International Journal of Image and Graphics
Vol. 7, No. 2 (2007) 273–290

World Scientific
www.worldscientific.com

DUAL ADAPTIVE PATHS FOR MULTIRESOLUTION HIERARCHIES

YOTAM LIVNY*, NETA SOKOLOVSKY† and JIHAD EL-SANA‡

Department of Computer Science
Ben-Gurion University of the Negev, Beer-Sheva, Israel
**livnyy@cs.bgu.ac.il*
†*netaso@cs.bgu.ac.il*
‡*el-sana@cs.bgu.ac.il*

Received 12 July 2005
Revised 1 June 2006
Accepted 13 August 2006

The recent increase in the generated polygonal dataset sizes has outpaced the performance of graphics hardware. Several solutions such as multiresolution hierarchies and level-of-detail rendering have been developed to bridge the increasing gap. However, the discrete levels of detail generate annoying popping effects, the preliminaries multiresolution schemes cannot perform drastic changes on the selected level of detail within the span of small number of frames, and the current cluster-based hierarchies suffer from the high-detailed representation of the boundaries between clusters. In this paper, we are presenting a novel approach for multiresolution hierarchy that supports dual paths for run-time adaptive simplification — fine and coarse. The proposed multiresolution hierarchy is based on the fan-merge operator and its reverse operator fan-split. The coarse simplification path is achieved by directly applying fan-merge/split, while the fine simplification route is performed by executing edge-collapse/vertex-split one at a time. The sequence of the edge-collapses/vertex-splits is encoded implicitly by the order of the children participating in the fan-merge/split operator. We shall refer to this multiresolution hierarchy as fan-hierarchy. Fan-hierarchy provides a compact data structure for multiresolution hierarchy, since it stores 7/6 pointers, on the average, instead of 3 pointers for each node. In addition, the resulting depth of the fan-hierarchy is usually smaller than the depth of hierarchies generated by edge-collapse based multiresolution schemes. It is also important to note that fan-hierarchy inherently utilizes fan representation for further acceleration of the rendering process.

Keywords: Geometric simplification; levels of detail; multiresolution hierarchies; view-dependent rendering.

1. Introduction

Polygonal mesh representations dominate the three-dimensional modeling from computer games and entertainments to modeling of aircrafts and submarines. The simplicity and compactness (compare to other representations) play a major rule in this dominance. In addition, the advances in computer graphics hardware have

been geared toward rendering polygonal meshes. Current graphics hardware on a commodity PC can render millions of polygons per second. However, the advances in three-dimensional shape acquisition, simulation, and design technologies have led to the generation of large polygonal datasets that exceed the interactive rendering capabilities of current graphics hardware. Several software algorithms and techniques, such as level-of-detail rendering and occlusion culling, have been developed to bridge the increasing gap between the currently available datasets and hardware rendering capabilities by reducing the complexity of the graphics datasets while keeping their visual appearance similar to the original.

Multiresolution hierarchies and view-dependent rendering enable the selection of various levels of detail over different regions of the model surface based on view parameters, such as viewpoint, illumination, and speed of motion. However, most of these multiresolution hierarchies are based on edge-collapse, which is inadequate for drastic change on the selected level of detail within the span of small number of frames. In addition, edge-collapse cannot adapt easily to the rate of changes between the various levels of detail. These limitations generate severe artifacts when the adapt process fails to select the appropriate level of detail quickly (to match the changing view parameters). For example, a fast fly over a large terrain could result in a high level of detail behind the viewer instead of his/her front.

Cluster-based multiresolution schemes try to overcome the above mentioned limitations by using aggressive simplification operators to quickly select the appropriate level of detail. These approaches start by dividing the dataset into disjoint regions, called clusters or patches, in a hierarchical manner and then simplify each cluster independently. To avoid generating cracks and fold-overs at real-time, the boundaries of the clusters are not simplified until they become internal to an ancestor cluster up the hierarchy. Therefore, some boundary regions may remain in the original resolution even in coarse levels of detail. These boundaries could account for large fraction of the triangle budget for coarse levels of detail. One could simplify these boundaries in real-time by performing edge-collapses, however such an approach could harm the frame rates and usually requires complex dependencies among vertices and clusters. In addition, most of cluster-based multiresolution schemes do not support view-dependent rendering within clusters (clusters are represented either in discrete levels of detail or linear progressive meshes). For that reason popping artifacts are not completely inevitable. Furthermore, some subdivision schemes could generate clusters that include disconnected regions that could not be simplified during the off-line construction process. The same problem arises when the dataset consists of large number of small objects.

To overcome these severe drawbacks we have developed a dual-paths adaptive operator, which enables fine as well as coarse changes on the selected level of detail (as shown in Fig. 1). The simplification rate is chosen based on view parameters and the difference between current and target levels of detail while maintaining a smooth change of the resolution over the model.

Fig. 1. The distribution of the fine (in black color) and coarse changes (in gray color) over a selected level of detail, and unchanged areas (in white color) of the Horse model. The view direction is near the head looking toward the body of the horse.

