A Graphical Representation of the State Spaces of Hierarchical Level-of-Detail Scene Descriptions

Ashton E.W. Mason and Edwin H. Blake, *Member*, *IEEE Computer Society*

Abstract—We present a new graphical representation of the level-of-detail state spaces generated by hierarchical geometric scene descriptions with multiple levels of detail. These level-of-detail graphs permit the analytical investigation of the hierarchical level-of-detail optimization problem that arises for such descriptions. As an example of their use, we prove the equivalence of two hierarchical level-of-detail algorithms.

Index Terms—Level of detail, hierarchical, scene description, graph, state space, optimization, approximation.

1 Introduction

Level-of-detail rendering techniques aim to manage intelligently the complexity of rendered scenes at render time in order to regulate frame rates while maximizing the perceptual benefit of the rendering to the user. Successful methods are typically based on the use of hierarchical scene descriptions that combine the elegance of a recursive hierarchical spatial or geometric decomposition of the scene with the ability to provide simplified drawable representations for groups of related subobjects. The use of such techniques for active frame rate control has created the need for intelligent hierarchically aware level-of-detail optimization algorithms whose task is to select at render-time between the multiple drawable scene representations that may be extracted from hierarchical scene descriptions. Their aim in this selection is the maximization of the perceptual benefit of the resulting rendering, subject to a constraint on its total rendering cost, so as to maintain regular frame rates while making the best use of available resources. Although several such algorithms have been proposed, there has been little in the way of robust analytical investigation of the hierarchical level-of-detail optimization problem. In this paper, we present a semantic aid to this investigation in the form of the *level-of-detail graph*, a graphical representation of the level-of-detail state spaces generated by hierarchical scene descriptions in which any number of simplified representations may be provided for groups of related objects. We demonstrate by example in Section 6 how these graphs have enabled us to gain a better understanding of the meaning of shared object representations and to derive new analytical results concerning level-of-detail optimization algorithms.

• *The authors are with the Department of Computer Science, University of Cape Town, Private Bag, Rondebosch, 7701 South Africa.*
E-mail: ashton_mason@hotmail.com, edwin@cs.uct.ac.za.

Manuscript received 23 Nov. 1998; revised 4 Apr. 2000; accepted 20 May 2000.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number 109331.

We begin in Section 2 with a review of related work. In Section 3, we define a generalized hierarchical level-of-detail scene description that will serve as the basis for the development of level-of-detail graphs and algorithms for their generation in Section 5. In this development, we make use of a transformation, described in Section 4, of the hierarchical description to an equivalent constrained non-hierarchical one. This transformation was originally proposed in [6]. In Section 6, we provide an example of the use of level-of-detail graphs in the form of a proof of equivalence for two hierarchical level-of-detail algorithms. Finally, we make some concluding remarks in Section 7.

2 Background

The first use of level-of-detail in rendering, suggested by Clark [3], was hierarchical in nature, making use of a recursive decomposition of the scene in order to associate simplified representations with groups of related subobjects. Most subsequent approaches, such as those of Blake [1], Chamberlain et al. [2], and Shade et al. [7], have been extensions of the same basic idea. Funkhouser and Séquin [4] were among the first to note that *predictive* level-of-detail selection could be used for active frame rate control. The approach of Maciel and Shirley [5], as well as that of Mason and Blake [6], represent, broadly, extensions of the predictive, but essentially nonhierarchical optimization approach of Funkhouser and Séquin to hierarchical level-of-detail scene descriptions.

The *level-of-detail optimization problem* is the task of selecting, for each frame, the scene representation that provides the maximum perceptual benefit (however it is approximately measured) for a limited rendering cost. While Funkhouser and Séquin note that the predictive level-of-detail optimization problem is equivalent to the Multiple Choice Knapsack Problem (MCKP) [4], Mason and Blake demonstrate in [6] that this equivalence is broken by the shared representations for groups of objects that characterize truly hierarchical level-of-detail scene

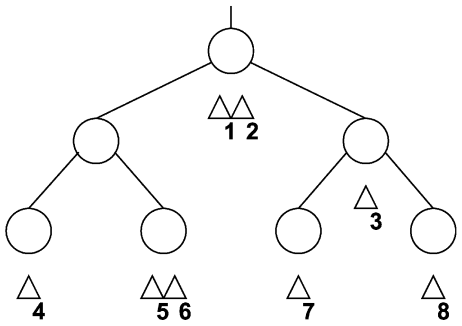

Fig. 1. A simple level-of-detail hierarchy. Objects are represented by circles and their impostors by triangles. The impostors of each object are shown in order of increasing detail from left to right. Impostors are labeled arbitrarily for convenience.

