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

Ashton E. W. Mason and Edwin H. Blake

CS98-09-01

Collaborative Visual Computing Laboratory
Department of Computer Science
University of Cape Town
Private Bag, RONDEBOSCH
7701 South Africa
e-mail: amason@cs.uct.ac.za, edwin@cs.uct.ac.za

Abstract

We present a new method for representating the state spaces of hierarchical level of detail descriptions, or scene descriptions with multiple hierarchical levels of detail. This representation, called a *level of detail graph*, permits the investigation and exploration of the state spaces of non-hierarchical and hierarchical level of detail optimization algorithms. We present algorithms for generating the level of detail graph representations of arbitrary level of detail descriptions. As an example of the use of level of detail graph representations we demonstrate the equivalence of two published level of detail optimization algorithms whose equivalence was previously stated without proof.

1 Introduction

Level of detail rendering permits dynamic and automatic control over the amount and nature of detail rendered in an interactive visualization. Several optimization algorithms have been proposed that provide approximate solutions to the *level of detail optimization problem*: selecting in each frame that scene representation which provides the greatest perceptual benefit for a certain limited total rendering cost [3] [4] [6].

Hierarchical scene descriptions at multiple levels of detail have proved useful in allowing groups of objects to share drawable representations. These descriptions are characterised by the provision of multiple shared representations for groups of objects. Some level of detail optimization algorithms allow the use of hierarchical level of detail descriptions [1] [4] [6] [7].

In this paper we introduce *level of detail graphs*, a new semantic representation of the state spaces generated by hierarchical (and non-hierarchical) level of detail descriptions. We have found them to be useful in the analysis of level of detail optimization algorithms, and they have allowed us to derive new results concerning such algorithms, as illustrated in Section 7. We provide algorithms for the generation of the level of detail graphs of arbitrary level of detail descriptions.

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 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 described in [6]. In Section 6 we review the operation of the incremental and non-incremental level of detail optimization algorithms of Funkhouser and Séquin and in Section 7 we use level of detail graphs to prove their equivalence. Finally we make some concluding remarks in Section 8.

2 Background

The *level of detail optimization problem* has evolved from its beginnings when Clark [2] suggested the use of multiple drawable representations, or impostors [4], for scene objects at varying *levels of detail*. Funkhouser and Séquin [3] provided the first *predictive* level of detail optimization algorithm in which the complexity of the rendered scene representation is predictively regulated from one frame to the next. Their scheme however suffers from a limitation in that it is inherently non-hierarchical and does not allow for the provision of shared representations for groups of objects, due to its foundations in a greedy algorithm for the Multiple Choice Knapsack Problem. Typical non-predictive optimization schemes, such as those of [2] and [1], make use of *hierarchical level of detail descriptions* that are characterised by the provision of multiple shared drawable representations for groups of related objects. Maciel and Shirley [4] as well as Mason and Blake [6] provide predictive algorithms that allow the use of hierarchical level of detail descriptions, effectively combining the hierarchical level of detail approach originally proposed by Clark with the predictive approach of Funkhouser and Séquin.

Thus far, few attempts have been made to formally investigate the *hierarchical level*

of detail optimization problem: the problem of predictively selecting for each frame of a realtime visualization system the subset of a hierarchical level of detail description that provides the greatest *perceptual benefit* for a limited total rendering cost (the available frame rendering time). In this paper we provide a formal, intuitive description technique for the state spaces generated by hierarchical (and, inclusively, non-hierarchical) level of detail descriptions.

3 Generalized Hierarchical Level of Detail Descriptions

Here we define a generalized hierarchical level of detail description which will be used as the basis for the following sections. We define an *object* recursively as a collection of smaller objects that are its *parts*, or children. Each object may be provided with a finite set of associated *impostors*, or drawable representations of that object. A hierarchical description consists at the highest level of a single object, called the *scene object*. Figure 1 shows a simple level of detail hierarchy. The impostors of each object are ordered by increasing detail, so that higher impostors have greater perceptual benefit and greater rendering cost. Objects in general have *explicit* representations (their own impostors) and *implicit* representations (those of their descendents). The explicit and implicit representations of an object together constitute the possible *levels of detail* of that object. Each level of detail corresponds to a unique set of selected impostors:

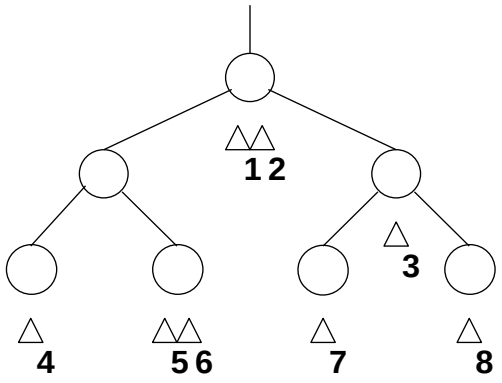

Figure 1: **A simple level of detail hierarchy.** Objects are represented by circles and their impostors by triangles, ordered by increasing detail. The replacement set of impostor 1 is $\{2\}$, and that of impostor 2 is $\{4, 5, 3\}$. The replacement set of impostor 3 is $\{7, 8\}$ and that of 5 is $\{6\}$. Impostors 4, 6, 7 and 8 have no replacement sets.

