to take further advantage of spatial coherence. Taubin and Rossignac [11] construct a mesh representation using two interlocked trees: a spanning tree of triangles for connectivity and a spanning tree of vertices. The geometry information is compressed by variable-length lossy encoding.

The generation of paths on triangulations has been studied in computational geometry. Dillencourt [5] investigates the complexity of finding Hamiltonian paths on the edges of Delaunay triangulations. Arkin et al. [2] consider several issues related to paths on the dual graph of general triangle meshes. In particular, they show that the problem of finding optimal triangle sequences is NP hard.

4 Triangle sequences from mesh refinement

In this section we give an overview of our method for constructing generalized triangle strips with mesh refinement.

A *refinable triangle sequence* is a triangle sequence whose total order can be preserved when its elements are subdivided. Note that this property depends exclusively on the subdivision scheme adopted.

A triangle subdivision scheme is called a *sequential refinement* if it is based on a subdivision template that:

- Decomposes each triangle into a generalized sequential triangulation
- Leads to a refined subsequence within each triangle that is compatible with the ordering of the initial triangulation.

More precisely, if

$$T_1, \ldots, T_{j-1}, \quad T_j, \quad T_{j+1}, \ldots, T_M$$

is the initial triangle sequence, then each triangle T_j has a refinement $T_{j1}, \ldots, T_{jN_j}$ that is itself a triangle sequence, and is such that the triangle mesh

$$T_1, \ldots, T_{j-1}, \quad \underbrace{T_{j1}, \ldots, T_{jN_j}}_{T_j}, \quad T_{j+1}, \ldots, T_M,$$

obtained by replacing triangle T_j by its refinement, is still a triangle sequence.

The sequential refinement we describe in this paper has two main parts:

1. *Initialization.* Generate a base mesh and an initial triangle sequence for it.
2. *Refinement.* Refine the base mesh, recursively propagating the triangle sequence from coarse to finer levels.

The construction of the initial triangle sequence can (and should) take advantage of properties of the base mesh. If the mesh has a regular structure, then there is a natural order of its elements. Figure 7 shows a simple regular base mesh with a triangle sequence following a serpentine path.

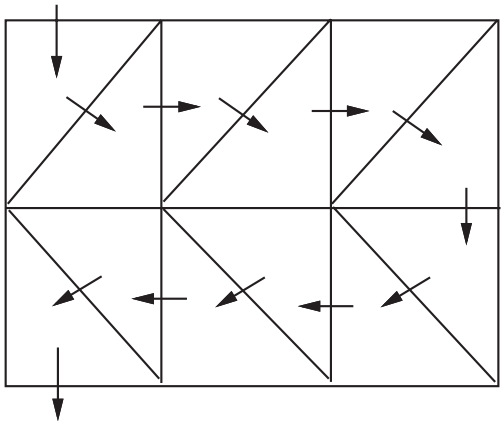
Even when no structural properties of the base mesh can be exploited, just the fact that the base mesh is coarse by construction makes it feasible to compute the initial triangle sequence with conventional algorithms, such as the ones described in Sect. 3.

The second stage of the method takes an ordered base mesh and refines the mesh, maintaining the order as it moves from coarse to finer levels.

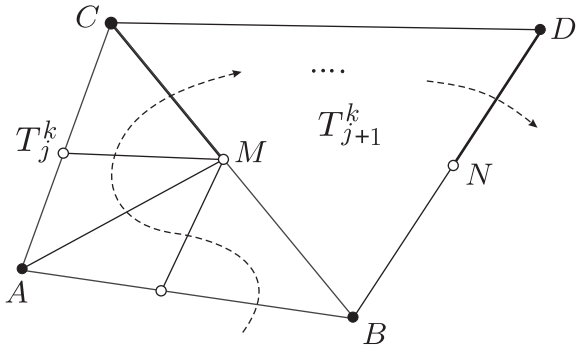
We now give a description of the refinement and prove its correctness. We start from an initial triangle sequence $\mathcal{T}^k$, with triangles ordered as: $T^k_1, \ldots, T^k_{j-1}, T^k_j, T^k_{j+1}, \ldots, T^k_{M_k}$. We need to subdivide each triangle T^k_j into a triangle sequence $\mathcal{T}^{k+1}_j = T^{k+1}_{j1}, \ldots, T^{k+1}_{jN_k}$, such that the refined triangle mesh

$$\mathcal{T}^{k+1}_1, \mathcal{T}^{k+1}_2, \ldots, \mathcal{T}^{k+1}_{M_k}$$

is, in this order, a triangle sequence.