descriptions and that the hierarchical level-of-detail optimization problem is equivalent to a more complex *hierarchical generalization* of the MCKP. Otherwise, the hierarchical level-of-detail optimization problem has remained largely unanalyzed. In this paper, we provide a formal, intuitive, and graphical description technique for the state spaces generated by hierarchical level-of-detail descriptions as an aid to this investigation.

3 Generalized Hierarchical Level-of-Detail Description

Here, we define a generalized hierarchical level-of-detail scene description which will serve as the basis for the following sections. An *object* is defined recursively as the union of other smaller objects which are its *parts*, or children. The hierarchy of objects forms a part-whole decomposition of the scene from a single *scene object* at the root to the smallest, indivisable components at the leaves (see Fig. 1). Each leaf object has a number of associated *impostors*, or drawable representations.[1] In addition, *group* (or nonleaf) objects may also each be provided with their own set of impostors. An impostor of a group object serves as a unified drawable representation of all the parts of the group. In this way, each object has as its drawable representations not only its own explicitly associated impostors at various levels of detail, but also the multiple more detailed combinations of the impostors of its descendents. Together these representations constitute the available levels of detail of that object.

Definition 1 (Level of Detail). *A* level-of-detail s *of an object* O *is a set of impostors* $\{i_1, i_2, i_3, \ldots, i_n\}$ *such that exactly one of the impostors on the path from* O *to each of the leaves of the subtree rooted at* O *is an element of* s.

For example, the valid levels of detail of the scene object in Fig. 1 are $\{1\}$, $\{2\}$, $\{4,5,3\}$, $\{4,6,3\}$, $\{4,5,7,8\}$, and $\{4,6,7,8\}$. Each constitutes a complete and unambiguous representation of the scene.

Definition 2 (Replacement Set). *The* replacement set *of an impostor belonging to an object* O *is the immediately higher detail impostor of* O, *if one exists, or the set of the lowest detail impostors of the nearest impostor-bearing descendents of* O, *otherwise.*

In Fig. 1, the replacement sets of impostors 1, 2, 3, and 5 are $\{2\}$, $\{4,5,3\}$, $\{7,8\}$, and $\{6\}$, respectively. Impostors 4, 6, 7, and 8 have no replacement sets.

An *incrementation* of a level-of-detail s of an object O is the replacement of some impostor $i \in s$ by its replacement set r to produce another level-of-detail s'. Conversely, a *decrementation* of s is the replacement of some complete replacement set $r \subset s$ by the impostor whose replacement set is r. The levels of detail of each object are partially ordered by the following relation:

Definition 3 (Partial Ordering of Levels of Detail). *Two levels of detail* s *and* t *of an object* O *are related by* $s \leq t$ *if there exist levels of detail* $l_1, l_2, l_3, \ldots, l_n$ *such that* $l_1 = s$, $l_n = t$, *and* l_{i+1} *is the result of some incrementation of* l_i *for all* $i \in \{1, 2, 3, \ldots, n-1\}$.

We say that s is a *lower* level of detail than t and that t is a *higher* level of detail than s. If $s \leq t$ and $s \neq t$, then we say that s is a *strictly lower* level of detail of O than t, denoted $s < t$. For example $\{2\} < \{4,5,3\} < \{4,6,3\}$ in Fig. 1. If $s \leq t$ and $s \cap t = \emptyset$, then we say that s is *uniformly lower* than t. The *lowest* and *highest* levels of detail of an object are those such that there exist no other levels of detail that are lower and higher, respectively.

Definition 4 (Covering of Replacement Sets). *We say that a replacement set* r *is* covered *by a level-of-detail* s *of an object* O *if there exists a level-of-detail* t *of* O *such that* $t \leq s$ *and* $r \subset t$.

This implies that, in some sense, s contains a representation of the part of the scene represented by r that is at least as highly detailed as r.