Definition 1 *Level of Detail*

A level of detail s *of an object O is a set of impostors* $\{i_1, i_2, i_3, \ldots, i_n\}$*. The impostors* $i_1, i_2, i_3, \ldots, i_n$ *are selected 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 the subset.*

We define the *replacement set* of an impostor to refer to the set of impostors that constitute the immediately higher level of detail of the object that owns that impostor:

Definition 2 *Replacement Set*
The replacement set *of an impostor belonging to an object O is:*

1. *The immediately higher detail impostor of O, if one exists.*
2. *The set of the lowest detail impostors of the nearest impostor-bearing descendents of O, otherwise.*

Figure 1 illustrates examples of various replacement sets in a typical level of detail hierarchy. All impostors have exactly one replacement set, except for the highest detail impostors of the leaves of the hierarchy, which have none. Conversely every impostor is an element of exactly one replacement set. The impostors which together constitute the lowest level of detail of the object are assumed to be the replacement set of an imaginary impostor corresponding to "no representation".

We define an *incrementation* of a level of detail s of an object O to be the replacement of some impostor $i \in s$ by its replacement set R. Conversely a decrementation of s is the replacement of some complete replacement set $R \subset s$ by the impostor whose replacement set is R. In general a level of detail s may be incremented and decremented in many different ways, where each corresponds to the replacement of a different impostor or replacement set in s.

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 of an object O, s and t, 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 $i \in \{1, 2, 3, \ldots, n-1\}$.

If $s \leq t$ and $s \neq t$ then we say that s is a *strictly lower* level of detail of O than t. The *lowest* and *highest* levels of detail of an object are those such that there exist no others that are strictly lower and strictly higher. Even if a level of detail s of an object O is strictly lower than another t, s and t may still share some impostors in common. If they do not, then we say that s is *uniformly lower* than t:

Definition 4 *Uniformly Lower Levels of Detail*
A level of detail s of an object O is uniformly lower *than another level of detail t of O if $s \leq t$ and $s \cap t = \emptyset$.*

Apart from the partial ordering of levels of detail, there is a sense in which a level of detail may contain a higher or lower representation of a particular object than a given replacement set. In order to talk about this ordering we define the *covering* of a replacement set by a level of detail:

Definition 5 *Covering of Replacement Sets*
If R is the replacement set of some impostor of an object O then R is covered *by a level of detail s of an ancestor M of O if there exists a level of detail t of M such that $t \leq s$ and t contains R.*

More simply, R is covered by s if there exists a lower level of detail t containing R. In that case, s can be reached by a series of incrementations from t. We speak also of the covering of impostors, when those impostors are in themselves complete replacement sets.

4 Constrained Non-Hierarchical Description

In this section we provide a transformation of the generalized hierarchical level of detail description defined in Section 3 to an equivalent *constrained* non-hierarchical one. We will make use of this transformation in Section 5.

The impostors of group objects in the hierarchical representation are equivalent to single shared low detail impostors for each of the children of those group objects (Figure 2), with the constraint that the children must take on those shared impostors together. We may therefore transform any given hierarchy by replacing the highest detail impostor of a group object with a shared lowest-detail impostor of the children of the group object.

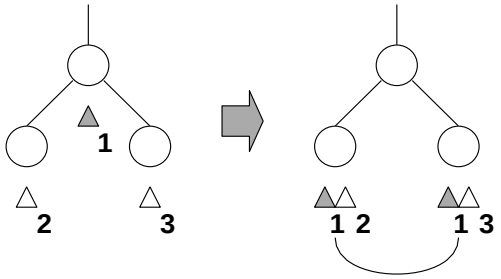

Figure 2: **Transformation of a group object impostor.** A group object impostor may be transformed to an equivalent shared impostor of its children. The inherited group impostor is shaded. The link indicates that the shared impostors must be selected in unison.

