Data Compression of Multiresolution Surfaces

Reinhard Klein, Stefan Gumhold

WSI/GRIS University of Tübingen, 72076 Tübingen, Germany

Abstract. In this paper we introduce a new compressed representation for multiresolution models (MRM) of triangulated surfaces of 3D-objects. Associated with the representation we present compression and decompression algorithms. Our representation allows us to extract the surface at variable resolution in time linear in the output size. It applies to MRMs generated by different simplification algorithms like local vertex deletion or edge and triangle collapse. The time required to transmit models over communication lines and the space needed to store the MRMs is significantly reduced.

1 Introduction and previous work

Triangle meshes are one of the most popular representations of surfaces for computer graphics applications. On the one hand, rendering of triangles is widely supported by hardware and, therefore, fast. On the other hand, there is an increasing set of data acquisition techniques which generate triangle meshes as output. However, most of these techniques generate much more triangles than necessary to represent the given object with a small approximation error. Isosurface generation can create 1-10 millions of polygons. A digital map of Germany with a resolution of 40 meters in North-to-South and West-to-East direction results in about 500M points. These huge amounts of data lead to problems with data storage and post-processing programs. Animation and real-time rendering of such data sets is almost impossible even on high performance graphics hardware.

Various techniques were published that aimed to reduce surface complexity in order to speed up rendering time [HH92,MSS94,KT96,SZL92,CVM+96,RKH96], [CCMS97,HDD+93,Hop96,RR96,RB93,Tur92,EDD+95]. Aside from the mesh simplification algorithms recent research also focuses on multiresolution representations of triangle meshes. A complex mesh is replaced by several levels of detail (LODs). There are already a number of multiresolution models (MRMs) which make it possible to extract view-dependently simplified meshes at variable resolution [dBD95,dFP95,CPS97], [KLR+95,Pup96,Hop96,KS96].

The models for terrain surfaces and parameterized free-form surfaces in our previous work [KHK96] allow us to refrain from storing the connectivity [1] of the meshes as well as the hierarchy between the different LODS. This leads to a massive data reduction for the MRM. In the other, more general models, the

[1] The connectivity of a triangle mesh comprehends all adjacency relationships of the mesh.

connectivity and the hierarchy have to be stored explicitly. Since the number of triangles of the multiresolution model is about three times the number of original triangles, it is often impossible to be keep large models in memory. Data reduction of the connectivity and the hierarchy is indispensable for real-time applications.

A first step into this direction were the progressive meshes proposed by Hoppe [Hop96]. Although with this approach high reduction rates are feasible only restricted selective refinement is possible. His recent paper [Hop97] concentrates on this deficiency but at the expense of storage efficiency. Note that the progressive meshes are based on a special mesh simplification technique, the edge collapse operation. The Multi-Triangulation (MT) introduced by Puppo [Pup96] is more general in the sense that it can be combined with different simplification techniques. In [KK97] we generalized the MT proposed by Puppo for surface meshes embedded in 3D, but for large models the storage needed to store the connectivity and the hierarchy is to high. Therefore, in this paper we propose a new approach, which significantly reduces the storage costs of the model.

2 The simplification algorithm and the multiresolution model

2.1 The simplification algorithm

The simplification algorithm simplifies the input triangulation by successively removing vertices [SZL92]. The minimal one-sided Hausdorff distance between the current triangulation and the simplified one determines which vertex is removed next. A priority queue is used to speed up the computation of the next vertex [KLS96]. All triangles incident to a removed vertex are eliminated from the current triangulation and the resulting hole is retriangulated. From the several possible retriangulations an application dependent optimal one is chosen. A closer look on the different retriangulation strategies reveals that the edge collapse operation can be considered as a special retriangulation technique [KK97]. The simplification algorithm stops if no further vertices can be removed from the simplified triangulation without violating an approximation criterion. The criterion used in our reduction algorithm limits the one-sided Hausdorff distance between the input triangulation and the simplified one.