Definition 5 (Ancestor Replacement Sets). *A replacement set* r *is an* ancestor *replacement set of another replacement set* q *if there exists a (possibly trivial) list of replacement sets* $r_1, r_2, r_3, \ldots, r_n$ *such that* $r_1 = r$, $r_n = q$, *and* r_{i+1} *is the replacement set of some impostor in* r_i *for* $i \in \{1, 2, 3, \ldots, n-1\}$.

Conversely, r is a *descendent* replacement set of q if q is an ancestor replacement set of r. In Fig. 1, $\{3,4,5\}$ is a descendent replacement set of $\{2\}$ and an ancestor replacement set of $\{6\}$ and $\{7,8\}$.

4 Constrained Nonhierarchical Description

In this section, we provide a transformation of the hierarchical level-of-detail description defined in Section 3 to an equivalent *constrained* nonhierarchical one.

Note that the impostors of group objects are equivalent to shared low detail impostors of the children, or parts, of those group objects. The shared impostor must either be selected for all of the parts at once or for none at all. By repeatedly transforming group object impostors to inherited shared impostors of the children of those group objects, we can create an equivalent *constrained nonhierarchical level-of-detail description*, as shown in Fig. 2. Each object in this new

1. We use *impostor* in its most general sense, referring to any drawable object representation [5].

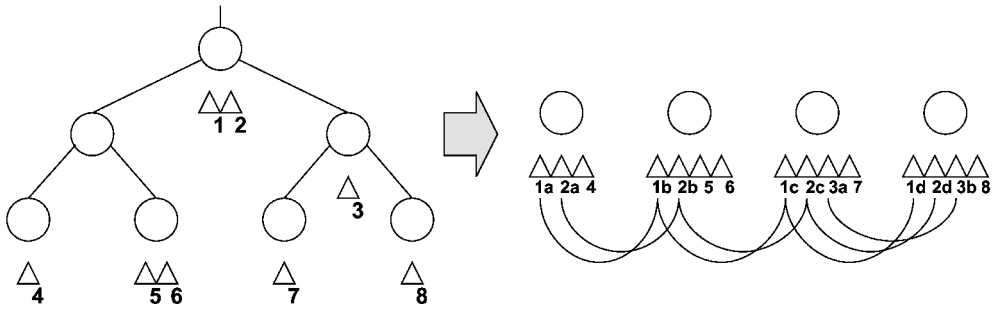

Fig. 2. Transformation of a level-of-detail hierarchy to an equivalent constrained non-hierarchical description. Constraints, shown as links, indicate that the inherited shared impostors must be selected in unison. The shared impostors are labeled with letters to distinguish them from one another.

description corresponds to a leaf object in the original hierarchy and has as its impostors all those that lay on the path from the root object to itself, in top-down order. Notice, for example, that impostor 1 in Fig. 2, being an impostor of the scene object, is implicitly a representation of every leaf object. The hierarchical set of constraints preserves the original structure by requiring that each set of inherited impostors is always selected in unison. A valid level-of-detail of such a constrained description is a set of impostors such that all constraints are satisfied and exactly *one* impostor is selected for each object. Note that if an impostor of an object O is constrained by a constraint A, then the immediately lower detail impostor of O must be constrained by some other constraint B. If an impostor of another object P is also constrained by A, then the immediately lower detail impostor of P must be constrained by B.

5 Level of Detail Graphs

Every valid level-of-detail of a level-of-detail description gives rise to a different rendering of the scene. Together, these levels of detail and the ordering relationships between them form a *state space*. Our aim is to provide a simple conceptual representation of this state space.

A *level-of-detail graph* consists of a set of nodes, a set of arcs connecting those nodes, and a partial ordering on the nodes. Each node corresponds to a level-of-detail. It is connected by arcs to all of the other nodes whose corresponding levels of detail may be reached from that one by means of a single incrementation or decrementation. The partial ordering $\leq$ that was defined for levels of detail in Definition 3 is also applied to the nodes of the associated level-of-detail graph. Any two nodes s and t such that $s < t$ are always represented in the graph such that s is lower (in some spatial dimension) than t.

Fig. 3 shows some example nonhierarchical level-of-detail descriptions and the level-of-detail graphs that they generate. In these descriptions, the replacement set of an impostor is always simply the immediately higher impostor of the same object, if one exists. The level-of-detail graphs generated by such nonhierarchical descriptions are all regular lattices in n dimensions, where n is the number of objects in the scene. The number of nodes on each side of the lattice corresponds to the number of impostors of each object, respectively. Notice that the arcs on opposite sides of any square in the lattice correspond to the selection of the same replacement set (or, in this case, impostor). Every path between the same two levels of detail involves the same series of replacements, although the ordering of the series is unique to each path. Notice too that every level-of-detail is reachable from every other level-of-detail by some series of incrementations and decrementations.