7



8

Fig. 7. Base mesh and serpentine triangle strip

Fig. 8. Propagating the triangle sequence during refinement

Suppose, by induction, that we have accomplished the task up to the *j*th triangle. We have thus constructed the refined triangle sequence

$$\mathcal{T}_1^{k+1}, \mathcal{T}_2^{k+1}, \ldots, \mathcal{T}_j^{k+1}.$$

To complete the induction, we must devise a method to refine the triangle T_{j+1}^k into a triangle sequence $\mathcal{T}_{j+1}^{k+1}$, such that

$$\mathcal{T}_1^{k+1}, \mathcal{T}_2^{k+1}, \ldots, \mathcal{T}_j^{k+1}, \mathcal{T}_{j+1}^{k+1}$$

is also a triangle sequence.
Note that the triangle T_{j+1}^k has an entry edge and an exit edge already determined by the original triangle sequence $\mathcal{T}_k$. Furthermore the entry edge of T_{j+1}^k is the exit edge of T_j^k (edge *CB* in Fig. 8). Therefore, by the induction hypothesis, this edge has possibly been subdivided in the refinement process of the triangle T_j^k, and one of the subedges of the refinement is the exit edge of the last triangle of the sequence $\mathcal{T}_j^{k+1}$ (subedge *MC* in Fig. 8). The refinement of triangle T_{j+1}^k subdivides its exit edge, and one of the subedges must be chosen for the exit edge of the last triangle in the refinement sequence $\mathcal{T}_{j+1}^{k+1}$ of the triangle T_{j+1}^k (subedge *ND* in Fig. 8).
The induction step given shows that the accomplishment of our goal is completely localized: we must devise a template for triangle refinement that makes the induction step possible. The template must be able to provide a sequence path from the entry subedge to one of the exit subedges. We have two strategies for the template construction.

Uniform template. We construct a single template that its used to subdivide every triangle, creating a uniform sequential refinement of the mesh.

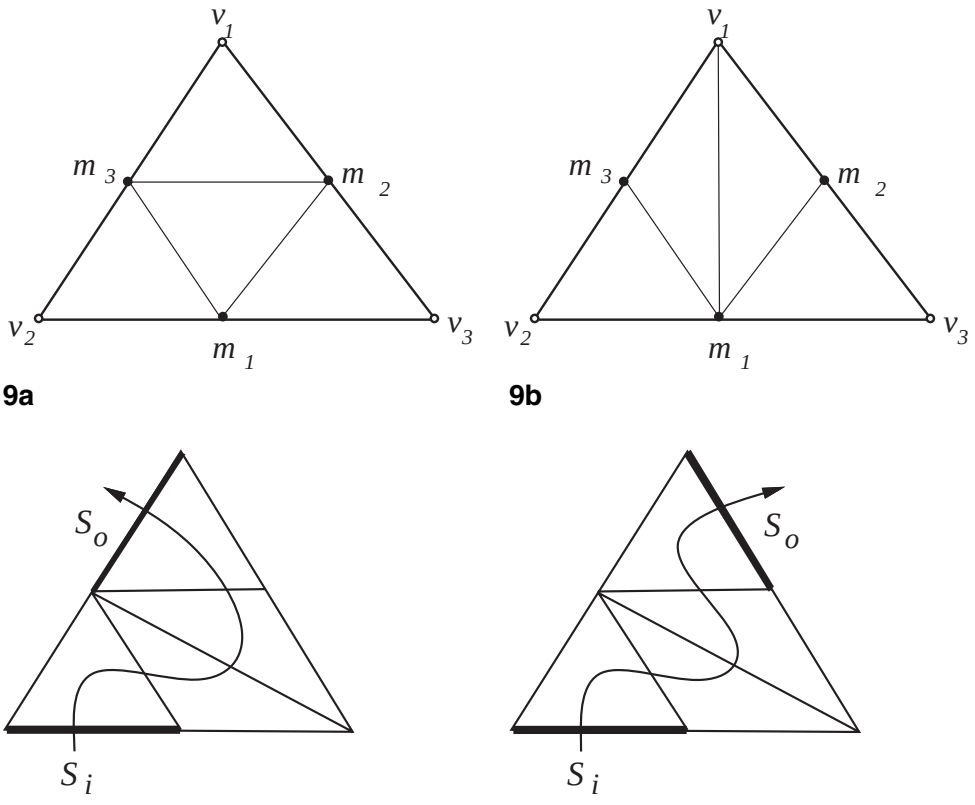


Fig. 9a, b. Uniform decomposition of a triangle: **a** isotropic template; **b** anisotropic template

Fig. 10. Realization of the template of Fig. 9b compatible with the bottom left and bottom right orders

Adaptive templates. We use different templates in order to construct an adaptive sequential refinement.