2.2 The Multi-Triangulation

In the following we briefly describe a data structure for the MT as proposed in [Pup96,KK97]. Besides a list of vertices and a list of triangles the data structure contains a list of *fragments* and a coarse and therefore small starting triangulation $\mathcal{T}_0$, e.g. a tetrahedron. The fragments encode localized refinement steps and consist of two small sets of connected triangles, the coarse *floor* and the finer *ceiling* with the same border polygon, see Figure 1. As the ceiling can replace the floor in order to refine the triangulation, the fragments implicitly define the hierarchy between different levels of detail. In the case of vertex removement,

each fragment represents one vertex elimination. The indices of the triangles incident to the removed vertex are listed within the fragment in the list *ceiling* and the indices of the triangles in the retriangulation are stored in the list *floor*. Additionally, the global one-sided Hausdorff distance between the mesh without the removed vertex and the original triangulation is stored in each fragment. Each triangle keeps two pointers to the *upper* and *lower fragment* it belongs to. These interconnect the fragments that overlap and complete the storage of the hierarchy among the fragments.

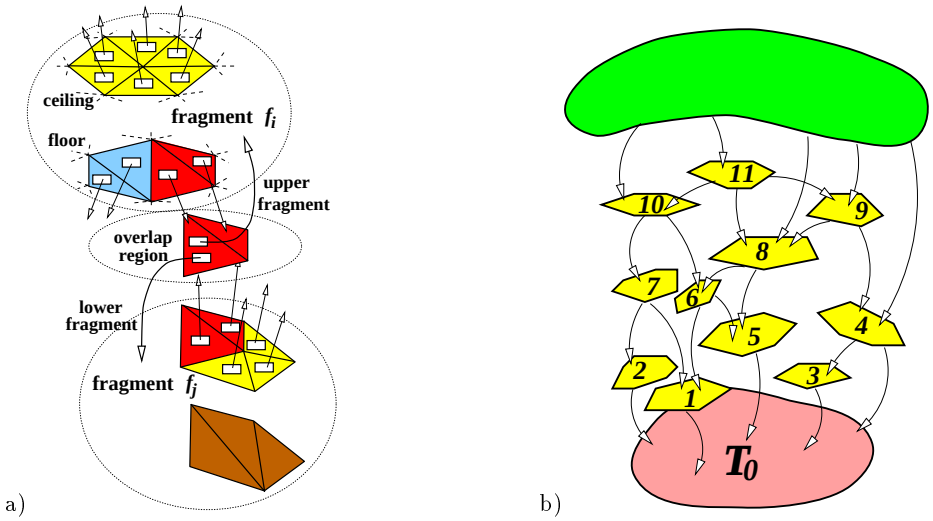

Fig. 1. a) The connectivity of a vertex removement operation is stored in a fragment composed of two lists of triangles – the floor and the ceiling. Each triangle stores its upper and lower fragment, what makes the hierarchy reconstructible. b) The acyclic graph of dependences among the fragments.

3 Using patterns for compression

The main idea of the compression algorithm is based on the following observations:

1. The number of triangles in the fragments is small and bounded by the maximum order of the removed vertices.
2. The fragments with the same number of vertices only differ in the triangulation of the floor.

Therefore, only a few different triangulation types appear in the fragments. For a pentagon all possible triangulations can be obtained from one of its triangulations by rotations. The storage needs for the connectivity of the MT can be

significantly reduced if the connectivity information is stored exclusively for the *equivalence classes* [2] of triangulations. To reconstruct a fragment in the current triangulation beside the index of its equivalence class, two indices of adjacent vertices must be known, see figure 2. From these so called *anchor vertices* together with the connectivity of the equivalence class the complete fragment in the current triangulation can be reconstructed, see figure 2. The first anchor vertex determines the position of the fragment in the current triangulation and the second fixes the orientation. In the following the different equivalence classes of triangulations are called *patterns*. In the more general case, where the con-

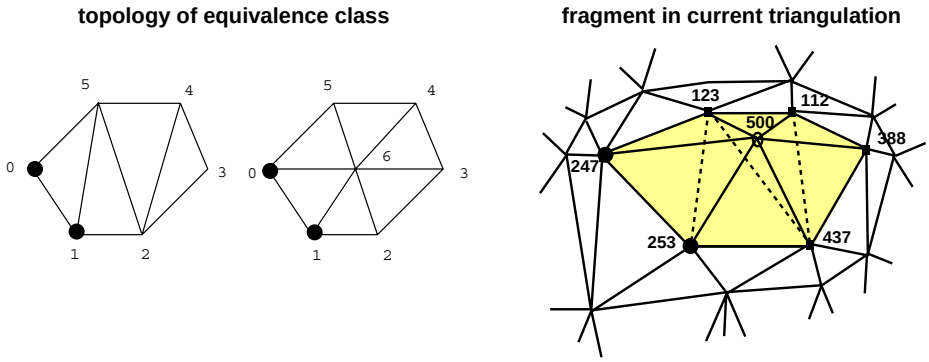

Fig. 2. A fragment is reconstructible from two anchor vertices together with the connectivity of the equivalent class. The anchor vertices 0,1 are equivalent to the vertices 247 and 253 in the current triangulation.

nectivity of the ceiling is not predetermined, the connectivity of the fragment is stored in form of two patterns composing a *rule*.

