EUROGRAPHICS '97 / D. Fellner and L. Szirmay-Kalos
(Guest Editors)

*Volume 16*, (*1997*), *Number 3*

# Automatic Hierarchical Level of Detail Optimization in Computer Animation

Ashton E. W. Mason and Edwin H. Blake [†]

Department of Computer Science, University of Cape Town, Cape Town, South Africa

**Abstract**

*We show that the hierarchical level of detail optimization problem is equivalent to a constrained version of the Multiple Choice Knapsack Problem, and present a new algorithm whose solution to it is at least half as good as the optimal one. The advantage of the hierarchical algorithm is that it allows the use of hierarchical level of detail descriptions in which shared representations may be provided for groups of objects. Rendering cost may then be saved to afford better renderings of more important objects, and the algorithm is capable of providing a complete representation of the visible scene even when the visible scene complexity is very high. Our algorithm has a worst case time complexity of $O(n \log n)$, and is incremental so that it typically completes in only a few iterations. We introduce the use of perceptual evaluation to demonstrate the effectiveness of the use of representations for groups of objects that our algorithm allows.*

## 1. Introduction

In order to guarantee bounded frame times during animation the dependency of the rendering complexity on the complexity of the visible scene must be limited. Level of detail or *multiresolution* techniques are one well established way of doing this (see [6] or [9] for a brief review). They allow automatic and dynamic selection of detail levels based on estimates of the rendering cost and perceptual benefit of object representations.

We propose a level of detail optimization algorithm that is a hybrid extension of those of Funkhouser and Séquin [5] and Maciel and Shirley [7]. We develop a transformation from a hierarchical level of detail description to a non-hierarchical one that allows us to apply the constrained optimization approach of [5] to a hierarchical description. The advantage is that groups of objects may then be replaced by single representations. This preserves the natural hierarchical description of scenes and saves additional rendering costs to allow better rendering of more important objects.

A contribution of this paper is to show that the hierarchical level of detail optimization problem is equivalent to a constrained version of the Multiple Choice Knapsack Problem (MCKP) [8] in which items may belong to more than one candidate subset. Our algorithm provides a solution to this constrained MCKP that is guaranteed to be at least half as good as the optimal one.

In Section 2 we review related work. In Section 3 we present the hierarchical level of detail description used by our algorithm and in Section 4 we present a description of the algorithm. In Section 5 we present a transformation of the hierarchical level of detail description to an equivalent constrained non-hierarchical one. In Section 6 we show that the level of detail optimization problem for this description is equivalent to a constrained version of the MCKP, present a simple greedy approximation algorithm for it, and show how our incremental algorithm relates to it. In Section 7 we discuss the algorithm with regard to optimality and efficiency. Section 8 describes an experiment conducted to test its effects, and Section 9 contains some concluding remarks and offers some directions for future work.

## 2. Background

The use of multiple drawable representations of scene objects at various levels of detail was suggested by Clark [4] to ensure appropriate detail levels for objects in a scene. Blake

[†] {amason|edwin}@cs.uct.ac.za

[1] provides metrics that predict the appropriate detail level of each object. Funkhouser and Séquin [5] apply the concept of *time critical computing* to the level of detail selection problem to limit the dependency of the rendering complexity on the complexity of the visible scene, and thereby to bound frame rendering times. The level of detail optimization problem then is to find the set of detail levels that together provide the best perception of the resulting frame, while ensuring that their total rendering cost is no higher than some predetermined limit.

Funkhouser and Séquin show that this level of detail optimization problem is equivalent to the Multiple Choice Knapsack Problem, in which an optimal subset of a set of items must be selected for a certain maximum cost [8]. The items each have a constant cost and profit and correspond in this case to the drawable representations, or *impostors*, of the scene objects. They are partitioned into subsets where each subset corresponds to the impostors of a single object. Only one item may be selected from each subset, since only one representation may 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. They use a simple greedy approximation algorithm for the MCKP in which items are placed into the knapsack in decreasing order of *Value* (Benefit/Cost), if they will fit. When another impostor is selected for an object that is already represented in the knapsack it replaces the existing one, if it will fit in its place. The Benefit of the later impostor is known to be higher than that of the earlier one because the impostors of each object are ordered by decreasing Value and increasing Benefit. They claim that this algorithm produces a solution to the MCKP that is guaranteed to be at least half as good as the optimal one. This is not strictly true [8], but it may be corrected by comparing the solution reached by the algorithm to the solution consisting of only the item of highest Benefit, and taking the better one. In any case, the cases where the algorithm’s solution is less than half optimal are not likely to occur in typical level of detail optimization problems.