The core of the algorithm is the construction of the triangle refinement template. In the next two sections, we discuss how to determine suitable subdivision templates for both uniform and adaptive mesh refinement.

5 Uniform sequential refinement

Uniform mesh refinement schemes recursively subdivide all triangles of a mesh, using the same template, until some resolution is obtained.

In the case of triangle meshes, the most common refinement scheme splits all edges (usually at the edge midpoint) into two parts and applies a subdivision template to decompose each triangle into subtriangles. In this case, there are two possible subdivision templates for the uniform decomposition of a triangle. Both subdivide a triangle into four subtriangles. These templates are shown in Fig. 9.

Template 9a induces an isotropic decomposition. It connects each new vertex m_i, at the midpoint of an edge, to vertex $m_{(i+1) \bmod 3}$, at the midpoint of the subsequent edge.

Template 9b induces an anisotropic decomposition. It selects the midpoint m_i of one edge, and connects it to the opposite vertex of the triangle

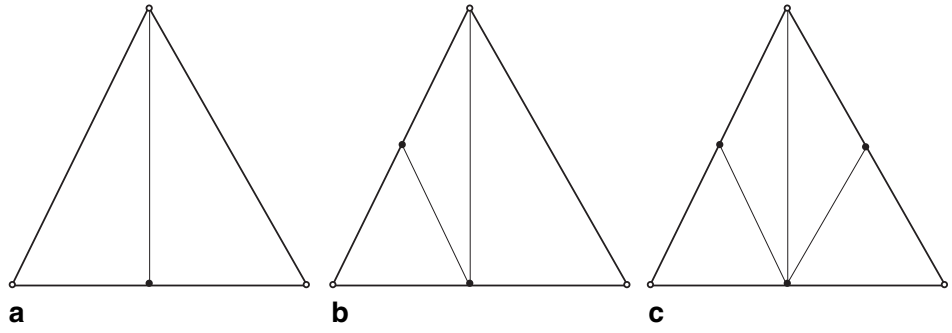


Fig. 11a–c. Templates for nonuniform subdivision

v_i, as well as to the midpoints of the two adjacent edges $m_{|(i-1) \bmod 3|}$ and $m_{|(i+1) \bmod 3|}$. This decomposition scheme can have three realizations, each corresponding to the selection of one of three edge midpoints. Note that the use of one of these particular configurations implies choosing a preferred direction to split the triangle. We have exploited this extra degree of freedom to conform the mesh geometry to a surface (see Sect. 7).

Simple inspection reveals that template 9a does not produce a generalized sequential triangulation, whereas template 9b does. Therefore, only template 9b is suitable for defining a sequential refinement scheme. This result is summarized in the following.

Proposition 1. *A triangle sequence is uniformly refinable if all its elements are subdivided using template 9b in a orientation compatible with the sequence order.*

Proof. From the discussions in the previous section, we need to determine an exit subedge and construct a sequence path from the entry subedge to the exit subedge. There are two possible choices for the exit subedge corresponding to the bottom left and the bottom right. Figure 10 shows a template subdivision and the sequence path for each case. □

We remark that the solution shown is not unique. In fact, template 9b gives two possible orientations for the case where the exit edge is not adjacent to the entry subedge. One alternative is shown in the right side of Fig. 10; the other alternative would be a configuration connecting the midpoint of the entry edge to the opposite vertex.

In the next section, we construct the templates for adaptive sequential refinement.

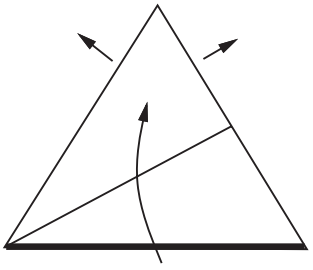
6 Adaptive sequential refinement

Adaptive refinement schemes decompose the mesh in a nonuniform manner.

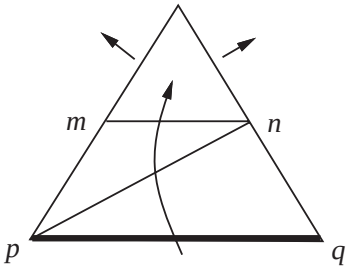
In nonuniform subdivision, triangle edges are also split into two parts (usually at the midpoint), but unlike uniform schemes, not all three edges of a triangle are subdivided. Mesh elements are subdivided to different levels based on some local adaptation criteria. The recursive subdivision of a triangle stops when all three edges satisfy the adaptation criteria.