The level-of-detail graphs of hierarchical level-of-detail descriptions differ from those of nonhierarchical descriptions in that they are not in general regular n-dimensional lattices. Recall from Section 4 that any given hierarchical level-of-detail description may be transformed to an equivalent *constrained* nonhierarchical one, where the constraints serve to preserve the hierarchical replacement set structure. The effect of introducing a single constraint is illustrated in Fig. 4. A constraint removes all the states that contain some, but not all, of the impostors that it constrains. Any arcs incident to a removed state are also removed. New arcs are created from each of the states containing all of the constrained impostors to the states that are identical except for the replacement of the constrained shared impostors by their replacement set. We refer to this as the *Constraint Algorithm*.

Typical hierarchical level-of-detail descriptions are equivalent to constrained nonhierarchical descriptions with more than one constraint, such as that in Fig. 2. The level-of-detail graph of any constrained nonhierarchical description that is equivalent to a valid hierarchical one may be generated by beginning with the graph of the unconstrained nonhierarchical description and applying the Constraint Algorithm for each constraint in turn, in *increasing order of*

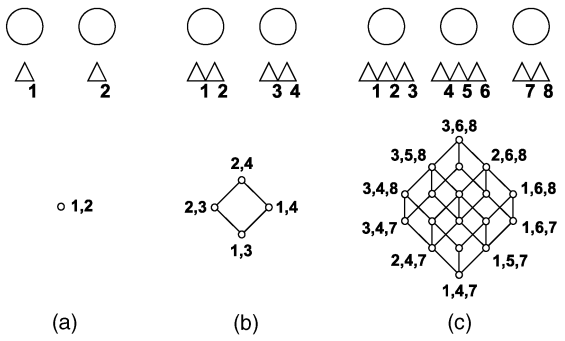

Fig. 3. Level of detail graphs of nonhierarchical descriptions. Three simple nonhierarchical level-of-detail descriptions, numbered (a) to (c), and their corresponding level-of-detail graphs. Some nodes are unlabeled in (c) for clarity.

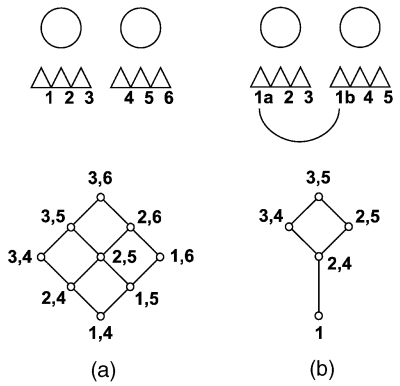

Fig. 4. Effects of the addition of a single constraint. The unconstrained description and its level-of-detail graph are shown in (a). The constrained description and its level-of-detail graph are shown in (b).

detail of the impostors they constrain. Fig. 5 shows an example hierarchical level-of-detail description, its equivalent constrained nonhierarchical description, and the generation of its corresponding level-of-detail graph.[2] Notice that the final graph accurately represents the valid levels of detail of the hierarchical description and the partial ordering of incrementations and decrementations defined on them. Fig. 6 shows the level-of-detail graph of the hierarchy in Fig. 1.

6 Proof of Equivalence of Two Algorithms

As an example of the application of level-of-detail graphs, we present a proof of the equivalence of two algorithms. These algorithms are corrected versions of those presented in [6]. They are hierarchical generalizations and improvements of those presented by Funkhouser and Séquin [4] and a slightly altered nonhierarchical specialization of this proof would serve as a proof of the previously unproven equivalence of the Funkhouser-Séquin algorithms.

Algorithm A is a greedy approximation algorithm for the hierarchical level-of-detail optimization problem. Algorithm B is an equivalent incremental version designed to exploit frame-to-frame coherence. Both attempt to find the level-of-detail with maximum (predicted) total perceptual benefit, subject to an upper limit on total rendering cost, given numerical predictions of the perceptual benefit and rendering cost of each impostor in the current frame. Whereas A always begins with the lowest level-of-detail, B begins with the solution found for the previous frame on the assumption that successive solutions are likely to exhibit significant coherence.