3.1 Progressive transmission

The model described so far makes progressive transmission possible. For each fragment the index of the corresponding rule, the first anchor vertex and the orientation of the fragment is needed. To store the orientation of the fragment we use the fact that the order in which the vertices are transmitted is fixed. Therefore, among the neighbor vertices of the first anchor vertex exists a vertex with smallest index. Starting from this vertex the neighbors of the first anchor vertex are indexed counterclockwise. If the order of the first anchor vertex is limited to 16, four bits are enough to uniquely define the second anchor vertex and thus the orientation of the fragment. For progressive transmission the storage of each fragment decomposes into one byte for the index of the pattern, four bytes for the index of the first anchor vertex and 4 bits to encode the orientation. This sums up to 44 bits per fragment. In the case of highly reduced models the

[2] Each type of triangulation forms an equivalence class.

number of fragments is the same as the number of vertices. This yields nearly the same compression rate as progressive meshes – also for the more general case of vertex removement. The only overhead compared to the progressive meshes is the storage needed for the rules, which is negligible as each floor triangulation can be encoded in one triangle strip and the ceilings are predetermined.

3.2 Compressing the hierarchy

As the hierarchy in this stage is not explicitly stored the model does not allow us to extract a triangulation at variable resolution in time linear in the output size. In the original MT all fragments contain pointers to their triangles and vice versa all triangles contain pointers to their upper and lower fragments to store the hierarchy. Without further compression techniques the amount of storage required is 160 bytes per vertex [3], [Gum98].

Since in our pattern based approach the triangles are implicitly stored in the equivalence classes, the hierarchy is stored using links between the fragments. Suppose for example that the floor of fragment $\mathbf{f}_i$ joins some triangles with the ceiling of fragment $\mathbf{f}_j$ (compare figure 1). This implies that the ceiling of fragment $\mathbf{f}_i$ cannot be inserted before the ceiling of $\mathbf{f}_j$ is present in the current triangulation. Therefore, $\mathbf{f}_i$ depends upon $\mathbf{f}_j$ and we say $\mathbf{f}_i$ *directly covers* $\mathbf{f}_j$. In the pattern based approach the hierarchy can be represented by all the direct dependences among the fragments. Therefore, we link for each fragment the directly covered fragments in a closed linked list called *loop* (see figure 3). The crucial point is that the links of each loop are stored within the covered fragments themselves. Thus in each fragment $\mathbf{f}$ one downward pointing link and a list of upward pointing links is stored. The downward link begins the loop containing the fragments which are directly covered by $\mathbf{f}$ and the list of upward links carry on the loops of the fragments which directly cover $\mathbf{f}$. Each link of a loop consists of a fragment index and a *loop index*. The fragment index specifies the next covered fragment $\mathbf{f}$ in the loop and the loop index gives the position of the next link of this loop within the list of upward links of fragment $\mathbf{f}$.

This allows us bi-directional navigation among the directly overlapping fragments. The covering fragments can be found by following all upward links to the end of the corresponding loop. As the number of triangles in a fragment is limited, also the lengths of the loops are limited and therefore the covering fragments can be found in constant time.

To analyze the storage needs of our approach we have to count the number of links in all loops. Each link corresponds to a direct overlap of two fragments. Thus we introduce the *number of overlaps* u. The measurements in section 4 show that the relation between u and the number of fragments f is $u \approx \frac{5}{2}f$ in the case of vertex decimation, which can also be founded with a probabilistic argument [Gum98]. In addition to the u links for each fragment the following quantities have to be stored to give full access to the connectivity and the hierarchy:

1. the indices of two anchor vertices
2. the index of the rule representing the fragment's connectivity

[3] We assume that vertex indices and pointers are encoded with 32 bits.