A consequence of the adaptive nature of nonuniform refinement is that the subdivision templates are more complex. They must take into account the cases of one, two, and three edges splitting. Therefore, the templates consist of all simplicial decompositions generated by internal edges connecting edge midpoints and/or triangle vertices. Figure 11 shows the subdivision templates corresponding to (a) a one-edge split, (b) a two-edge split, and (c) a three-edge split. Note that the template of Fig. 11c is the same one used for uniform subdivision. Obviously, the realization of these templates includes all permutations of different edges splitting, as well as all configurations for connecting them.

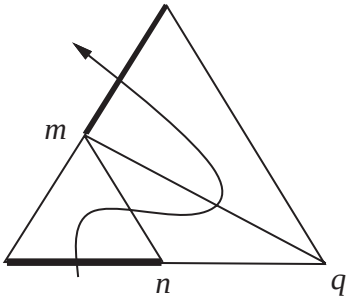
As for uniform refinement, in order to prove that we can construct a refinable triangle sequence, it is sufficient to show that the templates Figs. 11a–c produce triangle sequences that are compatible with the parent triangle sequence order.



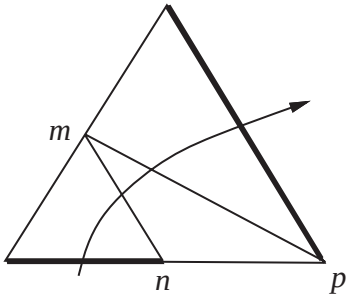
12a



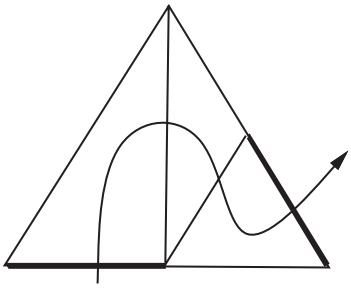
12b



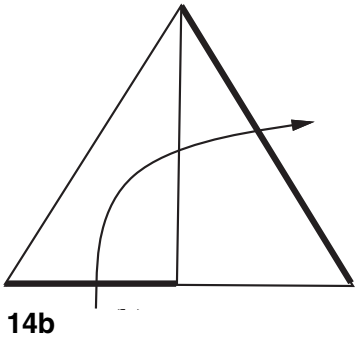
13a



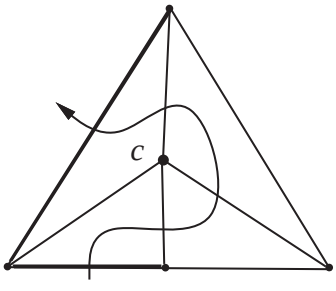
13b



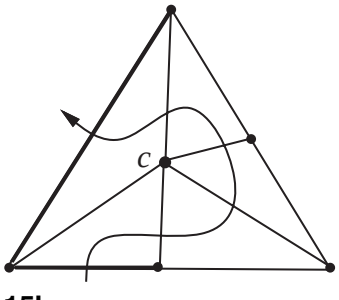
14a



14b



15a



15b

Fig. 12a, b. Subdivision when entry edge does not split

Fig. 13a, b. Subdivision when adjacent edge splits

Fig. 14a, b. Subdivision when adjacent edge does not split and exit edge is not the adjacent edge

Fig. 15a, b. New templates for subdivision when adjacent edge does not split and exit edge is the adjacent edge

Since the cases involving the template of Fig. 11c have already been analyzed in Sect. 5, there are three cases remaining to be considered.

1. *Entry edge does not split.* This is the simplest case. The triangle is subdivided by a main internal edge, transversal to the sequence path. The exit edge could be either to the left or to the right. This configuration is illustrated in Fig. 12. Note that there are two possible realizations for the configuration shown in Fig. 12b, i.e., a subdivision of the quadrilateral *pmnq* into the triangles *pmn* and *pnq* or *pmq* and *mng*.
2. *Entry edge splits, and the adjacent edge to the entry subedge also splits.* As shown in Fig. 13, in this case, the internal edges *mn* and *mp* produce a refinable triangle sequence that begins at the entry subedge and can exit at either one of the two other edges.
3. *Entry edge splits, and the adjacent edge of the entry subedge does not split.* This is the most difficult case, and we break it down in two subcases depending on the exit edge:
3a. When the exit edge is not adjacent to the entry subedge, the refined triangle sequence can be constructed easily, as shown in Fig. 14.
3b. When the exit edge is adjacent to the entry subedge, it is impossible to produce a triangle sequence with the templates of Fig. 11b, c. To overcome this situation, we create two new templates, by inserting an extra vertex *c* in the interior of the triangle and connecting it to the other vertices. These new templates are illustrated in Fig. 15.

In summary, this case analysis demonstrated that these five templates are sufficient to produce refinable nonuniform triangle sequences. They can be used with any adaptive mesh refinement method that is based on edge subdivision.
In the next section, we apply the adaptive sequential refinement to compute triangle strips for implicit and parametric surfaces.