Our adaptive scheme is based on the *fan-merge* simplification operator which is performed by merging fan of vertices into one vertex (the fan center). This operation eliminates more triangles than the edge-collapse operator in one execution. To support fine changes, our fan-merge operator has an alternative path, which is performed by executing a sequence of edge-collapses one at a time. It is important to notice that the two paths generate the same mesh upon the completion of the operation.

The fan-based multiresolution hierarchy is constructed bottom-up by iteratively performing fan-merge on the vertices of the mesh. The resulting data structure encodes the various levels of detail of the original mesh in a hierarchal manner. We shall refer to this multiresolution hierarchy as *fan-hierarchy*.

In the rest of this paper, we first overview related work in the area of level-of-detail rendering and multiresolution hierarchies. Then, we discuss the fan-hierarchy data structure, its off-line construction, and its real-time processing. We present implementation details and experimental results followed by some conclusions.

2. Related Work

In this section, we shall briefly overview related work in the fields of polygonal mesh simplification, multiresolution hierarchies, and view-dependent rendering.

Adaptive level-of-detail rendering has been developed to select an appropriate level of detail in a view-dependent manner by utilizing temporal coherence. Such a scheme has been widely used in terrain rendering by Cignoni *et al.*,[1] Duchaineau *et al.*,[2] Gross *et al.*,[3] and Lindstrom *et al.*[4]

View-dependent rendering approaches usually rely on multiresolution hierarchies that encode various levels of detail of the original mesh. Progressive meshes have been introduced by Hoppe[5] to provide continuous resolution representations of polygonal meshes. Merge trees have been introduced by Xia *et al.*[6] as a data structure built upon progressive meshes to enable real-time view-dependent rendering. Hoppe[7] has developed a view-dependent simplification algorithm that works with progressive meshes. His algorithm uses the screen-space projection and orientation of the polygons to guide the run-time view-dependent simplification.

Several data structures have been employed to efficiently support view-dependent approaches. Luebke and Erikson[8] have defined a tight octree over the vertices of the given model to generate hierarchical structure. De Floriani *et al.*[9] have introduced Multi-Triangulation (MT). Decimation and refinement in MT are achieved through a set of local operators that affect fragments of the mesh. Guéziec *et al.*[10] have demonstrated a progressive encoding scheme for surfaces in the form of a directed acyclic graph (DAG). Klein *et al.*[11] have developed an illumination-dependent refinement algorithm for multiresolution meshes. Pajarola[12] has introduced an efficient hierarchical multiresolution triangulation framework based on a half-edge triangle mesh data structure.

The real-time selection of level of detail is performed on the CPU and could be expensive for large datasets. For that reason several approaches have been developed to reduce the real-time CPU overhead. Kim and Lee[13] have managed to remove the dependency limitation of the split and merge operations. In their refinement scheme, each vertex-split or edge-collapse can be performed without incurring additional vertex-split and/or edge-collapse transformations. El-Sana *et al.*[14] have developed a data structure SkipStrip that efficiently maintains triangle strips during view-dependent rendering. For further reduction of the rendered mesh El-Sana *et al.*[15] and Yoon *et al.*[16] have integrated occlusion culling within the view-dependent rendering framework.

Different schemes for multiresolution hierarchies are based on various simplification operators, such as edge-collapse,[6,7,10,17] half-edge collapse,[12] cell-collapse,[8] and vertex-removal.[9] The above operators belong to the category of fine operators that affect small regions of the selected mesh.

View-dependent rendering schemes usually require the existence of the entire dataset in main memory. To overcome the memory size drawback El-Sana and Chiang[18] and DeCoro and Pajarola[19] have developed an external memory view-dependent simplification. Shamir *et al.*[20] have developed a view-dependent approach that handles dynamic environments with arbitrary internal deformation.

View-dependent rendering often fails to select the appropriate level of detail within the span of one frame. To overcome this limitation recently developed

approaches support aggressive operators on the selected level of detail. They divide the model into disjoint regions, called clusters or patches, which are simplified independently and stored in a hierarchical manner. For example, Erikson and Manocha[21] have used a hierarchy of static levels of detail, Cignoni *et al.*[22] have developed the adaptive TetraPuzzles for out-of-core view-dependent rendering, and Yoon *et al.*[23] have introduced Quick-VDR for interactive view-dependent rendering of large models.

3. Our Approach

Smooth changes on the selected level of detail are achieved by fine-change operators which update a small number of triangles. However, to carry out coarse changes we have to perform multiple passes over the active nodes (or recursively refine/simplify nodes). For that reason, fine-change operators fail to perform coarse changes on the selected level of detail within the span of one frame without compromising frame rates. Replacing these operators by more aggressive ones would generate undesired popping artifacts at close-to-viewer regions. Therefore, in view-dependent rendering approaches there is a need to support fine and coarse operators.

In this paper, we present the *fan-hierarchy* — a novel multiresolution hierarchy for view-dependent rendering. Our approach uses fan-merge operator for coarse changes on the selected level of detail and utilizes the advantages of aggressive simplification operators (similar to cluster-based schemes). In addition, it supports fine changes on the selected level of detail by providing a dual path for fan-merge as a sequence of edge-collapses. Thus, our approach provides smooth changes on the level of detail over the visualized surface.

Next, we discuss the fan-merge operation followed by the construction of the fan-hierarchy and the real-time rendering.