Their iterative algorithm exploits the coherence between successive frames by optimizing the solution set incrementally from the selection for the previous frame. It is equivalent to the greedy MCKP algorithm as long as the Value of each object’s impostors decreases monotonically as their level of detail increases. In each iteration the level of detail of the object with the highest *subsequent* Value is incremented; then the level of detail of the object with the lowest *current* Value is decremented, as long as the total rendering cost is greater than the permitted maximum. The algorithm terminates when an object is both incremented and decremented in the same iteration, or when no objects are available for incrementation or decrementation.

This approach assumes a non-hierarchical level of detail description in which objects are distinct. This is unfortunate since most scenes are more naturally described by hierarchies, and because rendering cost must consequently be unavoidably invested in some representation of *every* object in the scene. Maciel and Shirley [7] propose the use of an hierarchical level of detail description with simpler representations for groups of objects. However, the level of detail optimization problem for such a hierarchy is not equivalent to the MCKP and thus cannot use the above greedy approximation algorithm. We shall show that the hierarchical level of detail optimization problem is equivalent instead to a *constrained* version of the MCKP.

Maciel and Shirley employ a heuristic that predicts the inherent importance of each object, independent of its representation. Their algorithm successively replaces the selected representation of the most important object in the current scene representation with the selected representations of its children, until the available rendering time is used up. The importance of a group object is defined as the maximum of the importances of its parts. This approach however does not take the Cost or Value of object impostors into account, and its solution to the constrained MCKP is not guaranteed to be at least half as good as the optimal one. We can see this by noting that it is possible for an object with many expensive parts, one of which has a high importance, to be selected for replacement over one with many cheap parts whose total importance is much greater.

Chamberlain *et al* [3] and Shade *et al* [9] propose hierarchical level of detail schemes in which simple impostor representations for groups of objects are generated automatically. They are substituted for their dependent scene geometry when they are considered accurate enough representations of it. While these schemes have the advantage of not requiring previously generated object impostors at multiple levels of detail, they place no upper bounds on the total complexity of the rendered scene.

## 3. Hierarchical Level of Detail Description

Our level of detail description is hierarchical and allows the use of impostors for group objects [7]. Objects are defined recursively as groupings of smaller objects, and each object may be supplied with a set of impostors that are its explicit drawable representations at increasing levels of detail. The leaves of the hierarchy must each have at least one impostor, so that the scene is always completely represented.

The *level of detail* (LOD) of an object is its currently selected representation. Objects in general have explicit and implicit representations, and all objects have levels of detail. An object is *explicitly represented* if its currently selected representation is one of its impostors, and *implicitly represented* if it is represented by the explicit representations of its descendants. The explicit representations or impostors of an object are ordered by increasing rendering complexity, and are defined as lower LODs than any of its implicit representations. The LOD of an object may be *incremented* to a higher LOD and *decremented* to a lower LOD. We shall speak also of the incrementation and decrementation of objects, by which we mean the incrementation and decrementation of their LODs.

The *representation tree* of an object describes its currently selected representation. It is the smallest subtree rooted at that object whose leaves are all explicitly represented. The representation tree of the root object is called the *scene representation tree,* and describes the selected representation of the whole scene. Figure 1 shows a level of detail hierarchy and one of its possible scene representation trees.

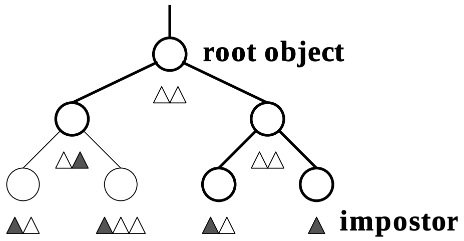

**Figure 1:** *A simple level of detail hierarchy and one of its possible representation trees.* Objects are represented by circles, and their impostors by triangles. The impostors of each object are shown from left to right in order of increasing detail. The selected impostor of each explicitly represented object is shaded and the corresponding scene representation tree is shown by the thick lines.

*Benefit* and *Cost* heuristics are defined for explicitly represented objects, and predict the “contribution to scene perception” and rendering cost of each object’s representation. A heuristic *Value* is defined as the ratio of the Benefit of an object to its Cost. Our optimization algorithm requires that the Value of an explicitly represented object should decrease as its level of detail increases, and also that the Value of an object at any of its explicit representations should be greater than the Values of all of its descendants at their explicit representations. In other words, there should be diminishing returns for greater levels of detail. The algorithm also requires that the Benefit and Cost of an explicitly represented object should be lower than the total Benefit and total Cost of any of its implicit representations. In other words, the impostors of group objects should be reduced detail representations of those objects.