7 Examples

We have implemented an algorithm that employs the sequential mesh refinement method described in this paper to compute simultaneously both a hierarchical triangulation and a hierarchy of triangle sequences. The algorithm produces results superior to methods that compute the triangle strip separately starting from a precomputed triangulation.
The first stage of the algorithm, as outlined in Sect. 4, creates a base mesh that will be used in the refinement stage. The second stage of the the algorithm recursively refines the base mesh, using the either the uniform or the nonuniform subdivision templates that we introduce in Sect. 5 and 6. The algorithm performs adaptive refinement based both on the error measure of the polygonal approximation and on the aspect ratio of the triangles in the mesh. Details of the implementation of this algorithm can be found in [12].
We now present some examples of triangulations and triangle sequences computed by the algorithm.

Unit square. Figure 16 shows the first three levels of a hierarchy of triangle sequences generated by uniform mesh refinement of the square $[0, 1]\times[0, 1]$ of the plane.
The left part of the figure displays the triangle meshes and the right part displays the corresponding triangle sequences, shown as a polygonal path connecting the midpoints of entry and exit edges. The base mesh was a decomposition of the square into two triangles along one diagonal. Note that the refinement process produces a triangle sequence that follows the path of a space-filling curve (a Sierpinski-like curve).

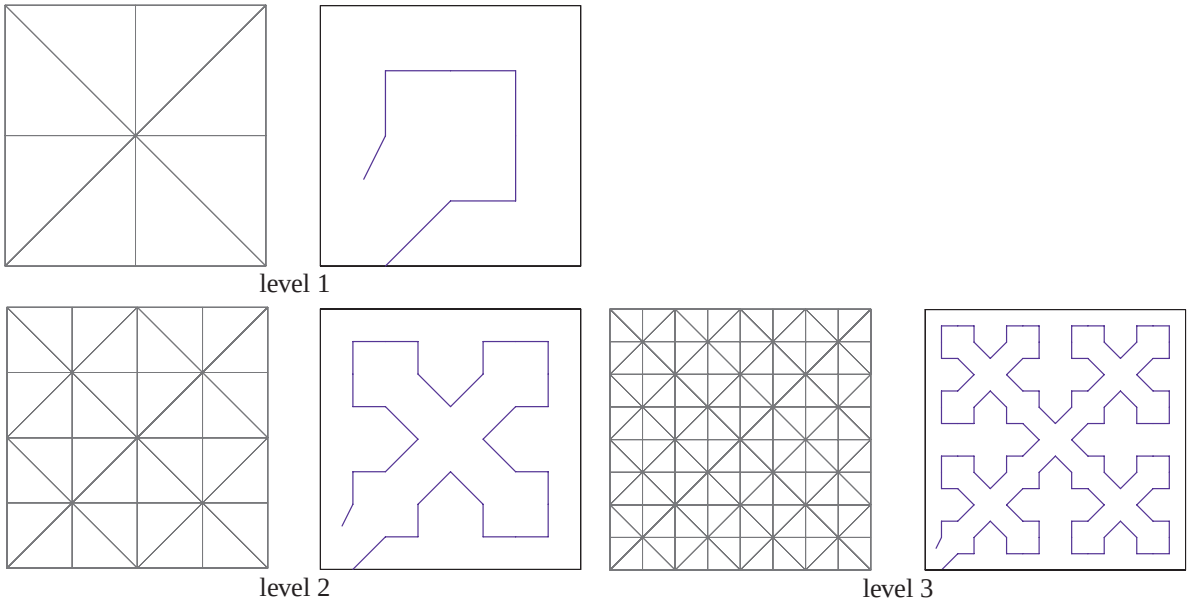
Height field. Figure 17 shows a height field surface given by the regular sampling of a terrain elevation model. The base mesh was a 2×2 rectangular grid, with the initial triangle sequence following a serpentine path. The mesh was refined down to four levels by uniform subdivision.

Sphere. Figure 18 shows the adaptive tessellation of a sphere without the polar caps, which is described parametrically by:

$$x = \cos u \cos v, \quad y = \sin u \cos v, \quad z = \sin v;$$

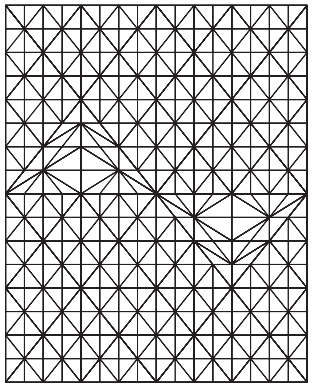
$$u \in [0, 2\pi], \quad v \in [\pi/2 + 0.2, 3\pi/2 - 0.2].$$

Figure 18a, b displays the decomposition of the parametric domain and the triangulation of the surface in 3-space. Figure 18c, d displays the paths corresponding to the triangle sequence in parameter space and on the surface. Notice that the triangulation forms a restricted structure, while the triangle sequence follows the path of a space-filling curve that is adapted to the surface.