3.1. *Fan-merge*

The *fan* of vertex v is defined as the set of triangles adjacent to v, and the *link* of vertex v is defined as the set of vertices directly adjacent to v. We shall use the notation $fan(v)$ and $link(v)$ for the fan and link of vertex v, respectively.

We define the *fan-merge* operation for a vertex v as the merge of all the vertices in the link of v to the vertex v (as shown in Fig. 2). A fan-merge operation on

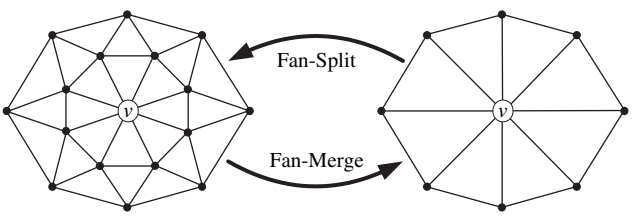


Fig. 2. The fan-merge and its dual operator fan-split.

a vertex v degenerates all the triangles which have two vertices in $link(v)$, and modifies all the triangles that have only one vertex in $link(v)$. The degenerated triangles are removed and the modified triangles are extended to cover the area of the degenerated ones. We shall refer to the set of modified triangles as $modified(v)$ and the removed degenerate triangles as $removed(v)$. We could mathematically represent these two sets as:

$$removed(v) = \{t | t \in \ fan(v) \text{ or } t \in \ adjacent(fan(v))\}$$
$$modified(v) = \cup_{u \in link(v)} \text{fan}(u) - removed(v).$$

The number of triangles removed by the fan-merge operation is equal to $2deg(v)$ (twice the degree of the vertex v). The fan-merge operation removes more triangles than edge-collapse upon the completion of one execution (edge-collapse usually removes two triangles). However, edge-collapse allows finer changes on the level of detail, which is often required for close-to-viewer regions. The fine changes are achieved by performing the fan-merge operation as a sequence of edge-collapses. In such a scheme, we provide two choices to perform fan-merge that result in the same simplified mesh.

3.2. *Error metric*

The execution of a fan-merge on a mesh usually introduces geometrical errors. For better mesh representation we perform fan-merges in a decreasing order of the introduced errors. To estimate these errors we use the quadratic-error metric[24] with small modifications for the fan-merge structure.

The quadratic-error metric provides a very good estimation of the distance between a point and a set of planes. According to this metric, a vertex u is associated with a set of planes and its error $\delta(u)$ is defined as the sum of squared distances to these planes, as shown in equation Eq. (1), where $p = [a\ b\ c\ d]^T$ represents the plane defined by equation $ax + by + cz + d = 0$ and $a^2 + b^2 + c^2 = 1$.

$$\delta(u) = \sum_{p \in planes(u)} (p^T v)^2 \tag{1}$$

During the fan-merge operation on vertex v the position of v remains constant while vertices of $link(v)$ are removed. Therefore, to estimate the introduced error $\Delta(v)$ of fan-merge of v we extend the previous metric by applying it on each vertex $u \in link(v)$ and accumulating the results, as shown in equation Eq. (2).

$$\Delta(v) = \sum_{u \in link(v)} \delta(u) \tag{2}$$

Recall that the modified triangles are extended to cover the area of the degenerated ones and for that reason they play a major role in determining the introduced geometrical error. Therefore, the changes on the modified triangles are used to estimate the quadratic error. To clarify that, let us consider the case where each

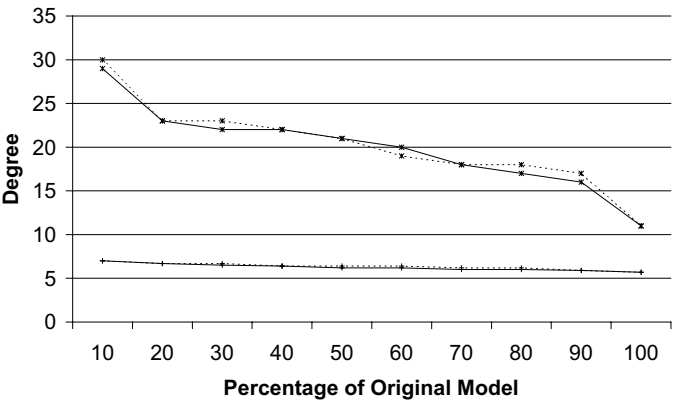


Fig. 3. These graphs show the behavior of maximum and average vertex degrees (marked with [∗] and [+], respectively) as a function of the simplification percentage using the Bunny model. The solid-line and dotted-line graphs show the results of the fan-merge and edge-collapse operators, respectively.

removed triangle is coplanar with an extended triangle, then the introduced geometric error is zero. Such a case occurs when the modified triangles are coplanar with the fan's central vertex.

The error metric we use is practically the quadratic-error metric on the modified triangles of the merged fan. This metric is also used to order the vertices of the mesh for fan-merge execution during the off-line construction process.