## 4. Hierarchical Optimization Algorithm

Our incremental hierarchical optimization algorithm is applied once per frame with a specified maximum total rendering cost, and the output is a detail level for each of the objects in the scene. The algorithm proceeds iteratively until a terminating condition is met. In each iteration the LOD of the root object is incremented; then it is decremented while the total rendering cost exceeds the prescribed maximum. Note that the LOD of an object may in general be incremented or decremented recursively in many possible ways, each of which terminates in the incrementation or decrementation of some leaf of its representation tree. The heart of the algorithm lies in the selection of the particular incrementations and decrementations performed, which is aimed at maximizing the total Benefit of the scene representation while ensuring that its total Cost is within the required maximum.

Each possible incrementation of the root object must terminate in the incrementation of a *terminal* object that is either:

1. A leaf of the scene representation tree that is incremented from one explicit LOD to another. Figure 2 shows the result of one such incrementation of the hierarchy in Figure 1.

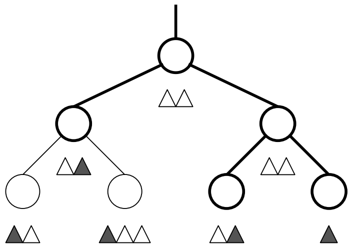

**Figure 2:** *The hierarchy of Figure 1, after the incrementation of a leaf of the scene representation tree from one explicit LOD to another.* The terminal object that is incremented is the third object in the third row. The scene representation tree is shown by the thick lines.

2. A leaf of the scene representation tree that becomes a non-leaf when it is incremented from its highest explicit to its lowest implicit LOD. Figure 3 shows the result of one such incrementation of the hierarchy in Figure 1.

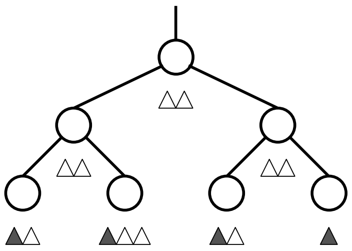

**Figure 3:** *The hierarchy of Figure 1, after the incrementation of a leaf of the scene representation tree that becomes a nonleaf after incrementation from its highest explicit to its lowest implicit LOD.* The terminal object that is incremented is the first object in the second row. The subsequent scene representation tree is shown by the thick lines.

We select the incrementation of the root object that results in the incrementation of the terminal object whose *subsequent* representation tree's highest Valued leaf has the *greatest* Value.

Similarly, each possible decrementation of the root object terminates in the decrementation of a terminal object that is either:

1. An object that is a leaf of the scene representation tree and is decremented from one explicit LOD to another. One such decrementation of the hierarchy in Figure 2 results in the hierarchy of Figure 1.
2. An object that becomes a leaf of the scene representation tree when it is decremented from its lowest implicit to its highest explicit LOD. For this to be possible, all of the children of that object must be at their lowest LODs. One such decrementation of the hierarchy in Figure 3 results in the hierarchy of Figure 1.

We select the decrementation of the root object that results in the decrementation of the terminal object whose *current* representation tree's highest Valued leaf has the *lowest* Value.

These incrementation and decrementation operations are implemented recursively. We store with each object the highest and lowest of the determining Values of the available leaves of its representation tree, and these are used in choosing between the children of objects at each level. They must be updated in post-order whenever their associated objects are incremented or decremented. The algorithm begins with an initialization step in which the heuristics and stored values of all objects are pre-calculated in a bottom-up fashion.

The algorithm terminates when any of the following occur:

1. The total rendering cost of the scene is less than or equal to the permitted maximum and the root object may not be incremented any further $\rightarrow$ there is sufficient time to render the entire scene at maximum detail.
2. The total rendering cost of the scene is greater than the permitted maximum and the root object may not be decremented any further $\rightarrow$ there is insufficient time to render the entire scene, even at minimum detail.
3. The LOD of a leaf of the scene representation tree is both incremented and decremented in the same iteration $\rightarrow$ the algorithm has reached its optimal solution, since the object that is incremented is immediately decremented.

After termination of the algorithm we render the selected representations of the leaves of the scene representation tree.

## 5. Transformation to Constrained Non-Hierarchical Description

In this section we present a transformation of the hierarchical level of detail description described in Section 3 to an equivalent constrained non-hierarchical one.