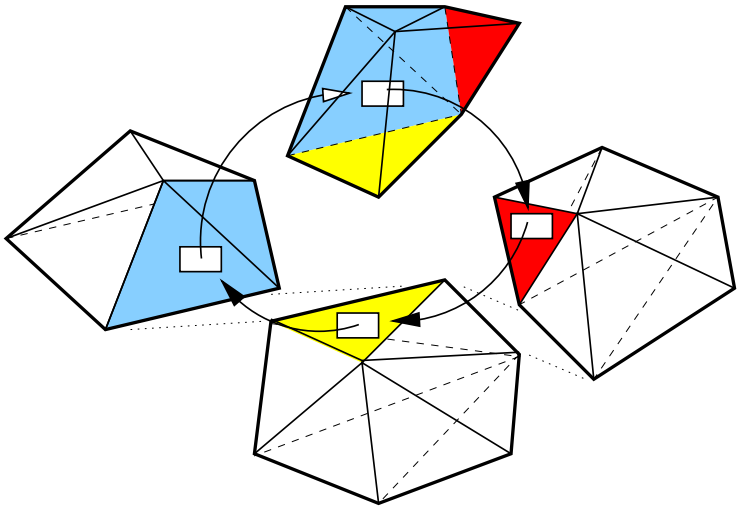

Fig. 3. The floor of the upper fragment covers three other fragments. The four fragments are linked together starting and ending at the covering fragment. Note that the elements of the single linked list are stored in the fragments.

3. the number of loops the fragment is contained in
4. the downward link.

As the extracted triangulation is only restricted to the hierarchy, the neighbors of a vertex are not fixed. This is the reason why the second anchor vertex has to be explicitly stored.

In [KK97] it is proposed to limit the order of the removed vertices to accelerate the simplification algorithm. A maximum order of ten limits the maximal number of rules to 7147 [Gum98]. As the number of triangles in the ceiling of the fragments is the same as the order of the removed vertex, also the maximal number of loops a fragment is contained in is limited by ten. Thus the rule index and the number of loops can together be encoded in sixteen bits.

As each loop points back to the beginning, it contains one element more than the number of overlaps it describes. Therefore, the downward links in the fragments must be counted separately. Each link consists of a fragment index and a loop index. If we assume 32 bits per vertex index, also a fragment index can be encoded in 32 bits and the loop index in 4 bits, which sums up to 36 bits per link.

Altogether, the storage needed for the connectivity and the hierarchy in the pattern based multiresolution model sums up to:

$$\begin{aligned} S_{pbMRM} &\approx f \times (64 + 16 + 36)\text{bits} + u \times 36\text{bits} \\ &\approx f \times \left(116 + \frac{5}{2} \cdot 36\right) \text{ bits} = 206 f \text{bits} \end{aligned}$$

Recalling that the number of fragments is about the number of vertices, 26 bytes per vertex are required for the pattern based multiresolution model. Compared to the MT data structure this corresponds to a reduction factor of six. Again the overhead needed to store the rules is negligible for large models.

3.3 Building a compressed multiresolution model

There are two possibilities to build a compressed multiresolution model. Either the model is extracted from a MT or it is directly build from the simplification algorithm. The first method is slightly simpler as the hierarchy is explicitly stored in the MT. The simplification algorithm successively provides the triangles in the ceiling and the floor of the fragment corresponding to the removed vertex. If for each triangle in the current triangulation temporarily its upper fragment is stored, the covering fragments can be determined from the triangles in the ceiling. The newly produced fragment must be inserted into the loops of the covering fragments.

In both methods the difficult part is to extract the rules from the fragments. Therefore, a comparison algorithm for patterns was developed, which runs in $O(n \times o_{max}^2)$ time, where n is the number of vertices in the pattern and o_{max} the maximal order of its vertices. If the comparison of two patterns is successful, the algorithm produces a map between the pattern vertices. This makes it possible to determine the anchor vertices of fragments whose rule was extracted earlier. As o_{max} is limited to eight [4] and n to ten, the comparison algorithm runs in constant time. Nevertheless to improve performance we defined a Hash-Code on patterns. Major increments are the number of vertices and the vertex orders. To achieve a distinct Hash-Code for patterns and their mirror images the order of the border vertices is exploited. Our Hash-Code is unique for all floor patterns with less than nine vertices [Gum98].