According to our experimental observation on various datasets, the fan-merge simplification based on quadratic error does not generate vertices with high degree. It is usually comparable to the degree of vertices in an edge-collapse simplification (see Fig. 3). Nevertheless, one could reduce the degree of vertices by assigning for each vertex a weight according to the number of its adjacent triangles. Such approach merges fans with small degrees earlier than those with high degrees among similar quadratic-error fans. It is important to select the weight carefully to comply with the principal of the quadratic-error metric.

3.3. *Fan-hierarchy*

The fan-merge operator is used to construct multiresolution fan-hierarchies that support view-dependent rendering for triangle meshes. Fan-hierarchies are constructed off-line by iteratively applying fan-merge operator in a bottom-up fashion.

To construct a fan-hierarchy H for a polygonal mesh M our algorithm starts by ordering the vertices of M in a priority heap based on the modified quadratic-error metric. Then, it iteratively extracts the vertex v from the top of the heap and performs two procedures: one on the simplified mesh and the other on the constructed fan-hierarchy (see Fig. 4). On the simplified mesh it merges the fan of v, updates the error value for the vertices in the newly created $link(v)$, and

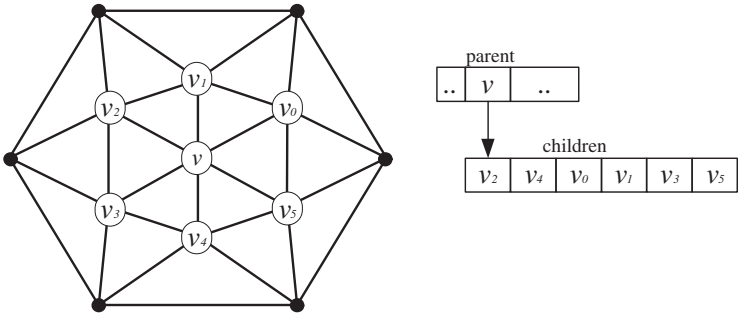


Fig. 4. The geometric fan (left) and its representation on the fan-hierarchy (right).

rearranges the priority heap accordingly. On the constructed fan-hierarchy H our algorithm performs the following steps:

- creates a node N_v that represents v,
- stores in N_v the relevant changes on the mesh as a list of modified triangles and a list of removed triangles,
- inserts N_v into H as the parent of the nodes which represent the vertices in $link(v)$,
- and orders the children of N_v by the introduced error as an appropriate sequence of edge-collapses (that forms the alternative path of simplification).

The algorithm stops when the priority heap becomes empty, the number of vertices reaches a predefine threshold, or the introduced error exceeds an upper limit (the pseudo-code of the construction algorithm is shown in Fig. 5). The resulting data structure has the form of a forest since not all the fans can merge. Each node of the fan-hierarchy stores the error value of the fan-merge, pointer to the list of children, pointer to the parent node, and two lists of indexes that correspond to the modified and removed triangles, respectively.

```
buildFanHierarchy(Mesh mesh, Metric metric){
 mesh.computeErrorMetric(metric);
 Heap heap = generateVerticesHeap(mesh);
 while (!heap.isEmpty()){
 Vertex v = heap.extract();
 node(v).setChildren(link(v));
 mesh.fanMerge(v);
 heap.updateErrorMetric(link(v), metric);
 }
}
```

Fig. 5. Fan-hierarchy construction algorithm.

To prevent the occurrence of fold-overs we have used the scheme of implicit dependences[17] with minor changes to adapt to the structure of fan-hierarchy. We use the same enumeration strategy during the off-line construction process and the same rules to prevent fold-overs at real-time.

The fan-hierarchy differs from previous multiresolution schemes in following ways:

- It can handle multiple simplification paths that perform different change rates.
- Its average depth is $\log_6(n)$ for n vertices comparing to $\log_2(n)$ for edge-collapse based multiresolution hierarchies.[a] Such a small depth enables quick coarse changes on the selected level of detail.
- It is compact as a result of storing less pointers for each node: 7/6 pointers, by average, instead of three pointers in view-dependence trees.[17] In our implementation each node has a pointer to an array of its children (six in average). The children nodes have a common pointer to their parent.

Similar to previous multiresolution schemes, fan-hierarchy encodes the various levels of detail of the original mesh. Regions that are close to the top of the hierarchy correspond to coarse levels of detail (low resolution) and regions that are close to the bottom of the hierarchy correspond to fine levels of detail (high resolution).

3.4. *Real-time traversal*

The fan-hierarchy, which is constructed off-line, is used at real-time to guide the selection of appropriate level of detail in an adaptive manner. A valid level of detail forms a breadth cut on the multiresolution fan-hierarchy. We shall refer to this cut as the *active-nodes list*. Note that active-nodes list can pass through regions with different levels of detail while maintaining smooth and transparent transition.

Our real-time traversal utilizes temporal coherence among consecutive frames by traversing only the active nodes (at each frame) and updates the selected level of detail to match the current view parameters. At each frame and for each active node the algorithm verifies the appropriateness of its level of detail and determines if there is a need to increase, decrease, or keep the current resolution. The decrease and increase of the resolution are performed by moving the slice up or down the fan-hierarchy, respectively.