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 4), with the constraint that the children must take on those shared impostors together. This presents a way to transform a given hierarchical description to an equivalent constrained non-hierarchical one. We can create an "empty" hierarchy with impostors only at the leaves by pushing the impostors of group objects recursively down the hierarchy. These leaf objects then form a non-hierarchical description (Figure 5). Each object has as its impostors the impostors of itself and all of its ancestors in the original hierarchy, in the order in which they are enumerated. The heuristics of objects in the non-hierarchical description are the same as the heuristics of the original objects. This equivalence between the hierarchical and non-hierarchical descriptions is subject to the important constraint: objects in the non-hierarchical description that share impostors inherited from the same group objects *must take on those impostors in unison*.

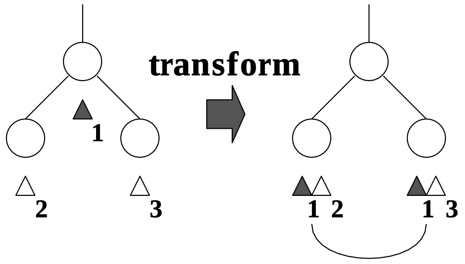

**Figure 4:** *Transforming an impostor of a group object to an equivalent shared impostor of its children.* For clarity, each object is assumed to have exactly one impostor, impostors are numbered, and the inherited impostor of the group object is shaded. The link attached to the shared impostor indicates that the objects which share it must take it on in unison.

Note that the impostors of each object in the non-hierarchical description are ordered by decreasing Value, and that the total Cost and Benefit of the immediately higher impostors of the group of objects that share each inherited group impostor are higher than the Cost and Benefit of the shared impostor itself. These orderings are required by our approximation algorithm.

## 6. Constrained Multiple Choice Knapsack Problem

We can now show that the level of detail optimization problem for the hierarchical description is equivalent to a constrained version of the Multiple Choice Knapsack Problem, shown schematically in Figure 6. The candidate items correspond to the impostors of the objects in the equivalent non-hierarchical description, and are divided into subsets such that each subset corresponds to the impostors of a single object. At most one item may be selected from each subset. The problem differs from the usual MCKP because

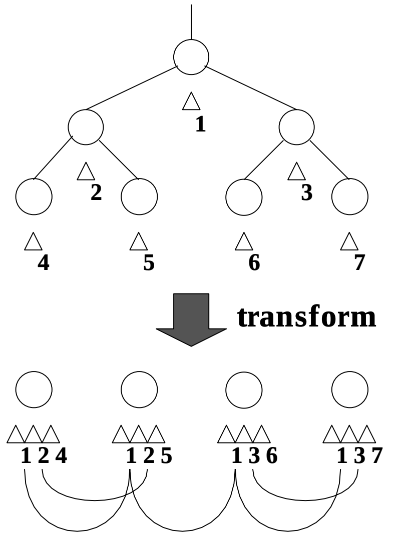

**Figure 5:** *Transforming a simple level of detail hierarchy to its equivalent non-hierarchical description.* The impostors of group objects have all been transformed into shared impostors of the leaf objects. Each object is assumed to have exactly one impostor. The constraints between shared impostors, shown as links, imply that the objects which share those impostors must take them on in unison.

of the constraints implied by the original hierarchical description. Some items are members of more than one subset: those which correspond to shared impostors in the non-hierarchical description. These represent more than one object and their corresponding items are capable of replacing items from several subsets at a time.

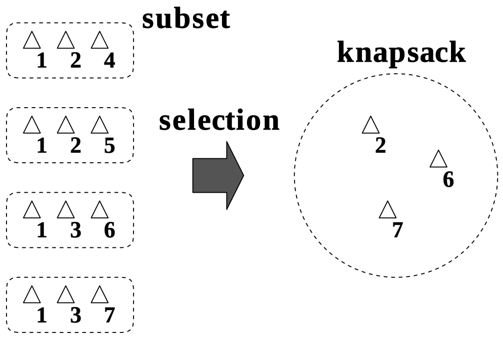

**Figure 6:** *A schematic representation of the constrained Multiple Choice Knapsack Problem.* The items correspond to the impostors of objects in the non-hierarchical level of detail description. They are divided into subsets corresponding to the objects to which the impostors belong, and items that correspond to shared impostors are members of more than one subset. Only one item may be selected from each subset.

