



Consiglio Nazionale delle Ricerche

Technical Report

Resolution Modeling

P. Cignoni, C. Montani, C. Rocchini, R.
Scopigno

TR C97-02

January 20, 1997

CNUCE

Istituto del C.N.R.

Via S. Maria, 36 - 56126 - Pisa ITALY

Resolution Modeling

P. Cignoni¹, C. Montani¹, C. Rocchini⁴, R. Scopigno⁴

¹ Istituto per l'Elaborazione dell'Informazione - Consiglio Nazionale delle Ricerche
Via S. Maria, 46 - 56126 Pisa ITALY - Email: [cignoni|montani]@iei.pi.cnr.it

⁴ Istituto CNUCE - Consiglio Nazionale delle Ricerche
Via S. Maria, 36 - 56126 Pisa, ITALY - Email: r.scopigno@cnuce.cnr.it

Category: research

Format: print

Contact: Roberto Scopigno
CNUCE - CNR, Via S. Maria 36, 56126 PISA (Italy)
phone: +39 50 593304
fax: +39 50 904052
email: r.scopigno@cnuce.cnr.it

Estimated # of pages: 10

Keywords: I.3.5 [CR Descriptors: Computer Graphics]: Computational Geometry and Object Modeling - *Curve, surface, solid and object representation*; I.3.6 [Computer Graphics]: Methodology and Techniques.

Additional Keywords: surface modeling, mesh simplification, multiresolution, selective refinements, resolution modeling.

Very large graphics models are common in a number of applications, and many different simplification methods have been recently developed. Some of them support the construction of multiresolution representations of the input meshes. On the basis of this innovative techniques, we foresee a modeling framework based on three separate stages (shape modeling, multiresolution encoding and resolution modeling), and propose a new approach to the last stage, *resolution modeling*, which is highly general, user-driven and independent on the particular simplification approach used to build the multiresolution representation.

The approach proposed is based on a new representation scheme, the Glued Multiresolution Model (*GMM*), and a set of kernel functionalities which operate on the *GMM* to support: *a*) efficient extraction of fixed resolution representation, *b*) unified management of selective refinement and selective simplification, *c*) easy composition of the selective refinement/simplification actions, *d*) guaranteed C_0 continuity of the variable resolution mesh produced, and *e*) interactive response times.

The purpose of *resolution modeling* is not limited to visualization speedup, but it also supports interactive modeling of details and allows the specification of "semantic" views of the data. A prototypical *resolution modeling* system has been implemented and evaluated on several practical models.

For this reason we believe that *resolution modeling* has a strong similarity with shape modeling, in the sense that it has to be fulfilled through a tight interaction with the user. While the construction of the multiresolution representation is a process which can be simply made in an automatic and unattended way, the resolution modeling phase generally involves a semantic interpretation of the data which cannot be fulfilled without human intervention.

Given this framework, the rationale of this paper is to propose a new data structure and algorithms which allow the user to “model resolution” directly. Standard CAD systems provide tools to assist users in the design of “shapes”, but none of them actually provides the tools needed to manage what we would conceive as a second-stage modeling session: given a first stage where the “shape” is designed in full detail, then lets the user play with *resolution* to derive different instances of the the input shape which are characterised by variable resolutions/details.

Our global modeling conceptual model is characterised by three phases as follows;

1. **shape modeling**, the canonical three-dimensional shape design/acquisition/fitting;
2. **multiresolution model construction**, supported by recent surface simplification methods [2, 19, 6];
3. **resolution modeling**.

Our goal is therefore to design interactive tools which will support the user in a selective and incremental resolution modeling session. In conclusion, the major contributions of this paper to address the cited goals are:

- a new multiresolution representation for surfaces in 3D space, which supports compact storing and efficient access;
- a new methodology for the interactive management of selective refinements over the mesh (either reducing or increasing the mesh detail on mesh subareas chosen by the user);
- an implementation of this selective refinement operator via an efficient navigation over the multiresolution representation which allows partial sub-mesh updates;
- guaranteed C_0 continuity on each intermediate result (i.e. the current output mesh at variable resolution);

3D Studio Max system. The inherent limitations with the current implementations of the LoD paradigm are the limited number (generally very few) of different approximations which are stored, to reduce redundancy and space occupancy, and the non dynamic nature of the representation itself. The represented levels are generally built in a pre-processing step, to allow the fastest access to data in rendering. But the selection of the resolution would ideally depend on dynamic parameters [6, 4, 30], e.g. to ensure data-independent constant frame rates.