By applying this transform repeatedly we can create an "empty" hierarchy with impostors only at the leaves. The leaf objects then form a constrained non-hierarchical description, as shown in Figure 3. Each object has as its impostors the impostors of itself and all of its ancestors in the original hierarchy, in top-down order. This equivalence between the hierarchical and non-hierarchical descriptions is subject to a set of *constraints*: the objects in the non-hierarchical description that share each inherited group impostor *must take on that shared impostor in unison*. Each constraint in the non-hierarchical description corresponds to exactly one group impostor in the equivalent hierarchical description, and constrains the inherited impostors to which the group impostor is equivalent. A constraint is characterised by the set of (shared) impostors that it constrains.

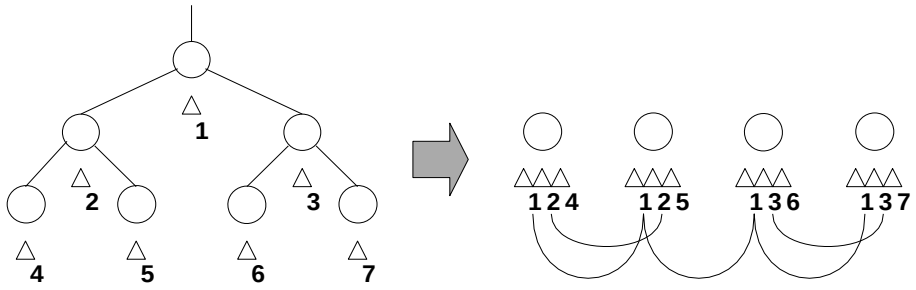

Figure 3: **Transformation of a level of detail hierarchy.** The transformation of a simple level of detail hierarchy to its equivalent constrained non-hierarchical description. For simplicity each object is assumed to have exactly one impostor.

5 Level of Detail Graphs

Every configuration, or level of detail, of a level of detail description gives rise to a different rendering of the scene. Together these levels of detail form a state space that is a complete description of the possible configurations of the description. Our aim in this section is to formalize the relationship between different levels of detail, and to provide a simple conceptual representation of that relationship.

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 or state. 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 (Definition 3) is also applied to the nodes of the associated level of detail graph. We require that any two distinct nodes s and t such that s is strictly lower than t are always represented in the graph such that s is lower (visually) than t.

5.1 Non-Hierarchical Level of Detail Descriptions

We first consider the level of detail graphs generated by non-hierarchical level of detail descriptions. 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 non-hierarchical 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, and the total number of nodes is the product of the numbers of impostors of all objects. Figure 4 shows some example non-hierarchical level of detail descriptions and the level of detail graphs that they generate.

Notice that arcs on opposite sides of the same square in the lattice correspond to the selection (or deselection, in the case of decrementation) of the same replacement set. Any path between the same two nodes involves the same series of replacements, although the ordering of the series is unique to that path.

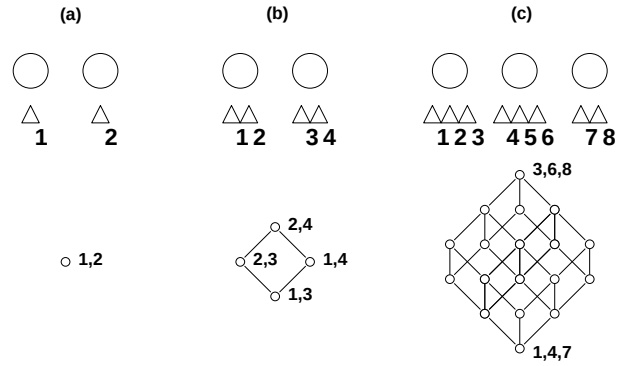

Figure 4: **Level of detail graphs of non-hierarchical descriptions.** Three simple non-hierarchical level of detail descriptions, numbered (a) to (c), and their corresponding level of detail graphs. Some nodes are unlabed in (c) for clarity.

5.2 Hierarchical Level of Detail Descriptions

We now consider the level of detail graphs of hierarchical level of detail descriptions. These level of detail graphs differ from those of non-hierarchical descriptions in that they are not regular n-dimensional lattices. Recall from Section 4 that any given hierarchical level of detail description may be transformed to an equivalent *constrained* non-hierarchical one. The constraints in the constrained non-hierarchical description serve to limit the possible levels of detail of that description.

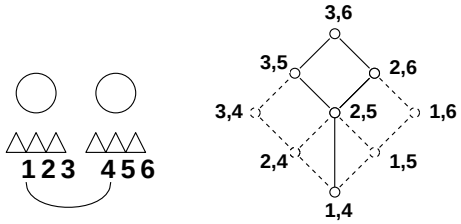

Figure 5: **Effects of a single constraint.** The effects of the application of a single constraint (on impostors 1 and 4) to a non-hierarchical level of detail description on the level of detail graph of the description. Removed states and arcs are shown with dotted lines. Note the addition of a new arc between state 1,4 and state 2,5.