3.4 Incremental selective refinement algorithm

Within a specific application, a criterion must be available to define the required resolution of the multiresolution model. The standard approach is to define a boolean condition c on the triangles of the multiresolution model. If a triangle is invalid, the triangles in the ceiling of its upper fragment are inserted to refine the triangle. In the pattern based approach the triangles are not stored explicitly. Here the condition c is defined on each overlap of two fragments, i.e. on each upward link. This is completely sufficient and if the conditions of the triangles in an overlap can be combined to one condition, the storage space for c is reduced. For example the Hausdorff distance between a triangle and the original mesh can easily be combined by choosing the maximal distance among the triangles in an overlap. Often it is sufficient to define the condition c only on the fragments.

The extracted triangulation additionally must obey to the criterion of "minimal refinement", i.e. it must be the coarsest triangulation that is extractable

[4] Only the floor patterns must be compared. These only contain border vertices, the number of which is restricted by the removed vertex.

from the multiresolution model satisfying c. Our incremental algorithm is based on the storage of the current triangulation, which satisfies c and the "minimal refinement" criterion. If the condition c is changed by the application, the current triangulation is adapted to the new condition c in two steps. First the triangulation is coarsened: top fragments [5] containing exclusively valid upward links are removed from the triangulation, until no such fragments are left over. To remove a fragment from the current triangulation, the vertices of the rule are mapped to the vertices of the current triangulation by using the two anchor vertices and a modified comparison algorithm. Then the triangles in the ceiling of the fragment are replaced by the triangles in the floor.

In the second step the current triangulation is refined until all overlaps contained in the current triangulation satisfy the condition c. Refinement is achieved by inserting a fragment: in the current triangulation the floor triangles of a fragment are replaced with its ceiling triangles, introducing the previously removed vertex. Two problems have to be solved:

1. How to find the fragments to the overlaps contained in the current triangulation ?
2. How to ensure that the floor of a to be inserted fragment is part of the current triangulation ?

To solve the first problem, we enumerate the vertices and their corresponding fragments in the order the vertices are removed by the reduction algorithm. This gives us a one to one correspondence between the removed vertices and the fragments. By the way, because of this trick we don't have to store the indices of the removed vertices in the fragments, as they are given by the fragment index. Each triangle in the current triangulation belongs to an overlap that has to be checked for refinement. From the fragments corresponding to the corner vertices of the triangles inserted at last – thus the one with the smallest index – must contain the triangle in its ceiling, because all triangles around an inserted vertex are newly introduced to the current triangulation.

The second problem can be solved with the recursive algorithm *layFoundation*:

Algorithm layFoundation(f)

for all fragments f_i in loop below of f **do**
    if f_i is not inserted **then**
        layFoundation(f_i)
        insertFragment(f_i)
    endif
endfor

Before insertion a fragment must be *founded*, i.e. all triangles in the floor of the fragment are forced into the current triangulation. This is achieved by following

[5] A fragment is a top fragment of the current triangulation, if all triangles of its ceiling are contained in the current triangulation. Then the fragment lays on top of the hierarchy restricted to the fragments with ceiling triangles in the current triangulation.

the downward loop to all covered fragments and insert them in the same way. As each fragment is visited exactly once by *layFoundation*, the refinement algorithm is linear in the output size.

4 Measurements

In this section we present measurements on several models of different size. The basic characteristics and the storage needs for the pattern based multiresolution model and the Multi-Triangulation are tabulated in table 1. The implicit surface was produced with a marching cube algorithm. The femur- and the jaw-bones where scanned with a CT-scanner and their triangulations were also obtained by a marching cube algorithm. The face was scanned with a 3D-scanner and triangulated. Satellite elevation data forms the landscape model. All multiresolution models were produced by a simplification algorithm based on vertex decimation.

model	n	f	u	S_{pbMRM}	S_{MT}	$S_{geometry}$	$\frac{S_{pbMRM}}{S_{MT}}$
surface	1340	1303	$2.31f$	36kB	178kB	31kB	4.9
femur	5594	5296	$2.40f$	155kB	795kB	131kB	5.3
face	12530	12446	$2.40f$	346kB	1770kB	294kB	5.2
jaw	12349	11990	$2.39f$	335kB	1759kB	289kB	5.3
landscape	29141	28995	$2.46f$	811kB	4261kB	683kB	5.4

Table 1. The table shows for several models the number of vertices n, the number of fragments f produced by vertex removement, the storage needs for connectivity and hierarchy in the cases of the pattern based model and the Multi-Triangulation, the additional storage needs for coordinates and normals and finally the fraction between the storage needs for connectivity and hierarchy.