Examples of *multiresolution* representation schemes for digital terrains have been reviewed in a recent paper by De Floriani and Puppo [8]. Methods for the construction of multiresolution representations of generic surfaces in \mathbb{R}^3 has been proposed by adopting classical face-based approaches [19, 30, 2] or wavelet-based approaches [10, 17, 1].

LoD or multiresolution representations have been adopted in many applications: to reduce rendering time in visualization [15, 30, 3, 29, 24, 6, 4, 8]; to apply progressive transmission of 3D meshes on low bandwidth lines [19, 1]; to implement selective refinements on surfaces [6, 4, 19].

3 Construction of a multiresolution model

A surface mesh S may be simplified by following either a *refinement* or a *simplification* strategy. In both cases, a multiresolution output may be simply built if a *global error* is computed and maintained for each local modification action. In the following we take into account a simplification heuristics based on vertex decimation, but the same holds for other simplification or refinement heuristics as well.

Given a simplification approach based on vertex removal, we call: S , the input mesh; S_i , an intermediate mesh obtained after i steps of the simplification process; v the vertex candidate for removal on mesh S_i ; T_i the patch of triangles in S_i incident in v ; and, finally, T'_v the new triangulation which will replace T_v in S_{i+1} after the elimination of v .

At each step, we maintain the **global error**, i.e. the measure of the error of approximation introduced when the section of the original input mesh S corresponding to T_v is represented by the new mesh parcel T'_v (e.g. by computing a Hausdorff distance). Simplification approaches which adopt a *global* estimate of the error have recently been proposed [2, 21]. A global error is therefore associated with each of the new facets in

a model S_r with a complexity of $O(m + \log k)$ where k is the number of different errors in the *history*.

But the *history* representation is not sufficient when more sophisticated accesses to the multiresolution data have to be managed. To support the efficient implementation of a selective refinement operator, a more sophisticated multiresolution meshes representation is proposed in the following section.

4 Resolution Modeling

This section describes the data structures and the kernel functionalities needed to build the proposed *resolution modeling* system. The key functionality is to support selective refinement or simplification actions, where each action: modifies incrementally the current mesh by increasing/decreasing the resolution on a sub-area of the mesh which is interactively selected by the user; has to be performed in interactive time; has to guarantee C_0 continuity on each intermediate result (the current output mesh).

Being the process user-driven, a complex dialog session has to be managed. The *user* has control over: the action (*refinement* or *simplification*) that has to be operated; the *focus point* p_f on the current mesh; the current *radius* r , which identifies the area size (to be refined/simplified) surrounding p_f on the current mesh; and the function $Err()$ which determines, for each element of the mesh, the required increase/decrease of precision by taking into account the distance of the element from p_f .

According to user inputs, the system modifies locally the current mesh, by decreasing or increasing the mesh precision in the mesh subsection of radius r and centred in p_f .

The interface of *Zeta*¹, our resolution modeling prototypal system, is presented in Figure 12. In the snapshot, a mesh has been extracted at a low resolution. Few composed refinement actions were then operated in the areas of the head and the arms. Six different stages operated on the rabbit mesh (distributed by M. Levoy) are presented in Figure 11, to highlight some *Zeta*'s capabilities. The mesh colors in the second, third and fourth clips represent the error of each mesh face (using a color ramp from blue to green).

In the following Subsection, we propose the *Glued Multiresolution*, a general multiresolution representation. Then, Subsection 4.2 describes how to build or convert a multiresolution mesh into the *Glued Mul-*

¹The first release of *Zeta* is available on the World Wide Web at address ... temporarily removed for blind referee...