5.2.1 Single Constraint

The effect of introducing a single constraint is to change the level of detail graph, as shown in Figure 5. A constraint removes all the states that contain only some, but not all, of the impostors that it constrains. Any arcs incident to a removed state are also removed. The states that remain are those that contain none or all of the constrained impostors. 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 shared impostor by its replacement set (that is, the immediately higher impostors of the linked objects in the constrained non-hierarchical description). The algorithm that applies the effects of a constraint on a level of detail graph is given in Figure 6.

begin
  for each level of detail s in G

    // if s contains some but not all of the impostors in C then
    // remove s from the level of detail graph

  if $C \cap s \neq \emptyset$ and $C - s \neq \emptyset$ **then**
    remove s and all arcs incident to it from G

    // if s contains all the impostors in C then create an arc from s to t,
    // the level of detail reached from s by replacing C

  if $C \subseteq s$ **then**
    let $R \leftarrow$ replacement set of the group impostor corresponding to C
    let $t \leftarrow (s - C) \cup R$
    create a new arc from s to t in G
end

Figure 6: **The constraint algorithm.** The constraint algorithm takes as input a non-hierarchical level of detail description, its level of detail graph G, and a set of impostors C that are consrained by a new constraint, and produces as output the level of detail graph of the description after application of the constraint.

Figure 7 compares the effects of two single constraints applied to a simple non-hierarchical level of detail description. Figure 7 (a) shows the original unconstrained description and its level of detail graph. (b) shows the result of linking the lowest impostors of the three objects (impostors 1, 4 and 7) by a single constraint, and (c) shows the result of constraining the lowest impostors of only the first two objects (impostors 1 and 4).

5.2.2 Multiple Constraints

Typical hierarchical level of detail descriptions are equivalent to constrained non-hierarchical descriptions with more than one constraint, such as that in Figure 3. There are nonetheless certain requirements that are satisfied by any constrained non-hierarchical description that is equivalent to a valid hierarchical description: Each impostor may be constrained by at most one constraint, and each constraint may constrain at most one impostor of each object. Secondly, if an impostor of an object is constrained then all of the lower detail impostors of that object must also be constrained (since if an impostor is an inherited group

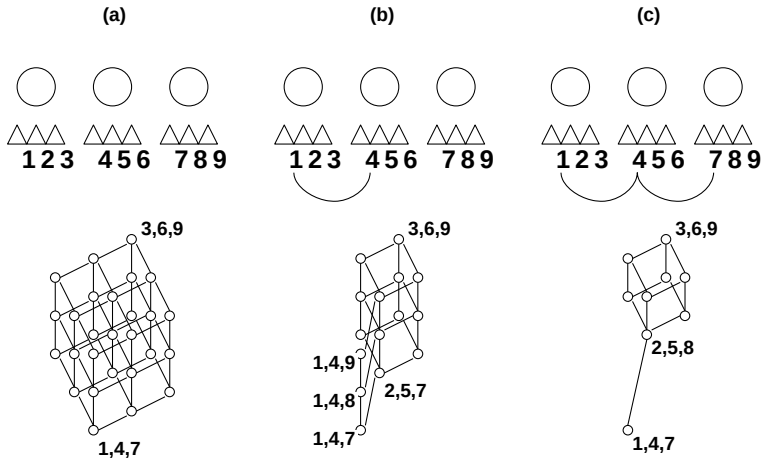

Figure 7: **Comparison of the effects of two single constraints.** The effects of two single constraints applied to a simple non-hierarchical level of detail description. The unconstrained non-hierarchical description and its regular 3-dimensional level of detail graph are shown in (a). In (b) the lowest impostors of the first two objects are constrained, and in (c) the lowest impostors of all three objects.

impostor then all lower impostors of that object are also inherited group impostors). Finally, if C and D are constraints on consecutive impostors of an object then C and D must constrain consecutive impostors on any object constrained by D, in the same order.

The level of detail graph of any constrained non-hierarchical description that is equivalent to a valid hierarchical one may be generated by beginning with the graph of the unconstrained non-hierarchical description and applying the Constraint Algorithm of Figure 6 for each constraint in turn, in increasing order of detail of the impostors they constrain. Figure 8 shows an example hierarchical level of detail description, its equivalent constrained non-hierarchical description, and the generation of its corresponding level of detail graph.

6 Funkhouser and Séquin Level of Detail Optimization Algorithm

In this section we describe the incremental and non-incremental versions of the Funkhouser and Séquin level of detail optimization algorithm [3]. We will prove the equivalence of these algorithms in Section 7.