Table 1 shows that the storage needs for connectivity and hierarchy in the case of the pattern based multiresolution model is in the order of the storage needs for the geometry, whereas the Multi-Triangulation consumes more than five times more for connectivity and hierarchy. The factor of six between the pattern based approach and the Multi-Triangulation is not quite achieved. The first reason is the storage needs for the rules, which influences especially the smaller models. The second reason is that the models are not completely reduced.

The pattern based approach makes only sense if the time performance is still acceptable for real-time applications. A comparison of the selective refinement algorithms[6] for the pattern based approach and the Multi-Triangulation show that the assymptotic running time is the same. As shown in [Pup96] for vertex decimation with limited fragment size, the running time is linear in the size of the output triangulation. To validate the linear running time for the pattern based approach we measured the time performance of our refinement and

[6] The selective refinement algorithm refines the axiom until a demanded resolution criterion is fulfilled.

coarsening algorithm with a view-independent resolution criterion. We started with the axiom, then set the allowed global error to zero and refined to the original triangulation. Subsequently, we set the allowed error to infinity and coarsened the current triangulation down to the axiom. We measured the time needed for several repetitions of this process and computed the average number of vertices which can be removed or inserted per second. It turned out that our implementation of the pattern based approach is only slightly (a factor of 1.5) slower than the corresponding process for our own [KLS96] implementation of the Multi-Triangulation. Table 2 tabulates the measurements for each model.

vert./sec	MHz	cache	second	surface	femur	face	jaw	landsc.
size/KB				36	155	346	335	811
R4000	100	8 KB	1 MB	14,320	10,787	8,884	8,677	7,890
R4600	100	16 KB	–	15,629	11,509	9,978	9,547	9,613
R8000	75	16 KB	2 MB	23,515	21,900	21,704	20,890	18,222
R10000	175	32 KB	1 MB	60,325	36,689	24,075	23,674	17,481

Table 2. The table shows for each model the number of vertices, i.e. the number of fragments, which can be inserted or removed per second. The measurements were performed on different SGI workstations with different cache sizes.

The measurements were performed on different SGI workstations with different cache and secondary cache sizes. The necessary main memory increases from $67KB$ for the implicit surface to $1.5MB$ for the landscape model. In addition to the model the current triangulation has to be stored during the refinement and coarsening process. In our implementation the representation of the current triangulation in the highest resolution consumes about 1.7 times the storage needs of the connectivity and hierarchy of the pattern based multiresolution model. Together with the geometry data the consumed main memory reaches from $130KB$ to $2.9MB$. The model of the jaw-bone consumes approximately $1MB$.

The R4600 has no secondary cache and the R8000 has enough secondary cache to keep the complete model. On these two work stations the linear time performance is nicely demonstrated. The smaller models are slightly faster because of the primary cache of $16KB$. With a secondary cache size of $1MB$ for the R4000 and the R10000 the analyzed models partially fit and partially do not fit into the secondary cache. Thus the running time is not linear anymore in the size of the model. The larger primary cache of $32KB$ of the R10000 additionally speeds up the algorithms for small models, such that the implicit surface performs 3.5 times faster than the landscape.

In a typical real-time application for selective refinement, there are only a few updates to the current triangulation between two successive frames. Suppose we want 30 frames per second. The refinement and coarsening algorithms can insert or remove about $18,000$ vertices per second on an O2/R10000, thus about 600 vertices per frame. If we further suppose that not more than ten percent of the

triangles change between two successive frames and that we may consume about half of the CPU time for the refinement and coarsening algorithm, we can render a simplified scene with about 3,000 triangles in real-time.

5 Conclusion

All simplification algorithms based on vertex removement, edge or triangle collapse are well suited to build up the pattern based multiresolution model proposed in this paper. This multiresolution model supports selective refinement as necessary for view-dependent visualization. The high reduction rates of the storage costs achieved by the use of patterns allow one to process much larger models than with the other multiresolution models described in the literature[7]. We also showed that the time performance of selective refinement is only slightly decreased by the pattern based approach if compared to the Multi-Triangulation.

References

[CCMS97] A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno. Multiresolution decimation based on global error. *The Visual Computer*, 13(5):228–246, 1997. ISSN 0178-2789.