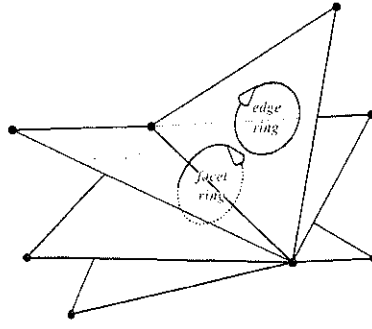


Figure 2: The standard facet-edge data structure: a *facet-edge* belongs to two rings.

In order to clarify the organization of the data, let us use a metaphor: we represent “visually” the adjacency between patches by representing a new patch as a curved bubble which shares with the removed one the chain of border edges (Figure 1). We can imagine that the resulting multiresolution data structure is built by warping each new patch of a delta value sufficient to contain the removed patch, and by welding it onto the old triangulation at the boundary of the influence region. The single simplification step is therefore visually represented as a new “bubble” popping up from the triangulation.

In this way, for each simplification step, we append the new mesh S_i to the multiresolution representation of the previous meshes S_0, \dots, S_{i-1} by (*metaphorically*) glueing all of the common faces. The resulting structure can be *topologically* (but not geometrically) interpreted as a 3D subdivision of the space (where a 3D cell corresponds to each bubble). This because geometry (i.e. vertices coordinates) is not modified at all.

The multiresolution scheme proposed, called *Glued Multiresolution Meshes* model (*GMM*), is designed following the metaphors above. It maintains in a compact format both the topological information, collected during the refinement process, and the information on the error of each triangle. The *GMM* scheme is encoded by adopting a *packed facet-edge* (PFE) representation.

The PFE has been designed as a modification of the *facet-edge*, a data structure originally introduced for the representation of 3D space subdivisions [9]. In the facet-edge scheme, an atomic entity is associated with each pair that is identified by a face f and one of its edges e : the so-called *facet-edge*. Each facet-edge denotes two rings: the *edge-ring*, composed by all the edges of the boundary of f ; and the *facet-ring*, composed by all the faces incident at e (see Figure 2). This structure is equipped with traversal functions that enable the complex to be visited. These functions are used to move from a facet-edge to an adjacent one,

PackedFacetEdge = **Record**

vert : \uparrow Vertex;

fother : \uparrow PackedFacetEdge;

size : **Short Integer**; (no. of incident faces)

pack : **Array**[1..*size*] of **Record**

face : \uparrow Face;

enext : \uparrow PackedFacetEdge

The actual *size* of each oriented edge e depends on the number of simplification steps which have e on the border of the simplification region. The incident faces are ordered in the variable length vector upon increasing error (as in the metaphor of the overlay of bubbles). Following the definition of the approximation error, the *life intervals* of two successive faces in a PFE have to be adjoining, and therefore the total *life interval* of a PFE e is delimited by the *birth error* of its first incident face and the *death error* of its last face. Moreover, by construction the life interval of a PFE is identical to the life interval of the adjacent one (linked by the *fother* field).

Basic access functions are provided in our implementation of the *GMM* scheme to compute properties or adjacency relations which are not explicitly stored. In particular, we define the following functions:

- *LifeInterval(e)*, which returns the total life interval of a PFE;
- *Star(e)* which, given a PFE e and its vertex v , returns the set of vertices which are adjacent to v on the maximal resolution mesh;
- *FindValidFace(e, ε)* which returns the face incident on a given PFE which satisfies approximation precision ε .

The traversal of the *GMM* data structure is implemented by alternating two different moves: traversing in the *spatial* domain and traversing in the *error* domain. In the first case, given an oriented PFE we want to move to the adjacent one on the opposite side of its oriented edge (through the *fother* link). In the latter, given an error, we want to adjust the precision by moving in the *pack* list of faces stored in the current PFE

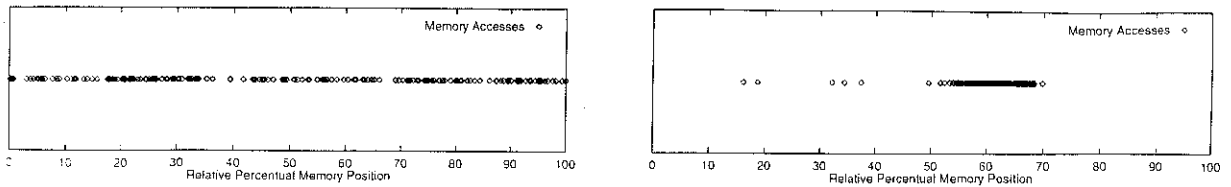


Figure 4: Distribution of the memory accesses in the case of not sorted record allocation (top) or sorted one (bottom).