Our algorithm and that of Maciel and Shirley provide approximate solutions to this NP-complete constrained MCKP, since they employ similar hierarchical descriptions. Their approach however differs significantly from ours. We use a simple greedy approximation algorithm whose solution to the constrained MCKP is guaranteed to be at least half as good as the optimal solution, if we also compare its solution to that consisting of only the item with highest Benefit, and take whichever is better. This greedy algorithm is an extension of the greedy approximation algorithm for the usual MCKP used by Funkhouser and Séquin to allow it to observe the hierarchical constraints which imply that some items belong to more than one subset.

Items are considered in order of decreasing Value, and placed into the knapsack if they will fit. When an item *i* is selected that belongs to a subset to which an item *j* already in the knapsack also belongs, the algorithm scans forward to find the *replacement set R* of the highest Valued remaining items including *i* that together belong to all the subsets to which *j* belongs. If this set *R* will fit in the knapsack in place of *j* it is substituted in its place, otherwise it is discarded. This ensures that items that belong to more than one subset are replaced by items that belong to the same subsets, so that the completeness of the representation is preserved. To ensure that the scene is completely represented, we initially place into the knapsack an imaginary item with no Cost or Benefit that belongs to all subsets.

The hierarchical version of our algorithm described in Section 4 is equivalent to this greedy algorithm for the MCKP, but is formulated incrementally.

As was noted in Section 4, each possible incrementation of the root object in the hierarchical description must terminate in the incrementation of some *terminal* object. The incrementation of such a terminal object corresponds to the incrementation of each of the objects that share its currently selected impostor in the equivalent non-hierarchical description. The incrementation of those objects corresponds in turn to the selection of the replacement set corresponding to their subsequently selected impostors, in the greedy algorithm. The suitability of each of the terminal objects for incrementation in the hierarchy is determined by the Value of the highest Valued leaf of its subsequent representation tree. Of the corresponding objects in the non-hierarchical description, it is the object with the highest subsequent Value that dictates the likelihood of the selection of the replacement set corresponding to their subsequently selected impostors. It is that object whose corresponding item is considered first. In the hierarchical description we therefore select the incrementation of the root object that results in the incrementation of the terminal object whose *subsequent* representation tree's highest Valued leaf has the *greatest* Value.

Similarly each possible decrementation of the root object terminates in the decrementation of some terminal object (Section 4). We select the decrementation that results in the decrementation of the terminal object whose *current* representation tree's highest Valued leaf has the *lowest* Value.

The decrementation of a terminal object corresponds to the “de-selection” of the replacement set corresponding to the currently selected impostors of the corresponding objects in the equivalent non-hierarchical description. The likelihood of the selection of the replacement set is determined by its highest Valued item.

## 7. Discussion of Algorithm

We have shown how a hierarchical level of detail description can be transformed to an equivalent non-hierarchical one in order to prove the equivalence of the hierarchical level of detail optimization problem to the constrained version of the MCKP, and to show the equivalence of our incremental algorithm to the greedy algorithm for the constrained MCKP. In practice our incremental algorithm operates directly on the hierarchical description, and the hierarchy is exploited to accelerate the algorithm through the storage of intermediate values.

Our incremental hierarchical optimization algorithm is an extension of the non-hierarchical algorithm of Funkhouser and Séquin to allow it to take into account the constraints on object LODs implied by the hierarchy. It takes advantage of frame-to-frame coherence by improving the set of selected items incrementally from the set selected for the previous frame, and is equivalent to the greedy approximation algorithm for the constrained MCKP described in Section 6, as long as the Value of each object’s LODs (both implicit and explicit) decreases monotonically.

The principle advantage of the algorithm over that of Funkhouser and Séquin is that it affords the use of simple representations for groups of objects. This allows the algorithm to produce scene representations in which all visible objects are completely represented at some level of detail, even when the complexity of the visible scene is very high. Groups of object representations may be replaced with cheaper shared group representations to an arbitrary degree (as long as available group impostors exist) without compromising the completeness of the scene representation. In addition, the freedom to select very low detail representations for unimportant group objects allows the algorithm to save additional rendering time in many situations that may be used to render better representations of more important objects.

Since the hierarchical algorithm is equivalent to the greedy algorithm for the constrained MCKP, its solution is guaranteed to be at least half as good as the optimal one. In this regard it is better than the algorithm of Maciel and Shirley, whose solution has no such guarantee.