Algorithm A iteratively considers currently selected impostors for replacement with their replacement sets. In each iteration, the impostor considered is that whose replacement set r has greatest *relative value*[3] $\mathrm{RV}(r)$. The incrementation is performed if it can be afforded without exceeding the available frame rendering time, otherwise the algorithm terminates.

Algorithm B differs in that, in each iteration, it both increments and optionally repeatedly decrements the selected level of detail. The impostor selected for replacement in each incrementation step is that whose replacement can be afforded and whose replacement set has greatest relative value. In the same iteration, the algorithm repeatedly decrements the selected level-of-detail until the total cost of the selected level-of-detail is reduced to satisfy the rendering cost limit. Each decrementation replaces a currently completely selected replacement set with its associated impostor. The replacement set selected for replacement in each step is that with *lowest* relative value. The algorithm terminates when, upon completion of an iteration, it finds that the replacement selected in the incrementation step of that iteration was subsequently deselected in one of the decrementation steps.

Algorithms A and B are equivalent as long as replacement sets always have greater total perceptual benefit and rendering cost than the impostors they replace and ancestor replacement sets always have *greater relative value* than their descendents (which implies that more detailed renderings provide increased perceptual benefit with diminishing returns). We therefore assume that to be the case in this proof. We prove the equivalence by denoting the level of detail corresponding to A's solution by g (for the "greedy" solution) and considering the actions of B in each of its possible current states. The states of B, corresponding to selected levels of detail, are partitioned into four classes in terms of their relation to g. For any level of detail s, it is true that either $s = g$, $s < g$, $s > g$, or s and g are not related (which we denote for convenience by $s \neq g$). This is shown by means of shading in Fig. 7, which illustrates the proof for a simple example.

Considering incrementations, we show that, whatever its current state t, B will always choose an incrementation selecting a replacement set covered by g, if one is available, over any that select replacement sets not covered by g. The only interesting case is that in which t and g are not related since if $t < g$, then *all* possible incrementations select replacement sets covered by g, and if $t = g$ or $t > g$, then all incrementations select replacement sets not covered by g. In the interesting case where t and g are not related, there may exist at least one incrementation selecting a replacement set i covered by g and at least one selecting a replacement set j not covered by g. Then, there must exist a level of detail t' from which A chose an incrementation selecting i over one selecting another replacement set j' that is an ancestor[4] of j since i is covered by g. Therefore, $\mathrm{RV}(i) \geq \mathrm{RV}(j')$. By assumption, $\mathrm{RV}(j') > \mathrm{RV}(j)$, hence

2. An unfortunate feature of this process is that the dimensionality of the initial graph is the same as the number of leaf objects in the hierarchy. Graphs of dimension higher than three, although perfectly well-defined, are difficult to visualize. Nevertheless, their sense is clear from the examples of lower dimensionality that may be visualized. Moreover, they may be generated automatically and their topology may be inspected by means of queries.

3. The *relative value* of the replacement set r of an impostor i is defined as the ratio of the difference in perceptual benefit and the difference in rendering cost between r and i, i.e., $\mathrm{RV}(r) = \frac{(\sum_{j \in r} \mathrm{benefit}_j) - \mathrm{benefit}_i}{(\sum_{j \in r} \mathrm{cost}_j) - \mathrm{cost}_i}$.

4. That is, there exists a list of replacement sets $r_1, r_2, r_3, \ldots, r_n$ such that $r_1 = j'$, $r_n = j$, and r_{i+1} are the replacement set of some impostor in r_i for $i \in \{1, 2, 3, \ldots, n-1\}$.