4.3 Selective refinements on the *GMM* scheme

In this section we present the *selective refinement* algorithm. As outlined at the beginning of Section 4, the user-machine dialog starts with the selection of a constant resolution representation of the mesh (see Figure 11.a-f). A number of successive selective refinement/simplification actions, centered on user defined focus points, are then applied. Therefore, we introduce first the algorithm to extract a constant resolution mesh out of the *GMM* scheme. Then, we describe the approach chosen to compute distances on the mesh, which are needed to select the area onto which each selective refinement action has to be operated. The selective refinement algorithm is specified in the last subsection.

4.3.1 Extraction at constant approximation

The constant approximation extraction is similar to the one proposed for terrain multiresolution representations [4]. Each intermediate triangulation S_i is represented implicitly in *GMM* and it can be retrieved by visiting the *GMM*, starting from a PFE $e \in S_i$ and propagating from e by means of the PFE adjacencies. This approach is more efficient than visiting a hierarchical representation of the simplification actions operated on the mesh [6], especially where the refined/simplified mesh has to be recomputed on the basis of a history of updates (e.g. k vertex split actions, which reverse the effect of k edge collapsing actions [19]). Moreover, the locality of our algorithm (facets are given in output following adjacencies) makes it possible to speedup rendering by using a meshed representation of the output data (e.g. OpenGL triangle strips output primitives).

Specifically, the algorithm starts by searching for a first valid PFE, i.e. a PFE which contains the precision ε in its life interval. To implement this search efficiently, we build *off-line* the small subset *Seeds* of PFE's

To define the distance between two vertices v_1 and v_2 in a multiresolution model, we take into account the mesh at maximal resolution S_0 , which gives the best approximation of the geodetic distance. The precision of the *approximated geodetic distance* depends on the regularity of the tessellated representation. If the mesh is composed of equilateral triangles, then the approximation is at most $2/\sqrt{3} \cong 1.154$ times the precise geodetic distance.

To compute approximated geodetic distances we consider a triangulated mesh as a weighted graph, whose arcs are the mesh edges and whose arc weights are the length of the associated edges. We need to compute the minimal distances from the focus point, and therefore in graph terminology we need to solve a *Shortest Path Tree* (SPT) problem with a single source. A number of standard solutions exist; we adopted a modified version of the *SPT_S_Heap* algorithm [16]. To increase efficiency, the *SPT_S_Heap* algorithm is not executed on the entire S_0 mesh. It has been modified to visit and process only the edges contained in the spherical domain defined by p_f and r . This is simply done by avoiding further expansions of arcs which are generated from nodes whose distance from the source p_f is greater than r .

All of the vertex distances computed are stored in the *GMM* model (in the field *dist* of the *Vertex* records). The distance of a PFE is defined as the mean of the distances of the vertices incident on it.

4.3.3 The Selective Refinement algorithm

The selective refinement algorithm locally modifies the current mesh T , which can be either a surface extracted at constant error from the *GMM* model, or the composition of n subsequent selective refinements. Selective refinement starts with the user selection of: the current *focus point* p_f , the radius r of the current selective update and the error function $Err() : \mathbb{R} \rightarrow [\varepsilon_1, \varepsilon_2]$ which sets the error expected for each element f_i of the mesh in the active area as a function of the distance between f_i and the focus point p_f . Obviously, $\varepsilon_1 < \varepsilon_2$ implies a request for a local refinement, or a local reduction of the mesh resolution in the opposite case. In the current implementation, the user may interactively define the current function *Err* by editing its graph in the *Error Function* graph subarea (bottom-right of the *Zeta* main window, see Figure 12 and 11). For each selective refinement action, distances from the focus point are defined and computed as described in the previous subsection.