The time complexity of the hierarchical optimization algorithm is $O(n\log n)$ in the worst case, like that of Funkhouser and Séquin. The number of iterations performed by the algorithm is $O(n)$, where $n$ is the number of objects. In each iteration the level of detail of the root object is incremented once and optionally decremented several times. The number of decrementations performed in each iteration is $O(1)$. Each recursive incrementation and decrementation, involving a pre-order selection step and a post-order update step, is $O(\log n)$. The iterative portion of the algorithm is preceded by an initialization stage in which the heuristics and stored values of all objects are pre-calculated, which is $O(n)$. The time complexity of the entire algorithm is therefore $O(n\log n)$. However it is also incremental and takes advantage of the coherence between successive frames to produce its solution in typically only a few iterations. The algorithm of Maciel and Shirley is $O(n)$, but is not incremental and requires a full optimization for every frame.

The algorithm is *predictive* [5] in that it bases its level of detail selection on predictions of the rendering cost and perception of object representations, and is therefore better able to provide constant rendering times than are those of Chamberlain *et al* [3] and Shade *et al* [9]. The total predicted rendering time of the selected scene representation is always guaranteed to be less than or equal to the permitted maximum.

The algorithm is subject to two limitations inherited from that of Funkhouser and Séquin: it is dependent on the existence of pre-generated impostors of all objects at varying levels of detail, and it assumes that the heuristics of objects are independent of viewing direction. The Cost and Benefit of the impostors of each object must always increase and their Value decrease monotonically as their level of detail increases.

Funkhouser and Séquin propose the incorporation of a “hysteresis” factor into the Benefit heuristic that measures the impairment of user conviction due to fluctuations in the detail levels of objects between successive frames. The Benefit of object representations is reduced by an amount proportional to the difference in detail level from the ones selected for the previous frame, in order to minimize the distraction caused by frequently changing LODs. A similar approach may likewise be incorporated into our algorithm, at no significant cost.

## 8. Experimental Evaluation

An experimental evaluation was conducted to test the effectiveness of the use of group impostors in the hierarchical algorithm. In this experiment, image sequences rendered with the hierarchical algorithm and the non-hierarchical algorithm of Funkhouser and Séquin were presented to each of a group of fifteen volunteer assessors in a controlled environment, and the assessors were asked to evaluate their perceptions of the sequences. The approach selected was the *stimulus comparison method* [2], in which image sequences produced using two distinct techniques are compared in pairs and an index of the relationship between the two sequences of each pair is provided by each assessor. This method produces a distribution of voting indices across the grading scale used, for each assessment pair. The average and standard deviation of each distribution is then taken as an indication of the relationship between each pair of sequences, as perceived by a typical viewer.

Figure 7 shows sample images from the animation sequences used in the experiment. The scene selected for testing consisted of a collection of geodesic domes constructed from cylinders. Each cylinder was provided with six drawable impostors at various levels of detail, plus one *null* impostor corresponding to "no representation", in the non-hierarchical case. These are required, if that algorithm is to be able to satisfy the rendering cost limit when the visible scene complexity is relatively high. The dome objects were each provided with a single group impostor consisting of a low resolution sphere approximation, in the hierarchical case. This allowed that algorithm to replace the individual representations of the cylinder parts of less important domes with single simple dome representations, where appropriate, so that null impostors were not required. The domes were made to rotate around their own axes and to oscillate towards and away from the viewer, so as to force the continual update of the detail levels of all objects by the optimization algorithms.

The Benefit and Cost heuristics supplied were chosen to provide significant but diminishing returns for more complex renderings rather than to function as accurate predictions of the perceptual benefit and rendering cost of object representations. In particular, our Cost heuristic is independent of view-dependent factors such the screen-space size of the object.

The heuristics of the objects in the non-hierarchical description were identical to those of the corresponding leaf objects in the hierarchical one. The heuristics of non-leaf objects in the hierarchical description were selected to provide low returns for group impostors while satisfying the requirements that the Value of an object at any of its explicit representations should be greater than the Values of all of its descendants at their explicit representations and that the Benefit and Cost of an explicitly represented object should be lower than the total Benefit and total Cost of any of its implicit representations (Section 3).

Figures 7(a) and (b) compare the output of the hierarchical and non-hierarchical algorithms respectively, with a rendering cost limit equal to half of the *nominal* cost of the scene. The nominal cost of a scene is the minimum cost of rendering that scene without the use of group or null impostors, and is the value of the rendering cost limit below which the use of null or group impostors becomes necessary in order to meet the rendering cost requirement. Notice that while the nominal cost is in general subject to change from frame to frame, in our experiment it is not since our Cost heuristic is dependent only on impostor complexity.