In real-time interactive visualization, the visual error tolerance varies across different regions of the visualized surface based on view parameters, such as viewpoint, view direction, and illumination. Thus, the visual error in vertex v is computed using the equation Eq. (3), where $dist(v, C)$ is the distance between v and the viewpoint C, N_v is the normal at v, and D_v and D_L are the view and light

[a]According to the Euler's Formula the average degree of a vertex in planner graph is six.

directions, respectively. Note that the case where $|N_v \cdot D_v| < \epsilon$ in the equation is designed to emphasize the silhouette of the model.

$$error(v) = \begin{cases} dist(v, C) * |N_v \cdot D_v| * |N_v \cdot D_L| & \text{if } |N_v \cdot D_v| > \epsilon \\ dist(v, C) * |N_v \cdot D_L| & \text{if } |N_v \cdot D_v| < \epsilon \end{cases} \quad (3)$$

The visual error tolerance is small for close-to-viewer regions and large for far-from-viewer regions. In addition, the visual error tolerance also varies with respect to the speed of motion. We could tolerate larger error for a fast flight over a terrain than for a slower one. Consequently, the visual error tolerance is usually proportional to the speed of motion. Fine changes are required for low visual error tolerance and could be performed by edge-collapse (the execution of the operator removes two triangles). Coarse changes on the selected level of detail are possible for high visual error tolerance and could be executed by the fan-merge operation that manages to remove about 12 triangles by average (if we consider six as the average degree in a typical triangle mesh).

The real-time traversal selects the rate of changes over the different regions of the active level of detail based on the various error tolerances. The rate of changes is controlled by selecting the appropriate number of edge-collapses for fine rates and fan-merges for coarse rates. In such a scheme, the execution of a fan-merge as a sequence of edge-collapses can spread over several frames. To maintain the quality of the simplified mesh the edge-collapses are performed in an increasing order of the introduced error. At run-time the state of each fan is maintained by storing an index that marks the last collapsed edge (as shown in Fig. 6).

The switch between the two paths depends on the difference between the current level of detail and the target level of detail. If the difference is large we use the fan-merge, otherwise we execute several edge-collapses until we reach the appropriate level of detail (as shown in Fig. 7).

For further optimization, we use heuristics to estimate the selection of the adaptive path without testing the difference between current and appropriate levels of detail. The heuristics are based on the following observations:

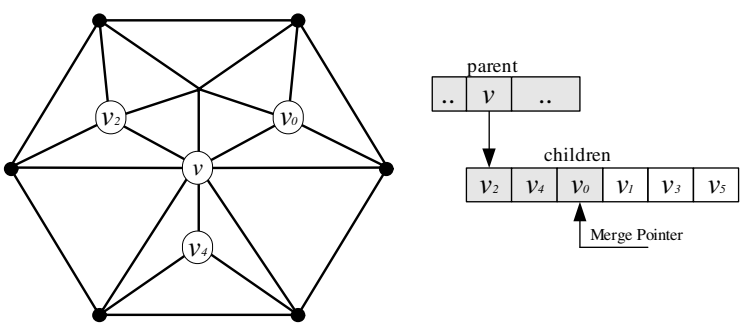


Fig. 6. Fine-rate changes on a fan through a sequence of edge-collapses (left) and the reflection of these changes on the fan-hierarchy (right). The merged nodes are marked by a white background.

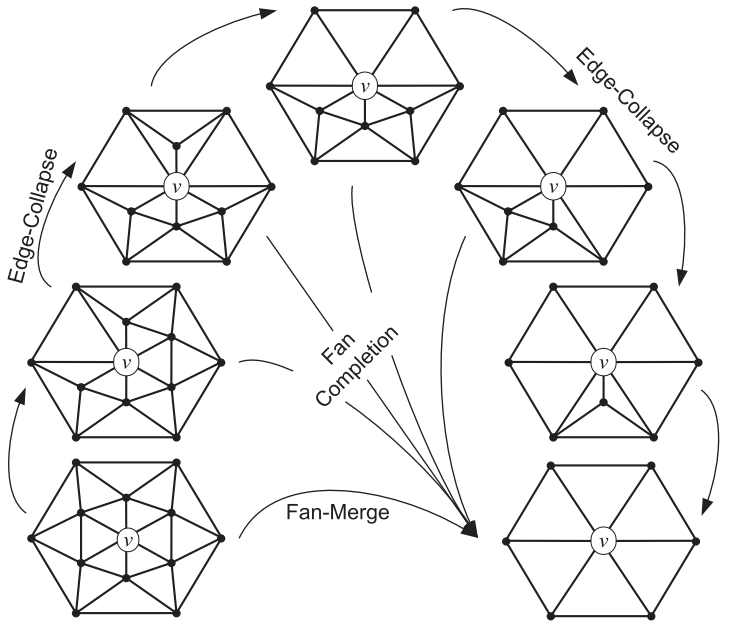


Fig. 7. The same simplification is performed either directly by a fan-merge or through a sequence of edge-collapses.

- Far-from-viewer regions have coarse level of detail and close-to-viewer regions have fine level of detail.
- We would like to perform fine changes on the level of detail for close-to-viewer regions and coarse changes for regions which are far from the viewer.

4. Implementation Details

We have implemented our algorithm in C++, performed several tests on various datasets, and received encouraging results for our implementation.