The Selective Refinement algorithm receives in input the multiresolution mesh *GMM* and the current

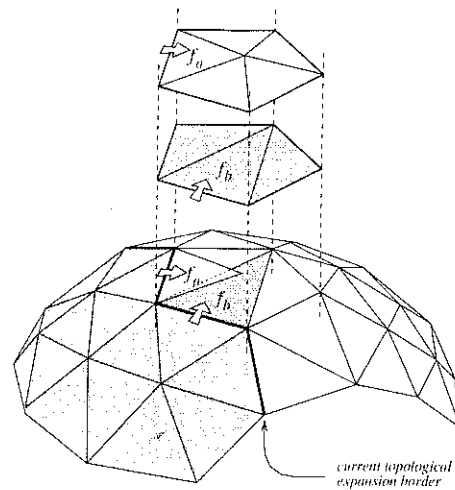


Figure 6: An inconsistency in variable approximation extraction: given two different expansion paths (f_a and f_b), the associated sub-meshes do not match.

- update the output mesh T by removing all of the facets that have been replaced with the new refined section (the *overlapStack* contains the border of the section which has to be removed).

FindFirstEdge function: the PFE nearest to the focus point which satisfies the error ε is returned by the *FindFirstEdge()* function. To guarantee efficiency, it has been implemented by adopting a spatial index on the PFE's (organized by partitioning the space into rectilinear sub-volumes). For each point in the space, FindFirstEdge retrieves efficiently a subset of PFE which are in the proximity of p_f , by visiting only a subset of the cells of the spatial indexing data structure. Then, starting from this set, it computes the vertex nearest to p_f . Finally, it chooses from the PFE's incident in the nearest vertex the one which is also nearest to p_f .

Priority queue management: the PFE extraction from the priority queue cannot be realized following a simple strategy, e.g. the order of insertion. In the case of a selective refinement, two different paths may lead to two different, not joining expansions, which give a multiple representation of the same surface sub-area (see an example in Figure 6).

This situation may occur if we expand facet-edges without taking into account the global current content of the priority queue. If, conversely, we impose the constraint that the life interval *intersection* for all the

- *error-based* popping: we extract the PFE whose error interval prevents the current priority queue life interval from being widened (i.e. the one whose interval gives rise to the bounds of the current priority queue interval). If we are *refining* the mesh, i.e. extracting a sub-mesh with an error which increases as we get far from p_f , then we extract the facet-edge with $\varepsilon_d = Q.\varepsilon_d^{\min}$; otherwise, if we are reducing resolution we select and extract the facet-edge with $\varepsilon_b = Q.\varepsilon_b^{\max}$;
- *composed distance-error based* popping: we follow a *distance-based* approach until we can extract facet-edges at the proper error; then, we use the *error-based* criterion until the current priority queue life interval has been sufficiently widened. This composed strategy is specified in the pseudo code of Figure 9.

The previous strategies were evaluated on a number of datasets. A comparison between the *distance-based* and the *composed* criteria is presented in the graphs of Figure 7 . A graphic element is plotted for each visited PFE, at abscissa equal to its distance from the focus point. The graphic element represents the life interval of the PFE as a vertical segment; the effective error value at which we have chosen one of its faces is represented by a diamond. If the diamond was drawn on one of the extremes of the segment, then during extraction we would be forced to choose one of the interval extremes (and in that case it would not be possible to choose the proper error according to the *Err* function).

In the runs reported in Figure 7 the user defined *Err* function increases from 0. to 0.21 in an interval of radius 120. The dashed line represents the error defined by the current *Err* function.

Facet-edge expansion and mesh update

For each facet-edge fe popped from the priority queue Q , if the distance of fe from p_f is greater than r , and fe is already contained in the current mesh T , then we do not proceed with the expansion of fe , and its adjacent facet-edge $fe.fother$ is inserted into the *overlapStack*.

If, conversely, fe is farther than r but is not contained in T , we cannot stop the expansion (to prevent C_0 discontinuities in the updated mesh T), and thus proceed with the expansion of fe .