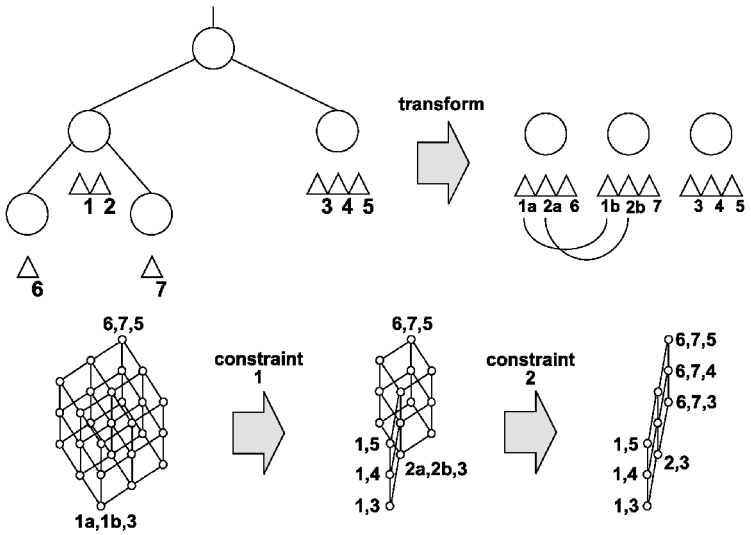

Fig. 5. Generation of a level-of-detail graph. A hierarchical level-of-detail description, its equivalent constrained nonhierarchical description, and the generation of its level-of-detail graph. We begin with the graph of the unconstrained nonhierarchical description and apply the constraints 1 and 2 in that order.

$\mathrm{RV}(i) > \mathrm{RV}(j)$. Therefore, B will choose the incrementation selecting i over that selecting j.

Considering decrementations, we show that, whatever its current state u, B will always choose a decrementation deselecting a replacement set not covered by g, if one is available, over any that deselect replacement sets covered by g. Here, the only interesting cases are those in which u and g are not related and those in which $u > g$, but u is not uniformly higher than g since if u is uniformly higher than g, then *all* possible decrementations deselect replacement sets not covered by g, and if $u = g$ or $u < g$, then all decrementations deselect replacement sets covered by g. In the interesting cases, there may exist at least one decrementation deselecting a replacement set p not covered by g and at least one deselecting a replacement set q that is covered by g. Then, there must exist a level-of-detail u' from which A chose an incrementation selecting q over one selecting another replacement set p' that is an ancestor of p since q is covered by g. Therefore, $\mathrm{RV}(q) \geq \mathrm{RV}(p')$. By assumption, $\mathrm{RV}(p') > \mathrm{RV}(p)$, hence $\mathrm{RV}(q) > \mathrm{RV}(p)$. Therefore, B will choose the decrementation deselecting p over that deselecting q.

Table 1 shows the actions of B in a given iteration for any possible current state. Recall that, in each iteration, B increments once and then decrements repeatedly, while the total rendering cost exceeds the limit. By inspection of the table, it can be seen that B must eventually halt in state $s = g$. If B begins in state $s = g$, then it increments once, but then immediately decrements back to g since the replacement set selected by the incrementation is then the only set not covered by g and, hence, will be deselected by the decrementation. If B begins in state $s > g$, then it increments once and then decrements repeatedly until it reaches $s = g$. In each decrementation step, a replacement set not covered by g will be available until the current state becomes g, whereupon the algorithm will have deselected the replacement set selected in the single incrementation. If the initial state is $s < g$, then the algorithm increments once, selecting a replacement set covered by g rather than one not covered by g, and the resultant state is either $s = g$ or $s < g$, and the algorithm iterates again. Finally, if the initial state s is not related to g, then the algorithm will increment once and then possibly decrement several times. The incrementation selects a replacement set covered by g and each decrementation, if

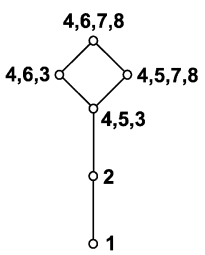

Fig. 6. Another example. The level-of-detail graph of the hierarchy in Fig. 1.

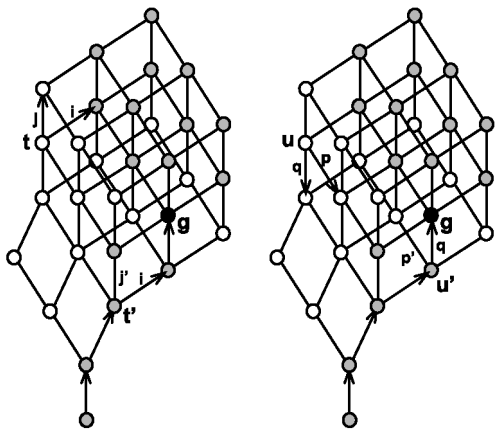

Fig. 7. Illustration of the proof. Levels of detail $s < g$ and $s > g$ are shaded. Arrows indicate the path taken by algorithm A in reaching its solution g. Arrows from t and u show incrementations and decrementations available at those levels of detail. Some arcs are labeled with the replacement sets whose selection or deselection they represent. Recall that arcs on opposite sides of any square represent the selection or deselection of the same replacement set.