Each fan-hierarchy node includes one pointer to its parent, one-byte merge pointer, an error value, two lists of pointers to the modified and removed triangles, and an array of pointers to its children nodes. In addition, each node includes a pointer to its represented vertex properties, such as coordinates, color, and normal. To perform an edge-collapse we need to determine the modified and removed triangles. To extract these triangles, we use the lists of modified and removed triangles in the parent node and rely on the index of the node (on the children array).

The fan-merge operation is performed based on the error value of the parent node, and the fan-split operation depends on the error value in the last child (the one with the highest error value). Currently, the selection of the level of detail for each node is determined based on the distance from the viewpoint, view-direction, and the stored error.

To accelerate the scanning of the active-nodes list at real-time, we have adopted a linked list of blocks where each block includes an array of 256 pointers. In addition, the activated nodes are added to the end of the array, and the deactivated nodes are replaced with the last active node (and the last block counter is decremented).

The rendering is performed either at the end of the scanning of active nodes or at the expiration of the adapt time budget which is a predefined time slice that aims to maintain interactive rates. Our multiresolution fan-hierarchy inherently maintains the mesh as a collection of triangle fans. Therefore, we accelerate the rendering of the selected level of detail by sending the mesh as a set of fans with no overhead cost. The incomplete fans (as results of the fine rate of changes) are treated separately.

5. Experimental Results

We have performed several experimental tests on various datasets of different types. In this paper, we present some of these results that were achieved on a Pentium4 with 2.4 GHz, 512 MB of memory, and nVidia GeForce4 FX 5200 graphics hardware with 64 MB texture memory.

Table 1 shows the running time of the off-line construction algorithm for fan-hierarchy (the *FH* column) and view-dependence tree (the *VDT* column). As can be seen our preprocessing algorithm runs more than 10 times faster than the one for view-dependence tree. In addition, our algorithm runs in nearly linear time with respect to the number of vertices of the model.

Table 2 shows the size of the fan-hierarchy and view-dependence trees for various datasets. The number of internal nodes of the view-dependence trees is 4–6 times the number of the fan-hierarchy's internal nodes for the reported datasets. It is important to notice that the small number of internal nodes is valuable to external memory view-dependent rendering algorithms. Even if we compare the total size of the hierarchies (the internal and leaf nodes), the size of the view-dependence tree is 1.3–1.7 times the size of the fan-hierarchy as shown in the last column of Table 2.

The fan-hierarchy approach accelerates the selection of adaptive levels of detail as well as the rendering of the selected levels. The run-time performances of fan-hierarchy and view-dependence tree are reported in Table 3. These results were

Table 1. Preprocessing time for various datasets.

	Vertices	Triangles	Preprocessing time (sec)	
Dataset	(K)	(K)	FH	VDT
Terrain	262	522	8:20	92:52
Hand	327	654	11:45	125:37
Buddha	543	1087	16:23	182:21
Scene	872	2449	12:34	134:33
A. Dragon	3609	7218	34:02	413:23
Armadillo	3390	7500	47:15	472:47

Table 2. Fan-hierarchy size versus view-dependence tree size.

	Fan-Hierarchy		View-Dependence Tree		
Dataset	Int. Nodes (K)	Size (KB)	Int. Nodes (K)	Size (KB)	Saving Factor
Terrain	52	23800	262	32000	1.35
Hand	60	30200	327	47700	1.58
Buddha	85	45100	543	72100	1.60
Scene	174	91500	1224	142500	1.60
A. Dragon	520	263900	3609	419200	1.59
Armadillo	553	289400	3750	479200	1.66

obtained over a sequence of frames with a screen error of two pixels. The run-time comparison between fan-hierarchy and view-dependence tree is performed by measuring the average number of adapt operations per frame, the average adaptation time per frame, and the generated frame rates using the same camera path and illumination for each dataset. In the section of the average number of adapt operations (*Adapt Operations*) we report for the fan-hierarchy the number of fan-merge (the left number) and edge-collapse (the right number) operations. For the view-dependence trees we report the number of edge-collapse operations. As can be seen by using fan-hierarchy we have managed to achieve 3–5 improvement in the number of adapt operations, 2–3.25 reduction in average adaptation time, and 1.2–2 frame rate acceleration comparing to view-dependence tree. It is important to note that the frame rate acceleration is increased as the datasets grow larger. This behavior is an immediate result of the fan-hierarchy's efficient adapt processing. The last column of the Table 3 shows the average number of triangles sent to graphics hardware every frame.

Figures 8, 9, and 10 show images generated by our system. The arrow in Fig. 8(a) and Fig. 9(a) marks the view-direction. Figure 8 shows three images of the same selected level of detail of the Armadillo model. The left image (Fig. 8(a)) shows the distribution of coarse (fan-merge) and fine (edge-collapse) rate of changes that

Table 3. Adaptive operators number and time performance for fan-hierarchy (FH) and view-dependence trees (VDT).