In canonical conditions, at the end of the topological expansions the boundary of the old subsection of the mesh T which has been refined is stored in the *overlapStack*. The *DeleteOverlap()* function removes from T the redundant faces: it starts from the facet-edges contained in the *overlapStack* and removes all

```

Procedure SelectiveRefinement (GMM: GluedMultiRes,
    Err: Function(d: Real ), radius: Real, fp: Point, Var T: Triangulation);
Var Q : TripleHeap;
    overlapStack : Stack;
    edge : PackedFacetEdge;
    face : Face;
Begin
    overlapStack :=  $\emptyset$ ;
    edge := FindFirstEdge(GMM, fp, Err(0));
    SPT_S_Heap_Local(edge.vert, radius, GMM);
    Q := {edge}  $\cup$  {edge.fother};
    While Q  $\neq$   $\emptyset$  Do
        PopEdge&ChooseFace(GMM, Q, T, radius, edge,
            face, overlapStack);
        If edge  $\neq$   $\emptyset$  AND face  $\neq$   $\emptyset$  Then
            T := T  $\cup$  {face};
            For i:=1 To 2 Do Begin
                edge := edge.enext(face);
                If edge  $\in$  Q Then Q := Q - {edge};
                Else Q := Q  $\cup$  {edge.fother};
            DeleteOverlap(overlapStack,T)
    End;

```

Figure 8: Selective Refinement pseudo code.

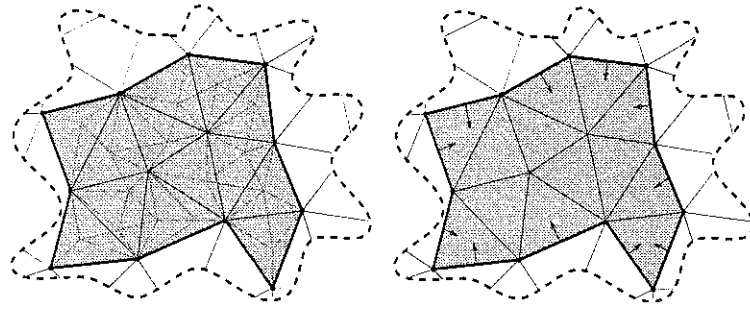


Figure 10: Updating the current T mesh, after local refinement, from the oriented border of the refined region stored in the *overlapStack*.

	$\eta=1$	$\eta=2$	$\eta=3$	$\eta=4$	$\eta=5$	$\eta=6$	$\eta=7$	$\eta=8$	η_{mean}
Bunny mesh (vert.:34,834 faces:189,351 facet-edges:568,053)									
<i>number</i>	241,066	99,139	28,821	7,529	1,821	401	67	14	1.49
<i>percentage</i>	63.6%	26.1%	7.6%	1.9%	0.4%	0.1%	0.01%	0.003%	
Sphere mesh (vert.:6,322 faces:33,801 facet-edges:101,403)									
<i>number</i>	43,788	18,427	4,744	1,114	322	63	11	1	1.48
<i>percentage</i>	63.9%	26.9%	6.9%	1.6%	0.47%	0.09%	0.01%	0.001%	

Table 1: Face adjacency factor evaluated on two different multiresolution meshes.

defined as follows.

Definition 5.1 *The face adjacency factor η is the number of faces which are adjacent to a PFE; the mean face adjacency factor η_{mean} is the mean of the η factors for all the faces of a mesh.*

The number of adjacent faces on each edge will depend on the shape of the original surface and of the simplification/refinement approach adopted to build the multiresolution mesh, and therefore it can only be determined empirically. Table 1 shows the number of faces with $\eta = 1..8$ and their percentage on the total number of faces in the multiresolution mesh.

Obviously, the larger η_{mean} is, the better the results of the selective refinement algorithm will be, because if a larger number of different options on each edge is available then the correct value of the error given by the $Err()$ function will be available with higher probability.