Funkhouser and Séquin [3] showed that the non-hierarchical level of detail optimization problem is equivalent to the Multiple Choice Knapsack Problem [5], in which an optimal subset of a set of *candidate items* must be selected with a certain maximum cost. The items each have constant *cost* (w_{ij}) and *profit* (p_{ij}) and are partitioned into k *candidate subsets*. Exactly one item must be selected from each subset. The items in this case correspond to the drawable object representations, or *impostors*, and the candidate

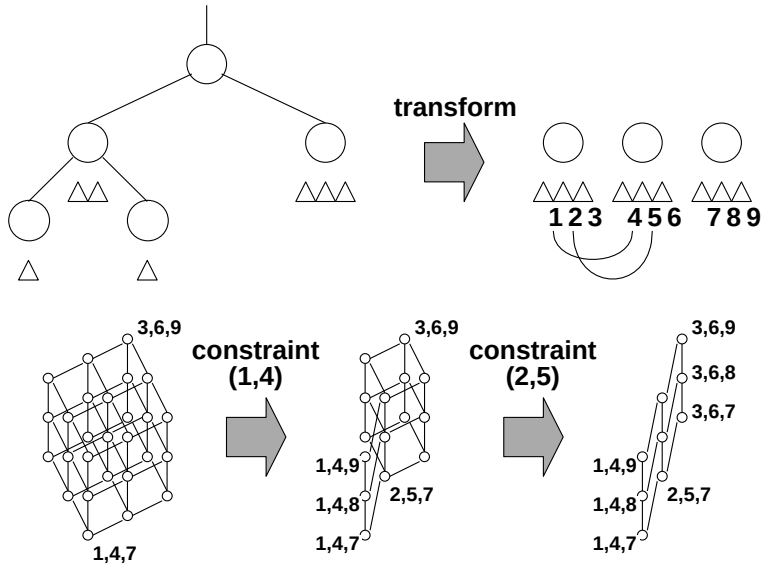

Figure 8: **Level of detail graph of a hierarchical description.** A hierarchical level of detail description, its equivalent constrained non-hierarchical description, and the generation of its level of detail graph. We begin with the graph of the unconstrained non-hierarchical description and apply the constraints $\{1, 4\}$ and $\{2, 5\}$ in that order.

subsets correspond to the objects themselves. Exactly one representation must be selected for each object.

Funkhouser and Séquin formulate *benefit* and *cost* heuristics that predict the "contribution to scene perception", or profit, and rendering cost, or cost, of object representations.

The non-incremental version of the Funkhouser and Séquin algorithm, shown in Figure 9, is simply a greedy algorithm for the MCKP. It considers object impostors (or items) in descending order of value, accepting each if it can be afforded and rejecting it if it cannot. When an impostor is considered that is a representation of an object already represented by another impostor already in the knapsack, it replaces the selected impostor only if it has higher profit.

The incremental version of the algorithm is equivalent, as long as the values of object impostors decrease monotonically with increasing cost. We make this assumption in our proof in Section 7. The algorithm, shown in Figure 10, begins with an initial best-guess solution which is typically the approximate solution reached for the previous frame. It then iteratively improves this selected level of detail by incrementing and decrementing until it reaches its final solution. In each iteration the algorithm increments once (regardless of its current state) and then decrements repeatedly while the total cost of its selected level of detail is greater than the available rendering time. The impostor replaced in each incrementation is that whose immediately higher detail impostor has greatest value. Its "replacement set", with which it is replaced, is the immediately higher detail impostor. Conversely the "replacement set" replaced during each decrementation is the currently selected impostor whose value is lowest, and the impostor that replaces it is the immedi-

```
begin
    start with nothing selected
    order the set of impostors by decreasing value (profit/cost)
    for each impostor j
    begin
        if j is an impostor of the same object as a selected impostor i then
            if p_j > p_i and j can be afforded in place of i then
                select j and discard i
            else
                discard j
        else
            if j can be afforded then
                select j
            else
                discard j
    end
end
```

Figure 9: **The non-incremental Funkhouser and Séquin level of detail optimization algorithm.** The algorithm selects a subset of a set of impostors such that their total cost is below some set limit and exactly one impostor is selected for each object.

ately lower valued impostor of the same object. The algorithm terminates when the level of detail of any object is both incremented and decremented in the same iteration, or when no objects are available for further incrementation or decrementation (that is, the highest or lowest level of detail is selected).

The level of detail optimization problem can be viewed as a search problem; namely the traversal of the level of detail graph generated by the level of detail description in search of the optimal level of detail (for the entire scene) that provides the greatest total perceptual benefit while limiting the total rendering cost. The actions of the Funkhouser and Séquin algorithms are exactly that; the non-incremental algorithm begins with the lowest level of detail selected and always increments, while the incremental algorithm begins with an arbitrary level of detail and both increments and decrements. Because we assume that higher detail impostors of the same object have lower value, the selection of impostors in order of descending value in the non-incremental algorithm corresponds to a continuous traversal of the graph from lower to strictly higher levels of detail.