[CPS97] P. Cignoni, E. Puppo, and R. Scopigno. Representation and visualization of terrain surfaces at variable resolution. *The Visual Computer*, 13(5):199–217, 1997. ISSN 0178-2789.

[CVM+96] Jonathan Cohen, Amitabh Varshney, Dinesh Manocha, Greg Turk, Hans Weber, Pankaj Agarwal, Frederick P. Brooks, Jr., and William Wright. Simplification envelopes. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 119–128. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[dBD95] M. de Berg and K. T.G. Dobrindt. On levels of detail in terrains. Technical Report UU-CS-1995-12, Department of Computer Science, Utrecht University, April 1995.

[dFP95] Leila de Floriani and Enrico Puppo. Hierarchical triangulation for multiresolution surface description. *ACM Transactions on Graphics*, 14(4):363–411, October 1995.

[EDD+95] M. Eck, T. DeRose, T. D., H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 173–182. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[Gum98] S. Gumhold. Compression of discrete multiresolution models. Technical Report WSI–98–1, Wilhelm-Schickard-Institut für Informatik, University of Tübingen, Germany, January 1998.

[HDD+93] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Mesh optimization. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 19–26, August 1993.

[7] The reduction factor compared to Puppo's Multi-Triangulation [Pup96] or Hoppe's Progressive Meshes [Hop97] is about 6.

[HH92] Charles Hansen and Paul Hinker. Isosurface extraction SIMD architectures. In *Visualization'92*, pages 1–21, oct 1992.

[Hop96] H. Hoppe. Progressive meshes. In *Computer Graphics Proceedings, Annual Conference Series, 1996 (ACM SIGGRAPH '96 Proceedings)*, pages 99–108, 1996.

[Hop97] Hugues Hoppe. View-dependent refinement of progressive meshes. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 189–198. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[KHK96] R. Klein, , T. Hüttner, and J. Krämer. Viewing parameter dependent approximation of nurbs-models for fast visualization and animation using a discrete multiresolution representation. In B. Girod, editor, *Herbsttagung '96 3D Bildanalyse und -synthese*, 1996.

[KK97] R. Klein and J. Krämer. Multiresolution representations for surface meshes. In *Proceedings of the SCCG*, pages 57–66, 1997.

[KLR+95] David Koller, Peter Lindstrom, William Ribarsky, Larry F. Hodges, Nick Faust, and Gregory Turner. Virtual gis: A real-time 3d geographic information system. Technical Report 95-14, Graphics, Visualization and Usability Center, Georgia Institute of Technology, USA, 1995.

[KLS96] R. Klein, G. Liebich, and W. Straßer. Mesh reduction with error control. In R. Yagel, editor, *Visualization 96*. ACM, November 1996.

[KS96] R. Klein and W. Straßer. Generation of multiresolution models from cad-data for real time rendering. In W. Straßer, R. Klein, and R. Rau, editors, *Theory and Practice of Geometric Modeling)*. Springer-Verlag, 1996.

[KT96] Alan D. Kalvin and Russel H. Taylor. Superfaces:polygonal mesh simplification with bounded error. *IEEE Computer Graphics and Appl.*, 16(3), May 1996.

[MSS94] C. Montani, R. Scateni, and R. Scopigno. Discretized marching cubes. In R. D. Bergeron and A. E. Kaufman, editors, *Visualization '94 Proceedings*, pages 281–287. IEEE Computer Society, IEEE Computer Society Press, 1994.

[Pup96] E. Puppo. Variable reolution of terrain surfaces. In *Proceedings Eight Canadian Conference on Computational Geometry*, August 1996.

[RB93] J. Rossignac and P. Borrel. Multi-resolution 3d approximation for rendering complex scences. In B. Falcidieno and T. L. Kunii, editors, *Modeling in Computer Graphics: Methods and Applications*, pages 455–465. Springer Verlag, 1993.

[RKH96] Daniel Cohen-Or Reinhard Klein and Tobias Hüttner. Incremental view-dependent multiresolution triangulation of terrain, 1996. submitted to Pacific Graphics.

[RR96] R. Ronfard and J. Rossignac. Full-range approximation of triangulated polyhedra. *Computer Graphics Forum*, 15(3):C67–C76, C462, September 1996.

[SZL92] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 65–70, July 1992.

[Tur92] Greg Turk. Re-tiling polygonal surfaces. In Edwin E. Catmull, editor, *Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 55–64, July 1992.