Empirical time complexity depends on a number of factors: the size of the multiresolution model, the value of the face adjacency factor, the current position of p_f on the mesh and the value of r , and the current Err function. The empirical complexity was measured on a number of meshes using an SGI Indigo2 XZ (200MHz R4400 CPU, 16Kb data and instruction cache, 1Mb secondary cache). We obtained:

- *constant precision extraction* from the *GMM* scheme at a rate of approximately 110K faces per second;
- *selective refinements* at a rate of approximately 15K faces per second, with a mean size of 200 - 1000 faces extracted for each selective refinement action.

Both the above rates were measured by not performing graphics output to the OpenInventor toolkit (which is used by the *Zeta* system to manage graphics output). The complexity of the meshes used in the tests was in the range 50K - 200K faces (e.g. the rabbit multiresolution mesh in Figure 11 holds 189K faces).

Even if we include the overhead due to the visualization of intermediate results by the OpenInventor toolkit, the system response is still sufficiently interactive, with response times in the order of a second even on a low performance workstation such as the system used.

A demonstration of the *Zeta* system throughput is in the accompanying videotape².

6 Applications

We believe that *resolution modeling* is a very general methodology, and the selective refinement approach proposed in the previous section is probably just one of the possible ways to support it. Consequently, the potential application domain is broad, and in the following we only glance to some possible uses.

We subdivide resolution modeling applications into two broad classes: those where resolution modeling can be managed in an unattended, computer-driven manner, and those which have to be operated under strict user control.

²The tape submitted is a low quality NTSC video directly taken from the ws monitor, and we apologize for that; a more professional video will be produced in case of acceptance of the paper

different variable resolution representations, which are built to take care of different presentation contexts (e.g. a foreground character which is either *half length* or *full length*).

Assembly Instructions

Assembly instructions for 3D compound objects or systems. To produce illustrations or animated sequences of an assembly, often the entire and highly complex description of each subcomponent is not needed or even counterproductive. A variable resolution representation may be adopted to convey a complete description of only the subcomponents that play a major role in the current assembly action, therefore enhancing the semantic importance of the most detailed part and reducing both image cluttering and rendering times.

Intermixing *emotional* and *informational* content

Resolution modeling may also be adopted to help the user in the selection or integration of *informational* and *emotional* graphics modalities. Alvy Ray Smith introduced his own characterisation of the graphics content of a [digital] media product [26]. He used the term **informational** to describe the typical SciViz way of presenting data, i.e. where visual presentation is oriented to increase the user's insight of a phenomenon under study. On the other hand, **emotional** describes those techniques which convey more "emotional" than informative content (e.g. computer art or the use of computer animation in advertising or movies).

We believe that a key point is the conscious distinction between informational and emotional content. We think that these two ways of conveying "information" are not orthogonal, and may be intermixed in a number of different applications. There are many different ways to render data, from the purely informational to those where the emotional content is predominant; the choice depends on the purposes of the visualization itself.

One possible use of resolution modeling may be to support the intermixing of informational and emotional representations in the same presentation/visualization. Given a resolution modeling tool, then we can design a class of visualization systems where the mixture of different presentation modes will be driven by the level of approximation of the variable resolution mesh.

Let us assume that the standard way to give an *informational* representation is the classical Z-buffered and

mesh produced; interactive response times.

A prototypal *resolution modeling* system has been implemented and evaluated on several practical models.

Unlike other approaches based on the selective inversion of simplification actions (vertex decimation or edge collapsing), our proposal is based on a complete multiresolution encoding, and selective refinements/simplifications are operated through a simple topology-based navigation of the multiresolution representation. Multiresolution encoding resulted in limited memory overhead. On the other hand, it ensures high efficiency in performing resolution modeling actions and easy incremental composition of the updates. Moreover, it can be paired with more sophisticated simplification methods (i.e. methods where each simplification step is not limited to a simple edge collapse or vertex decimation, see for example the use of *edge flipping* to improve the mesh quality [2]).

The purpose of *resolution modeling* is not limited to visualization speedup, but it also supports interactive modeling of details and allows the specification of “semantic” views of the data. Many possible applications have been sketched out, and we plan to further investigate on that.

Acknowledgments

We wish to thank Enrico Puppo of I.M.A. - CNR and Leila De Floriani of the University of Genova for many useful discussions and comments. This work was partially financed by the Progetto Coordinato “*Modelli multirisoluzione per la visualizzazione di campi scalari multidimensionali*” of the Italian National Research Council (CNR).