	Adapt Operations		Adapt Time		Frames		Number
Dataset	FH Fan:Edge	VDT	FH (ms)	VDT (ms)	FH	VDT	of trian. (K)
Terrain	312:101	1875	6	21	65	46	95
Hand	438:95	2462	8	25	59	38	92
Buddha	600:67	2607	12	36	47	26	107
Scene	1036:362	6654	17	67	41	18	113
A. Dragon	513:32	2992	8	28	51	24	287
Armadillo	632:134	4810	11	57	44	21	342

were performed to generate the image. The black, dark gray, and light gray colors depict the edge-collapse, fan-merge, and no-adapt operations, respectively. As can be seen from Fig. 8(a) the density of fine changes is high in close-to-viewer regions and low in far-from-viewer regions. The distribution of the fan-merge operations is the highest in regions nearby but not very close to the viewer (in a kind of a second layer) and fades out as regions become far from the viewer. It is important to notice that the vast majority of the adapt operations are performed nearby the viewer. Figure 8(b) shows the Armadillo model in wire-frame to depict the distribution of the resolution over the surface of the model. Figure 8(c) shows the surface of the selected level of detail. The images in Fig. 9 were obtained as the images in Fig. 8 for the Dragon model.

Figure 10(a) shows the image of a selected level of detail of the Terrain model. The three images in Fig. 10(b), Fig. 10(c), and Fig. 10(d) show three snapshots from the same geographical position during three different-speed flights over the

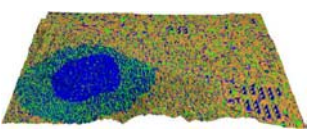
(a) Adaptive operations (b) Wire frame (c) Selected level of detail

Fig. 8. Armadillo model: original size 320K.

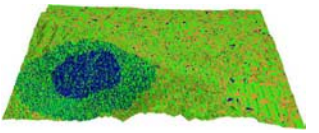
(a) Adaptive operations (b) Wire frame (c) Selected level of detail

Fig. 9. Dragon model: original size 202K.

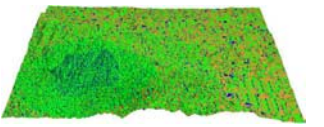
(a) Selected level of detail (92K)



(b) Slow camera motion



(c) Medium camera motion



(d) Fast camera motion

Fig. 10. Terrain model: original size 522K.

Terrain model using the same route. This route is a straight line that starts from the lower-left corner and ends at the middle of the model. The flight speed is measured by the number of frames at the same frame rates over the same route. The slow-speed flight takes 100 frames, the medium-speed flight takes 50 frame, and the fast-speed flight takes 10 frames. The images in Fig. 10(b), 10(c) and 10(d) show the distribution of the edge-collapse (black color), fan-merge (dark-gray color), and no-adapt operations (light-gray color). As can be seen in these images the number of edge-collapse operations decreases and the number of fan-merge operations increases as the speed of motion increases. Such an adaptive behavior reduces the adaptation time by increasing the number of fan-merge operations to quickly provide an appropriate level of detail for the changing view parameters. It is important to note that appearance of the boundaries between the fine and coarse operations is resulted from categorizing (and thus the same coloring) of any sequence of edge-collapses as an edge-collapse operator.

6. Conclusion

We have presented a framework for multiresolution hierarchy — fan-hierarchy — which is based on the fan-merge operation. The fan-hierarchy data structure encodes the various levels of detail to support real-time view-dependent rendering. The fan-merge operator provides a dual-paths adaptive operation, which enables fine as well as coarse changes on the selected level of detail. The selection of the appropriate change rate is chosen based on view parameters and the difference between current and target levels of detail. We have shown that the fan-hierarchy is more compact than previous multiresolution hierarchies. In addition, it has achieved a respected improvement in adapt time and the overall frame rates. Furthermore, it accelerates the rendering process by sending the selected level of detail as a collection of fans with no extra overhead.

Our fan-hierarchy could be plugged into other multiresolution schemes including the cluster-based hierarchies to improve the rendering performance and image quality.

Acknowledgments

Our research and particularly this work is supported by the Lynn and William Frankel Center for Computer Sciences, the Israel Ministry of Science, and Technology and Tuman fund. In addition, we would like to thank the reviewers for their constructive comments.