The extensive use of group and null impostors in (a) and (b) respectively is visible in the form of simplified and missing dome representations. The hierarchical algorithm is able to employ group impostors to ensure a complete representation of the scene, while the non-hierarchical algorithm must resort to the omission of objects in the form of null impostors. The disappearance and reappearance of objects in the non-hierarchical case as null impostors are selected and deselected is particularly apparent when the sequences are animated.

Figures 7(c) and (d) compare the output of the hierarchical and non-hierarchical algorithms respectively with a rendering cost limit equal to twice the nominal cost of the scene. The use of group and null impostors respectively is still visible, even though they are not required in order to satisfy the rendering cost limit. Group and null impostors represent savings that may be used to render more important objects at higher detail.

| Trial | Sequence 1 | Sequence 2 |
|---|---|---|
| 1 | full detail | hier, cost=1920 |
| 2 | non, cost=3840 | hier, cost=3840 |
| 3 | non, cost=1920 | full detail |
| 4 | hier, cost=7680 | non, cost=7680 |

**Table 1:** *The image sequences compared in each of the four experimental trials.* The experiment consisted of four trials, each comparing two sequences. The algorithm and rendering cost limit used for each sequence is shown, where "hier" represents the hierarchical algorithm and "non" the non-hierarchical algorithm. The nominal cost of the scene is 3840.

| Trial | Result |
|---|---|
| 1 | [-1.78, -0.35 ] |
| 2 | [ 0.17, 1.57 ] |
| 3 | [ 0.95, 2.66 ] |
| 4 | [-0.62, 0.76 ] |

**Table 2:** *The results of the perceptual experiment.* The results represent the 95% confidence intervals of the average voter indices for each trial. Voting indices are on a grading scale between -3 and +3, where -3 indicates that the first sequence was much better and +3 that the second was much better.

The experiment consisted of four assessment trials, each of which compared two image sequences. Table 1 shows the image sequences compared in each trial. Image sequences rendered with each algorithm were compared for rendering cost limits equal to the nominal cost and twice the nominal cost. Sequences rendered with each algorithm for a rendering cost limit of half the nominal cost were each compared against a reference sequence rendered at full detail. Table 2 shows the results of the experiment. The assessors on average considered their perceptions of the image sequences

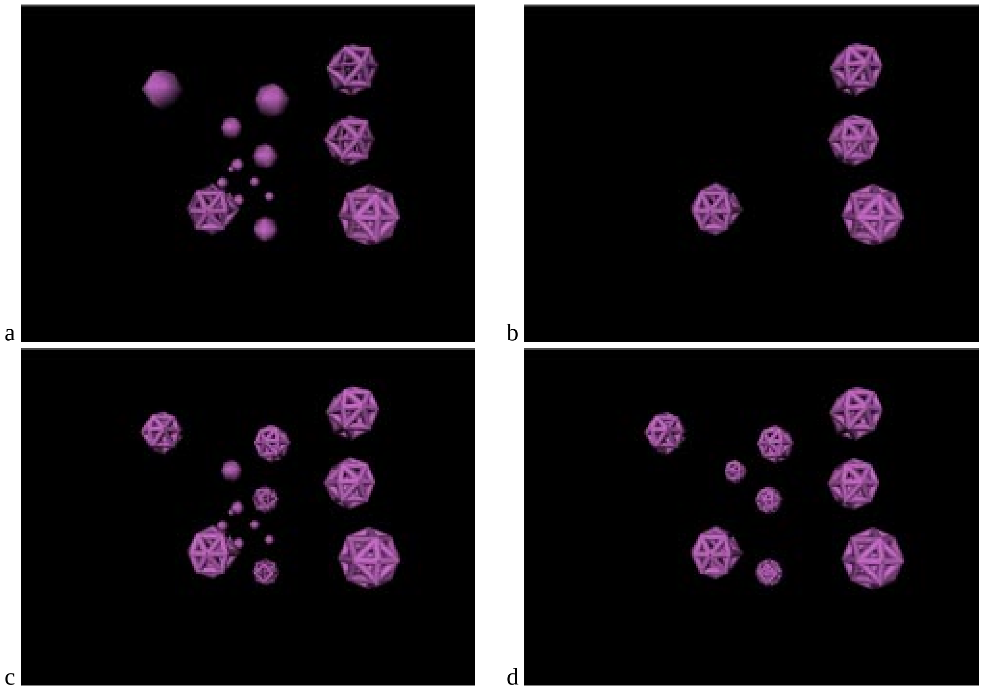

**Figure 7:** *Images rendered with the hierarchical and non-hierarchical LOD optimization algorithms, with various rendering cost limits.* Images (a) and (b) compare the results of the hierarchical and non-hierarchical algorithms with rendering cost limits equal to half the nominal cost of the scene. Images (c) and (d) compare the results of the same algorithms with a rendering cost limit equal to twice the nominal cost of the scene. Note the use of group impostors in (a) and (c) and null impostors in (b) and (d), visible as simplified and missing object representations respectively.