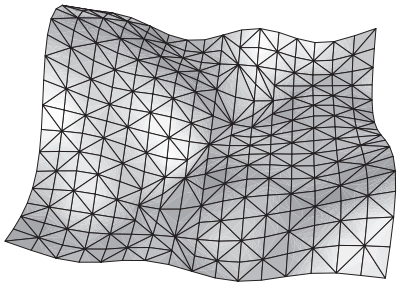


16

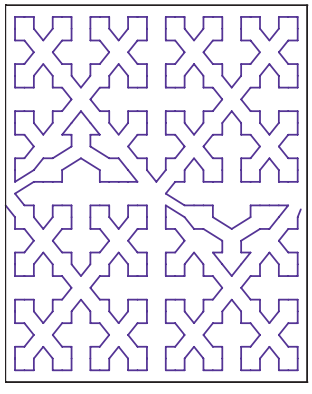
Fig. 16. Hierarchical mesh and triangle sequences



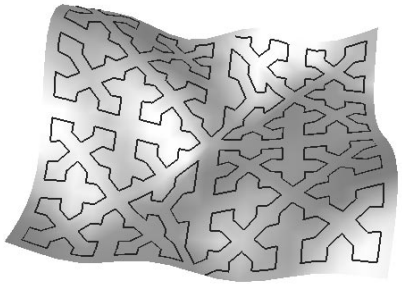
17a



17b

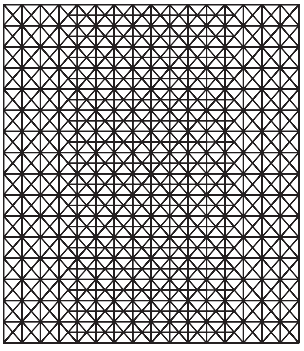


17c

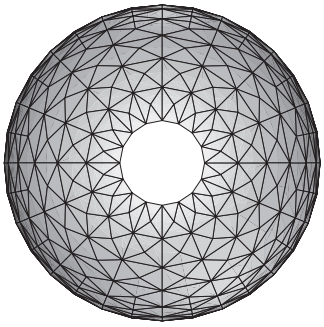


17d

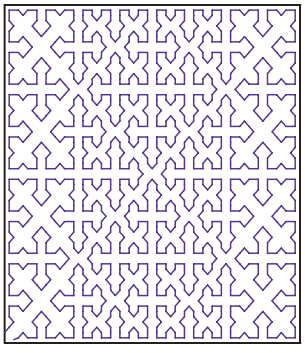
Fig. 17a–d. Uniform tessellation of terrain model