The nodes of the level of detail graph are partitioned into two classes by the available rendering time: those whose cost is less than or equal to the limit and those whose cost is greater. Since it is assumed that higher levels of detail always have higher cost, we can imagine the level of detail graph being cut into two parts by an imaginary surface. Figure 11 shows the actions of the incremental and non-incremental algorithms with respect to this surface.

```
begin
    set L ← the LoD selected for the previous frame
    while not done
    begin

                    // increment the current LoD

        if L is not the highest LoD then
        begin
            increment L, removing the impostor i ∈ L whose
            replacement impostor j has greatest value, and inserting j
        end

                    // while the total cost is too high, decrement the current LoD

        while the total cost of L is greater than the available rendering time
        begin
            if L is not the lowest level of detail then
            begin
                decrement L, inserting the impostor f whose replacement
                impostor g ∈ L has lowest value, and removing g

                    // terminate, if the same impostor was both incremented
                    // and decremented

                if i = f then set done ← TRUE
            end
        end

                    // or if we can't increment or decrement any further

        if L is the highest LoD and its total cost is not too high
        or L is the lowest LoD and its total cost is too high then
            set done ← TRUE
    end
end
```

Figure 10: **The incremental version of the Funkhouser and Séquin level of detail optimization algorithm.** The *replacement impostor* of an impostor is the immediately higher detail impostor of the same object.

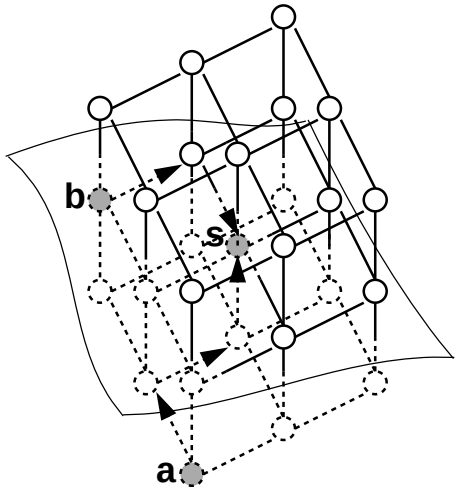

Figure 11: **Actions of the incremental and non-incremental algorithms.** The imaginary cutting surface between the levels of detail whose cost is greater than the available rendering time (dotted lines) and those whose cost is not (solid lines). The non-incremental Funkhouser and Séquin algorithm begins at the lowest level of detail (a) and increments until it reaches a final solution (s). The incremental algorithm begins at the previous frame's solution (b) and both increments and decrements to reach (s).

7 Equivalence of the Greedy and Incremental Algorithms

In this section we prove as an example of the use of level of detail graphs the equivalence of the incremental and non-incremental versions of the Funkhouser and Séquin level of detail optimization algorithm (Section 6). We use level of detail graphs (Section 5) to represent the state space of the non-hierarchical level of detail description that the algorithms assume, hence showing that both reach the same final state.

When the incremental algorithm is in a given state s and increments or decrements, there are generally multiple distinct incrementations and decrementations available, each of which replaces a different currently selected impostor. In general, some of the incrementations will select replacement impostors that are covered by the non-incremental algorithm's solution g but not by s, and others will select impostors that are covered by neither s nor g. Similarly decrementations will generally be available that deselect impostors that are covered by s but not by g and others that deselect impostors that are covered by both s and g. The central idea behind the proof is to show that the incrementations and decrementations chosen always serve to bring the selected state "closer" to g, by selecting whenever possible impostors that are covered by g and deselecting ones that are not.

7.1 Incrementation and Decrementation

In this section we prove two lemmas that together characterise the behaviour of the incremental algorithm with respect to the final solution state g of the non-incremental algo-

rithm, for all possible states. To this end we partition the set of states into four distinct subsets, or classes, according to their relation to g (See Figure 12). For any state s, exactly one of the following is true:

1. $s = g$
2. $s < g$ (s is strictly lower than g)
3. $s > g$ (s is strictly higher than g)
4. s and g are not comparable (we refer to this as $s \neq g$).