rendered with the hierarchical algorithm to be significantly better than that of those rendered with the non-hierarchical algorithm, in cases where the rendering cost limit was equal to the nominal cost and half the nominal cost, but could distinguish no significant difference for a rendering cost limit equal to twice the nominal cost.

## 8.1. Discussion of Experimental Results

The experiment compares the effects of having group impostors with the effects of not having them, for cases where the rendering cost limit is relatively low (or equivalently, the visible scene complexity is relatively high). In such cases, we expect the non-hierarchical algorithm to resort to the use of null impostors and the hierarchical one to the use of group impostors. In the non-hierarchical case object representations are dropped completely, resulting in “holes” in the representation of the scene. In the hierarchical case the individual representations of groups of objects (or parts) are replaced by shared simpler representations.

The results of the experiment suggest that the fluctuation of domes between group and non-group impostor representations was less distracting to viewers than their disappearance and reappearance due to the use of null impostors as required by the non-hierarchical method of Funkhouser and Séquin.

We expect that no significant difference would be detected for less stringent rendering cost limits, since the effects of the two algorithms become more similar, eventually becoming the same when no group or null impostors are used. Conversely, we expect that the difference would become more apparent as the rendering cost limit became more stringent, or as the relative visible scene complexity increased.

We believe that the results of the experiment demonstrate that the appropriate use of group impostors in the hierarchical algorithm may result in improved perception of animation sequences, in cases where the visible scene complexity is sufficiently high to cause the omission of objects in the non-hierarchical algorithm of Funkhouser and Séquin. Whilst the scene content selected for the experiment does not represent worst-case material for the effects tested, we expect that this should apply to many other typical scenes in which objects are composed of smaller parts.

Notice that the results do not reflect the effects of possible variations in rendering or optimization times between the algorithms, since the sequences were rendered offline and presented at a fixed frame rate. We expect the hierarchical algorithm to show no more significant variation in frame rendering times than that of Funkhouser and Séquin, since both are predictive and limit the predicted rendering times to below some predetermined target frame time. The consistency of the rendering times is dependent for both algorithms on the accuracy of the Cost heuristics supplied.

## 9. Conclusion

We have shown that the level of detail optimization problem for an hierarchical level of detail description is equivalent to a constrained version of the Multiple Choice Knapsack Problem. We presented a hierarchical level of detail optimization algorithm which has the following advantages:

1. It allows the representation of groups of objects by single impostors, so as to afford better renderings of more important objects while providing a complete representation of the visible scene, even when the visible scene complexity is relatively high.
2. Its solution to the hierarchical level of detail optimization problem is guaranteed to be at least half as good as the optimal one.
3. Its worst case time complexity is $O(n \log n)$ but it is incremental and typically completes in only a few iterations.

The algorithm requires pre-generated impostors of all scene objects and assumes that their perceptual benefit and rendering cost are independent of viewing direction. Further work is needed to address these issues, perhaps through a combination of this approach with one in which impostors are generated automatically.

## Acknowledgments

Thanks to the Interactive Systems Department, CWI, Amsterdam for hosting the authors for a working visit that contributed to the completion of this paper, particularly Drs. Tanja van Rij. Thanks to the Dutch Institute for Perception Research for assistance in designing the evaluation experiments, and to Prof. Theo Stewart of the Department of Mathematical Statistics of the University of Cape Town for information on the Knapsack Problem. This work was funded by the Foundation for Research Development of South Africa.

## References

1. E. H. Blake. *Complexity in Natural Scenes: A Viewer Centered Metric for Computing Adaptive Detail.* PhD thesis, Queen Mary College, London University, 1989.

2. The Centre for Communication Interface Research, Department of Electrical Engineering, The University of Edinburgh. *Recommendation 500-4: Method for the Subjective Assessment of the Quality of Television Pictures*, 1990.

3. 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.

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

5. 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.

6. P. S. Heckbert and M. Garland. Multiresolution modeling for fast rendering. In *Graphics Interface '94*, pages 43–50, 1994.

7. 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.

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

9. 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.