References

- [1] A. Certain, J. Popovic, T. DeRose, T. Duchamp, D. Salesin, and W. Stuetzle. Interactive multiresolution surface viewing. In *Comp. Graph. Proc., Annual Conf. Series (Siggraph '96)*, ACM Press, pages 91–98, Aug. 6-8 1996.
- [2] A. Ciampalini, P. Cignoni, C. Montani, and R. Scopigno. Multiresolution decimation based on global error. Technical Report C96-021, CNUCE – C.N.R., Pisa, Italy, July 1996.

- [3] P. Cignoni, L. De Floriani, C. Montani, E. Puppo, and R. Scopigno. Multiresolution modeling and rendering of volume data based on simplicial complexes. In *Proceedings of 1994 Symposium on Volume Visualization*, pages 19–26. ACM Press, October 17-18 1994.
- [4] P. Cignoni, E. Puppo, and R. Scopigno. Representation and visualization of terrain surfaces at variable resolution. In R. Scateni, editor, *Scientific Visualization '95 (Proc. Inter. Symposium on)*, pages 50–68. World Scientific, 1995.
- [5] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright. Simplification envelopes. In *Computer Graphics Proc., Annual Conf. Series (Siggraph '96)*, ACM Press, pages 119–128, Aug. 6-8 1996.
- [6] M. de Berg and K. Dobrindt. On levels of detail in terrains. In *11th ACM Computational Geometry Conf. Proc. (Vancouver, Canada)*, pages C26–C27. ACM Press, 1995.
- [7] L. De Floriani. A pyramidal data structure for triangle-based surface description. *IEEE Comp. Graph. & Appl.*, 9(2):67–78, March 1989.
- [8] L. De Floriani, P. Marzano, and E. Puppo. Multiresolution models for topographic surface description. *The Visual Computer*, 12(7), 1996.
- [9] D.P. Dobkin and M.J. Laszlo. Primitives for the manipulation of three-dimensional subdivisions. *Algorithmica*, 4:3–32, 1989.
- [10] M. Eck, T. De Rose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *Computer Graphics Proc., Annual Conf. Series (Siggraph '95)*, ACM Press, pages 173–181, Aug. 6-12 1995.
- [11] H. Edelsbrunner. Dynamic data structures for orthogonal intersection queries. Report F59, Inst. Informationsverarb., Tech. Univ. Graz, Graz, Austria, 1980.
- [12] Gershon Elber. Line art rendering via a coverage of isoparametric curves. *IEEE Trans. on Visualization and Computer Graphics*, 1(3):231–239, 1995.

- [24] Lori Scarlatos and Theo Pavlidis. Hierarchical triangulation using cartographics coherence. *CVGIP: Graphical Models and Image Processing*, 54(2):147–161, March 1992.
- [25] William J. Schroeder, Jonathan A. Zarge, and William E. Lorensen. Decimation of triangle meshes. In Edwin E. Catmull, editor, *ACM Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 65–70, July 1992.
- [26] A. R. Smith. Bringing visualization to the user (keynote speech). In R. Yagel and G. Nielson, editors, *Proceedings of Visualization '96*, page 16, 1996.
- [27] Greg Turk. Re-tiling polygonal surfaces. In Edwin E. Catmull, editor, *ACM Computer Graphics (SIGGRAPH '92 Proceedings)*, volume 26, pages 55–64, July 1992.
- [28] *The Virtual Reality Modeling Language Specifications, Version 2.0*, Aug. 1996. (available on the web at <http://vrmf.sgi.com>).
- [29] J. Wilhelms and A. van Gelder. Multi-dimensional Trees for Controlled Volume Rendering and Compression. In *Proceedings of 1994 Symposium on Volume Visualization*, pages 27–34. ACM Press, October 17-18 1994.
- [30] J.C. Xia and A. Varshney. Dynamic view-dependent simplification for polygonal models. In R. Yagel and G. Nielson, editors, *IEEE Visualization '96 Proc.*, pages 327–334, 1996.