References

1. P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio and R. Scopigno, "BDAM — batched dynamic adaptive meshes for high performance terrain visualization," *Computer Graphics Forum* **22**(3), p. 505 (2003).
2. M. Duchaineau, M. Wolinsky, D. E. Sigeti, M. C. Miller, C. Aldrich and M. B. Mineev-Weinstein, "ROAMing terrain: Real-time optimally adapting meshes," *IEEE Visualization '97 Proceedings*, pp. 81–88 (1997).
3. M. H. Gross, R. Gatti and O. Staadt, "Fast multiresolution surface meshing," *IEEE Visualization '95 Proceedings*, pp. 135–142 (1995).
4. P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hodges, N. Faust and G. A. Turner, "Real-time continuous level of detail rendering of height fields," *SIGGRAPH '96 Proceedings*, pp. 109–118 (1996).
5. H. Hoppe, "Progressive meshes," *ACM Computer Graphics Proceedings, Annual Conference Series, (SIGGRAPH '96)*, pp. 99–108 (1996).
6. J. C. Xia, J. El-Sana and A. Varshney, "Adaptive real-time level-of-detail-based rendering for polygonal models," *IEEE Transactions on Visualization and Computer Graphics* **3**(2), p. 171 (1997).
7. H. Hoppe, "View-dependent refinement of progressive meshes," *ACM Computer Graphics Proceedings, Annual Conference Series, (SIGGRAPH '97)*, pp. 189–198 (1997).
8. D. Luebke and C. Erikson, "View-dependent simplification of arbitrary polygonal environments," *ACM Computer Graphics Proceedings, Annual Conference Series, (SIGGRAPH '97)*, pp. 199–207 (1997).
9. L. De Floriani, P. Magillo and E. Puppo, "Efficient implementation of multi-triangulations," *IEEE Visualization '98 Proceedings*, pp. 43–50 (1998).
10. A. Guéziec, G. Taubin, B. Horn and F. Lazarus, "A framework for streaming geometry in VRML," *IEEE Computer Graphics & Applications*, **19**(2), p. 68 (1999).
11. R. Klein, A. Schilling and W. Straßer, "Illumination dependent refinement of multiresolution meshes," *Proceedings of Computer Graphics International*, pp. 680–687 (1998).
12. R. Pajarola, "Fastmesh: efficient view-dependent meshing," *Proceedings of Pacific Graphics*, pp. 22–30 (2001).
13. J. Kim and S. Lee, "Truly selective refinement of progressive meshes," *Proceedings of Graphics Interface*, pp. 101–110 (2001).
14. J. El-Sana, E. Azanli and A. Varshney, "Skip strips: Maintaining triangle strips for view-dependent rendering," *IEEE Visualization '99 Proceedings*, pp. 131–138 (1999).

15. J. El-Sana, N. Sokolovsky and C. Silva, "Integerating occlusion culling with view-dependent rendering," *IEEE Visualization 2001 Proceedings*, pp. 371–378 (2001).
16. S. Yoon, B. Salomon and D. Manocha, "Interactive view-dependent rendering with conservative occlusion culling in complex environments," *IEEE Visualization '2003 Proceedings*, pp. 163–170 (2003).
17. J. El-Sana and A. Varshney, "Generalized view-dependent simplification," *Computer Graphics Forum* **18**(3), p. 83 (1999).
18. J. El-Sana and Y. Chiang, "External memory view-dependent simplification," *Computer Graphics Forum* **19**(3), p. 139 (2000).
19. C. DeCoro and R. Pajarola, "Xfastmesh: Fast view-dependent meshing from external memory," *IEEE Visualization 2002 Proceedings*, pp. 363–370 (2002).
20. A. Shamir, V. Pascucci and C. Bajaj, "Multi-resolution dynamic meshes with arbitrary deformation," *IEEE Visualization 2000 Proceedings*, pp. 423–430 (2000).
21. C. Erikson and D. Manocha, "Hierarchical levels of detail for fast display of large static and dynamic environments," *Proceedings of Interactive 3D Graphics*, pp. 111–120 (2001).
22. P. Cignoni, F. Ganovelli, E. Gobbetti, F. Marton, F. Ponchio and R. Scopigno, "Adaptive TetraPuzzles — efficient out-of-core construction and visualization of gigantic polygonal models," *ACM Transactions on Graphics* **23**(3), p. 796 (2004).
23. S. E. Yoon, B. Salomon, R. Gayle and D. Manocha, "Quick-VDR: Interactive view-dependent rendering of massive models," *IEEE Visualization 2004 Proceedings*, pp. 131–138 (2004).
24. M. Garland and P. S. Heckbert, "Surface simplification using quadric error metrics," *Computer Graphics Proceedings, Annual Conference Series (SIGGRAPH '97)*, pp. 209–216 (1997).

Yotam Livny is a PhD student of computer science at the Ben-Gurion University of the Negev, Israel. His research interest includes interactive rendering of 3D graphic models using acceleration tools, such as view-dependent rendering and dynamic data organization for optimal graphics hardware usage. Yotam received a BSc in mathematics and computer science from the Ben-Gurion University of the Negev, Israel in 2003. He is currently pressuring a PhD in computer science under the advisory of Dr. Jihad El-Sana.

Neta Sokolovsky is a PhD student of computer science at Ben-Gurion University of the Negev, Israel. Her research interests include 3D interactive graphics, occlusion culling, view-dependent rendering, image processing, and video visualization. Neta received a BSc and MSc in computer science from Ben-Gurion University of the Negev, Israel in 1999 and 2001, respectively. She is pursuing her PhD in computer science under the advisory of Dr. Jihad El-Sana.

Jihad El-Sana is a senior lecturer of computer science at Ben-Gurion University of the Negev, Israel. El-Sana's research interests include 3D interactive graphics, multiresolution hierarchies, geometric modeling, computational geometry, virtual environments, and distributed and scientific visualization. His research focus on polygonal simplification, occlusion culling, accelerating rendering, remote/distributed visualization, and exploring the applications of virtual reality in engineering, science, and medicine. El-Sana received a BSc and MSc in computer science from Ben-Gurion University of the Negev, Israel in 1991 and 1993. He received a PhD in computer science from the State University of New York at Stony Brook in 1999. El-Sana has published over 20 papers in international conferences and journals. He is a member of the ACM, Eurographics, and IEEE.