Lemma 1 *Whatever its current state, the incremental algorithm will always choose an incrementation selecting an impostor covered by g, if one is available, over any that select impostors not covered by g.*

Proof: (refer to Figure 12)

1. *In the case where the current state t of the incremental algorithm is in the class $s < g$, all possible incrementations select an impostor covered by g and the proof is immediate.*
2. *In classes $s = g$ and $s > g$ all incrementations select impostors covered by neither t nor g, and the proof is immediate.*
3. *In class $s \neq g$, there must exist at least one incrementation from t that selects an impostor i covered by g. Assume that at least one incrementation selects an impostor j not covered by g – otherwise the proof is immediate. There must exist a state t' from which the non-incremental algorithm chose an incrementation selecting i over one selecting a lower impostor j' of the object owning impostor j. Impostor i must therefore have greater value than j'. Since higher impostors of the same object have lower value, j' has greater value than j and so i has greater value than j. Therefore the incremental algorithm will choose the incrementation selecting i over that selecting j.*

Lemma 2 *Whatever its current state, the incremental algorithm will always choose a decrementation deselecting an impostor not covered by g, if one is available, over any that deselect impostors that are covered by g.*

Proof: (refer to Figure 12)

1. *In the case where the current state u of the algorithm is uniformly higher than g, any decrementation must deselect an impostor not covered by g, and the proof is immediate.*
2. *In the cases where u is in class $s < g$ or $s = g$, no decrementations exist that deselect impostors covered by g, so the proof is immediate.*

3. *In the case where u is either not comparable to g or strictly higher but not uniformly higher than g, there must exist at least one decrementation from u that deselects an impostor p not covered by g. Assume also that at least one decrementation from u deselects an impostor q that is covered by g – otherwise the proof is immediate. There must exist a state u' from which the non-incremental algorithm chose an incrementation selecting q over one selecting a lower impostor p' of the object owning impostor p. Impostor q must therefore have greater value than p'. Since higher impostors of the same object have lower value, p' has greater value than p, so q has greater value than p. Therefore the incremental algorithm will choose the decrementation deselecting p over that deselecting q.*

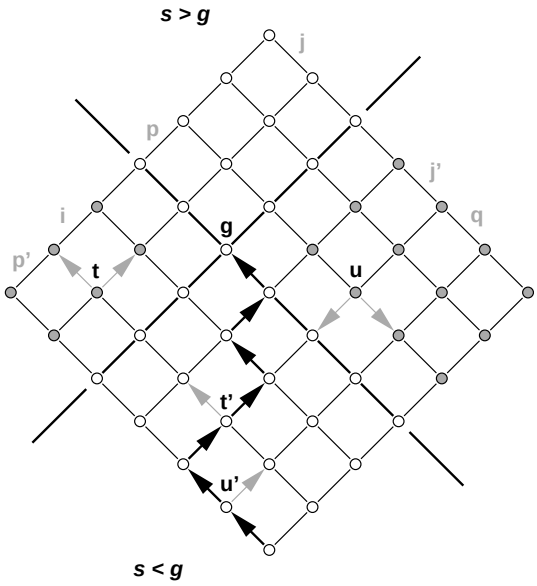

Figure 12: **Partitioning of states.** The partitioning of the set of states into four distinct classes: $s = g$, $s > g$, $s < g$ and $s \neq g$ (the shaded states). The dark arrows show, for this example, the path taken by the non-incremental algorithm in reaching state g. The light labels on rows of arcs name the impostors selected and deselected by all the incrementations and decrementations corresponding to the arcs in that row, and the light arrows to possible incrementations and decrementations mentioned in the text.

These results show that the incrementations and decrementations performed by the incremental algorithm serve, wherever possible, to bring the algorithm's current state closer to the final solution state of the non-incremental algorithm.

7.2 Actions of the Algorithm

Table 1 shows the actions of the incremental algorithm in any iteration for each state class. Firstly, recall that the algorithm always increments exactly once in each iteration (unless

it is already at the highest level of detail). The incrementations chosen always select an impostor that is covered by g and not by the current state, except in classes $s = g$ and $s > g$ where this is not possible. In these classes it increments once and then decrements until the total cost is less than or equal to the limit. The repeated decrementations must reach $s = g$, and then stop. The algorithm then terminates, having selected and deselected the same impostor in the same iteration.

class	incr.	selected	decr.	deselected	halt?	new class
$s = g$	1	not covered by g	1	not covered by g	yes	$s = g$
$s > g$	1	not covered by g	> 1	not covered by g	yes	$s = g$
$s < g$	1	covered by g	0	none	no	$s < g$, $s = g$
$s \neq g$	1	covered by g	≥ 0	not covered by g	no	$s \neq g$, $s < g$, $s = g$

Table 1: Table showing the actions of the incremental algorithm in any given iteration. Columns show the current state class, the number of incrementations performed, whether or not the impostor selected is covered by g, the number of decrementations performed, whether or not the impostor(s) deselected are covered by g, the class of the resulting state, and whether the algorithm terminates in this iteration.