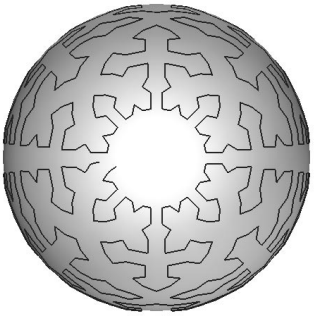
18a



18b

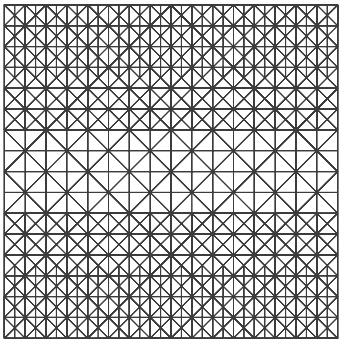


18c

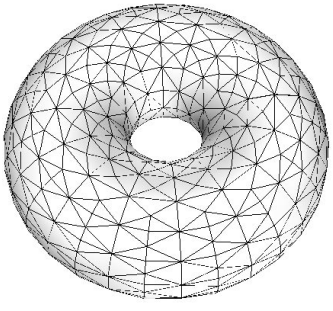


18d

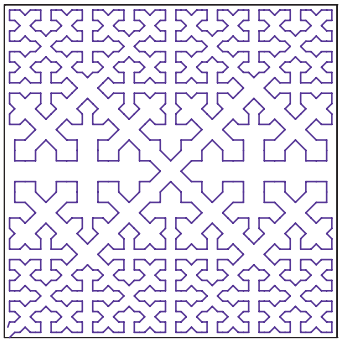
Fig. 18a–d. Sphere



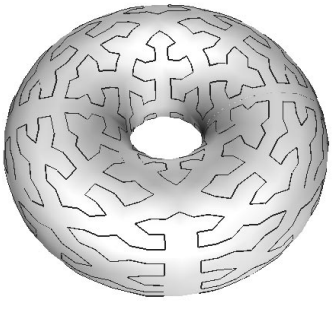
19a



19b



19c



19d

Fig. 19a–d. Torus

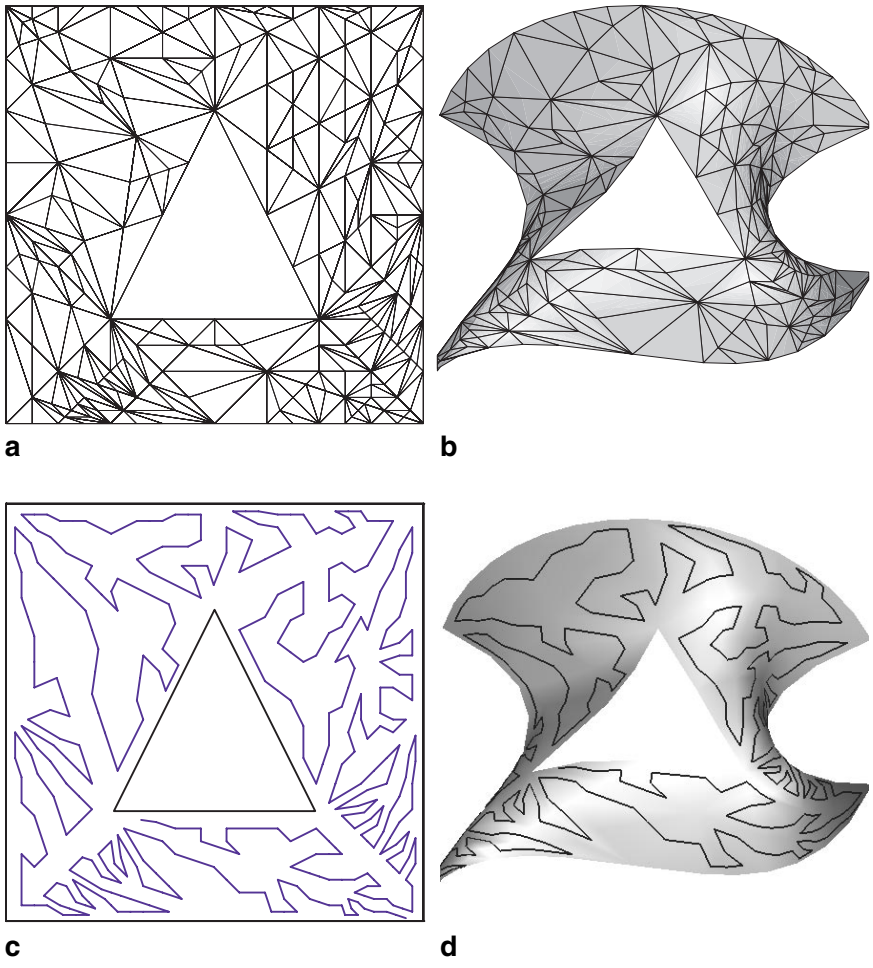


Fig. 20a–d. Bézier surface

Parametric torus. Figure 19 shows the adaptive tessellation of a torus, which is described parametrically by:

$$x = \cos u(r + \cos v), \quad y = \sin u(r + \cos v),$$

$$z = \sin v;$$

$$u \in [0, 2\pi], \quad v \in [0, 2\pi], \quad r = 1.6.$$

As in the previous example, the triangle sequence also follows a space-filling path.

Trimmed Bézier patch. Figure 20 shows the polygonization of a trimmed Bézier surface patch with control points

$$\begin{bmatrix} (-1,0,0) & (0,3,0) & (3,3,0) & (4,0,0) \\ (1,0,1) & (1,-3,1) & (3,3,1) & (2,0,1) \\ (0,0,2) & (0,3,2) & (3,3,2) & (3,0,2) \\ (-1,0,4) & (0,3,3) & (3,-3,3) & (4,2,3) \end{bmatrix}.$$

Despite of the complex topology of the parametric domain, we were able to generate a well-adapted mesh, as well as a triangle sequence that covers the entire patch.

8 Conclusions and future work

We have presented a new method for constructing generalized triangle strips from an initial, coarse sequence. The method uses both uniform and adaptive refinement schemes. We have also applied the method to obtain refinable generalized triangle sequences that approximate parametric or implicit surfaces.