TABLE 1
Actions of Algorithm B

state	incr.	selected	decr.	deselected	new state	halt?
$s = g$	1	not covered	1	not covered	$s = g$	yes
$s > g$	1	not covered	> 1	not covered	$s = g$	yes
$s < g$	1	covered	0	–	$s < g$ or $s = g$	no
$s \neq g$	1	covered	≥ 0	not covered	$s \neq g$, $s < g$ or $s = g$	no

Columns show the current state s, the number of incrementations performed in this iteration, whether the replacement sets selected are covered by g, the number of decrementations, whether the replacement sets deselected are covered by g, the new state, and whether the algorithm halts in this iteration.

any, deselects a replacement set not covered by g and the resultant state is $s < g$ or $s = g$ or $s \neq g$. In any event, the algorithm iterates again. Note that it is impossible for the algorithm to remain in state $s < g$ or $s \neq g$ indefinately since the incrementations and decrementations performed serve to reduce the distance between s and g at each step (measured as the minimum total number of incrementations and decrementations by which g may be reached from s). Therefore, B's solution tends to and must eventually reach A's solution, and the two algorithms are equivalent.

7 Conclusion

We have presented a new graphical representation of the state spaces generated by hierarchical level-of-detail scene descriptions. These *level-of-detail graphs* allow the simulation and analysis of hierarchical (and nonhierarchical) level-of-detail optimization algorithms and serve as a conceptual tool for the exploration of level-of-detail state spaces. We have described algorithms for the generation of the level-of-detail graphs of a well-defined class of hierarchical level-of-detail scene descriptions. As an example of the use of level-of-detail graphs, we have provided a proof of the equivalence of two hierarchical level-of-detail optimization algorithms.

Acknowledgments

The work for this paper was conducted while Ashton E.W. Mason was a student at the University of Cape Town.

References

[1] E.H. Blake, "Complexity in Natural Scenes: A Viewer Centered Metric for Computing Adaptive Detail," PhD thesis, Queen Mary College, London Univ., 1989.
[2] B.L. Chamberlain, T. DeRose, D. Lischinski, D. Salesin, and J. Snyder, "Fast Rendering of Complex Environments Using a Spatial Hierarchy," *Proc. Graphics Interface '96,* 1996.
[3] J.H. Clark, "Hierarchical Geometric Models for Visible Surface Algorithms," *Comm. ACM,* vol. 19, no. 10, pp. 547-554, Oct. 1976.
[4] T.A. Funkhouser, C.H. Séquin, "Adaptive Display Algorithm for Interactive Frame Rates during Visualization of Complex Virtual Environments," *SIGGRAPH '93, Computer Graphics Proc., Ann. Conf. Series,* pp. 247-254, Aug. 1993.
[5] P.W.C. Maciel and P. Shirley, "Visual Navigation of Large Environments Using Textured Clusters," *Proc. 1995 Symp. Interactive 3D Graphics,* pp. 95-102, Apr. 1995.
[6] A.E.W. Mason and E.H. Blake, "Automatic Hierarchical Level of Detail Optimization in Computer Animation," *Computer Graphics Forum,* D. Fellner and L. Szirmay-Kalos, eds., vol. 16, pp. 191-199, 1997.
[7] J. Shade, D. Lischinski, D.H. Salesin, T. DeRose, and J. Snyder, "Hierarchical Image Caching for Accelerated Walkthroughs of Complex Environments," *SIGGRAPH '96, Computer Graphics Proc., Ann. Conf. Series,* pp. 75-82, Aug. 1996.

Ashton E.W. Mason received his BSc (hons) degree in computer science in 1994 and his PhD degree in 2000, both from the University of Cape Town. He is currently employed as a software engineer at 3Dlabs UK. His interests lie in level-of-detail optimization, realtime rendering, 3D modeling, and animation.

Edwin H. Blake heads the Collaborative Visual Computing Laboratory of the Department of Computer Science at the University of Cape Town. His research interests are evaluation of collaborative virtual environments, data visualization, and viewer centered graphical algorithms, as well as IT policy for development. He is a member of the CSSA, SAICSIT, IEEE Computer Society, and ACM.