In class $s < g$ the incremention selects an impostor covered by g. The state after incrementation is either $s = g$ or still $s < g$. Either way, the total cost of the state must be less than or equal to the limit so no decrementation is performed. The resulting state is therefore either $s = g$ or $s < g$.

In class $s \neq g$ the incremention also selects an impostor covered by g. The state after incrementation is either $s > g$ or still $s \neq g$. If $s > g$ then the rendering cost is greater than the limit and the algorithm must decrement until it is not. If $s \neq g$ then it may or not be greater than the limit, and so may or may not decrement. Any decrementations that are performed will deselect impostors that are not covered by g. The decrementation halts when the total cost is less than or equal to the limit. At this point the resultant state is either still $s \neq g$, $s < g$, or $s = g$.

The state graph of the algorithm, when collapsed into state classes, is shown in Figure 13. The algorithm moves from one state class to another, possibly sometimes staying in the same class from one iteration to the next. However since the algorithm's state always approaches g in every iteration, it cannot stay in the same class indefinately and must eventually move to class $s = g$ and terminate. Its final solution is therefore always g, the non-incremental algorithm's solution.

In this proof we have assumed that the incremental algorithm terminates by means of its most common terminating case: that it selects (by incrementation) and deselects (by decrementation) the same impostor in a single iteration. The other two terminating cases are special cases in which there is either too little available rendering time to render even the cheapest scene representation or enough to render even the most complex, and are unlikely to occur in practice. In the first special case the non-incremental algorithm selects "no representation" and the incremental algorithm selects the cheapest valid representation. In the second both algorithms select the most expensive representation.

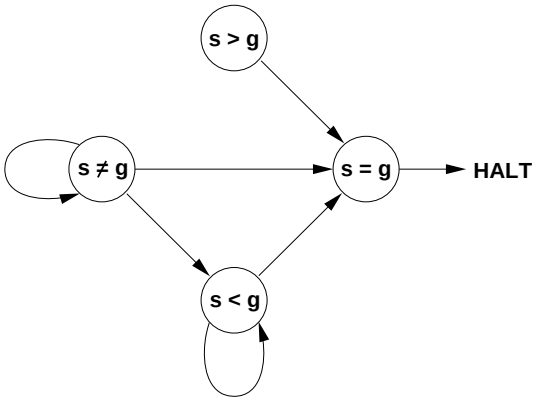

Figure 13: **Collapsed state diagram of the Funkhouser and Séquin incremental algorithm.** The four state classes (represented by circles) correspond to the four possible relationships that the algorithm's current state (or level of detail) s can have to the solution state of the non-incremental algorithm, g. In each iteration of the incremental algorithm it changes its current state, either moving from one state class to another or staying in the same class.

8 Conclusion

We have presented a new representation, called level of detail graphs, of the state spaces of general hierarchical and non-hierarchical level of detail scene descriptions. These level of detail graph representations allow the simulation and analysis of hierarchical and non-hierarchical level of detail optimization algorithms and serve as a conceptual tool for the exploration of level of detail state spaces. We have provided algorithms for the generation of the level of detail graphs of arbitrary hierarchical level of detail descriptions.

As an example of their use we have proved the previously unproved equivalence of two algorithms; the incremental and non-incremental versions of the Funkhouser and Séquin non-hierarchical level of detail optimization algorithm [3].

References

[1] B. L. Chamberlain, T. DeRose, D. Lischinski, D. Salesin, and J. Snyder. Fast rendering of complex environments using a spatial hierarchy. In *Graphics Interface '96*, 1996.

[2] J. H. Clark. Hierarchical geometric models for visible surface algorithms. *Communications of the ACM*, 19(10):547–554, October 1976.

[3] T. A. Funkhouser and C. H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *SIGGRAPH'93, Computer Graphics Proceedings, Annual Conference Series*, pages 247–254. ACM SIGGRAPH, August 1993.

[4] P. W. C. Maciel and P. Shirley. Visual navigation of large environments using textured clusters. In *1995 Symposium on Interactive 3D Graphics*, pages 95–102, April 1995.

[5] S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley and Sons Ltd., 1990.

[6] A. E. W. Mason and E. H. Blake. Automatic hierarchical level of detail optimization in computer animation. In D. Fellner and L. Szirmay-Kalos, editors, *Computer Graphics Forum*, volume 16. Eurographics, Blackwell Publishers, 1997.

[7] J. Shade, D. Lischinski, D. H. Salesin, T. DeRose, and J. Snyder. Hierarchical image caching for accelerated walkthroughs of complex environments. In *SIGGRAPH'96, Computer Graphics Proceedings, Annual Conference Series*, pages 75–82. ACM SIGGRAPH, August 1996.