The recursive nature of the process actually produces a hierarchy of triangle sequences, one for each level of refinement. This structure can be employed to advantage in the compression of multiresolution models and in accelerated progressive rendering.

As future work, we plan to investigate several issues, including the properties of space-filling curves on manifolds, the potential of locality of reference to increase geometric compression, and applications of triangle sequences to artistic rendering of surfaces.

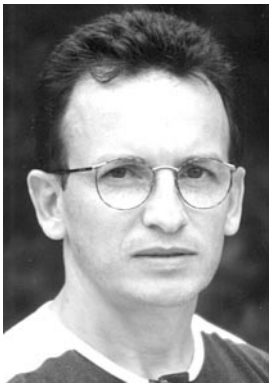
Acknowledgements. The figures in Sect. 7 were generated with Geomview [9]. The authors are partially supported by research grants from the Brazilian Council for Scientific and Technological Development (CNPq) and Rio de Janeiro Research Foundation (FAPERJ).

References

1. Akeley K, Haeberly P, Burns D (1990) tomesh.c: C program on SGI developer's toolbox CD
2. Arkin E, Held M, Mitchel J, Skiena S (1994) Hamiltonian triangulations for fast rendering. The 2nd Annual European Symposium on Algorithms, (Lecture Notes in Computer Science, vol 855). Springer, Berlin Heidelberg New York, pp 36–47
3. Open GL Architecture Review Board (1993) OpenGL Reference Manual. Addison Wesley
4. Deering MF (1995) Geometry compression. In: Cook R (ed) SIGGRAPH'95 Conference Proceedings, Annual Conference Series, pp 13–20
5. Dillencourt M (1992) Finding Hamiltonian cycles in Delaunay triangulations is NP-complete. Proceedings of the 4th Canadian Conference on Computational Geometry, pp 223–228
6. Evans F, Skiena S, Varshney A (1996) Completing sequential triangulations is hard. Technical Report Department of Computer Science, State University of New York, Stony Brook, NY
7. Evans F, Skiena S, Vashney A (1996) Optimizing triangle strips for fast rendering. IEEE Visualization'96
8. Kilgard MS (1997) Realizing OpenGL: two implementations of one architecture. In: Molnar S, Schneider B-O (ed) 1997 SIGGRAPH/Eurographics Workshop on Graphics Hardware, New York City, NY
9. Levy S, Munzner T, Phillips M Geomview. Software written at the Geometry Center, University of Minnesota. Available at http://www.geom.umn.edu/software/
10. Speckmann B, Snoeyink J (1997) Easy triangle strips for TIN terrain models. Canadian Conference on Computational Geometry, 239–244
11. Taubin G, Rossignac J (1998) Geometric compression through topological surgery. ACM Transactions on Graphics, Vol 17, No 2, pp. 84–115, April 1998
12. Velho L, Figueiredo LH de, Gomes J (1998) Hierarchical adaptive polygonization of parametric and implicit surfaces: Instituto de Matemática Pura e Aplicada, Rio de Janeiro, preprint

LUIZ VELHO is an Associate Researcher at the Instituto de Matematica Pura e Aplicada (IMPA) of CNPq, and a member of the Visgraf project. He received a BE in Industrial Design from ESDI/UERJ in 1979, an MS in Computer Graphics from the MIT/Media Lab in 1985, and a PhD in Computer Science in 1994 from the University of Toronto under the Graphics and Vision groups. His experience in computer graphics spans the fields of modeling, rendering, imaging and animation. From 1987 to 1991 he was a Principal Engineer at Globo TV Network in Brazil, where he created special effects and visual simulation systems. In 1994 he was a visiting professor at the Courant Institute of Mathematical Sciences of New York University.

LUIZ HENRIQUE de FIGUEIREDO has a BSc and an MSc in Mathematics from PUC-Rio, the Catholic University of Rio de Janeiro, and he received a PhD in Mathematics in 1992 from IMPA. His current research interests include computational geometry, geometric modeling, and interval methods in computer graphics. He is currently an Associate Researcher at the Laboratório Nacional de Computacão Cientifica (LNCC) and a collaborator of the Visgraf project at IMPA.



JONAS GOMES is a full Professor at the Institute of Pure and Applied Mathematics (IMPA) in Rio de Janeiro, and he is a member of the Brazilian Academy of Sciences. He received a PhD in Mathematics in 1984 from IMPA. From 1984 to 1988 he was the R&D manager of the computer graphics group at Globo TV network. In 1990 he started the Vision and Graphics Lab at IMPA (Visgraf Lab). The Lab is involved with research and development, and also supports a graduate program in computer graphics. He has published several books and research articles in computer graphics. His current research interest include modeling, imaging, animation, and mathematical methods of computer graphics.